UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA
MAESTRÍA Y DOCTORADO EN CIENCIAS E INGENIERÍA

SISTEMA MULTICAPA QUE ESTABLECE COMUNICACIÓN CONFIABLE ENTRE UN SITIO WEB Y UNA RED DE SENSORES INALÁMBRICA BASADA EN EL ESTÁNDAR IEEE 802.15.4

T E S I S

que para obtener el grado de
MAESTRO EN INGENIERÍA

Presenta

ARTURO JESÚS LAFLOR HERNÁNDEZ

DIRECTOR DE TESIS:
DR. JUAN IVAN NIETO HIPÓLITO.

Ensenada B.C. **Noviembre de 2010**

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE INGENIERÍA

UNIDAD ENSENADA

SISTEMA MULTICAPA QUE ESTABLECE COMUNICACIÓN CONFIABLE ENTRE UN SITIO WEB Y UNA RED DE SENSORES INALÁMBRICA BASADA EN EL ESTÁNDAR IEEE 802.15.4

TESIS

Que para obtener el grado de maestría en ingeniería presenta:

Arturo Jesús Laflor Hernández

Aprobada por:

Dr. Juan Ivan Nieto Hipólito
Director de tesis

Drá. Mabel Vázquez Briseño
Miembro del comité

Dr. Juan de Dios Sánchez López
Miembro del comité

M.I. Luz Evelia López Chico
Miembro del Comité

M.C. José Antonio Michel Macarty
Miembro del Comité

Ensenada Baja California, México. Noviembre de 2010

RESUMEN de la Tesis de Arturo Jesús Laflor Hernández, presentada como requisito parcial para la obtención del grado de MAESTRO EN INGENIERÍA. Ensenada, Baja California, México. Noviembre de 2010.

SISTEMA MULTICAPA QUE ESTABLECE COMUNICACIÓN CONFIABLE ENTRE UN SITIO WEB Y UNA RED DE SENSORES INALÁMBRICA BASADA EN EL ESTÁNDAR IEEE 802.15.4

Resumen aprobado por:

Dr. Juan Ivan Nieto Hipólito
Director de tesis

Las redes de sensores inalámbricas debido a la versatilidad que presentan, están encontrando aplicación en cada vez más áreas de la vida cotidiana, se utilizan en la industria, el comercio, la salud y la milicia, entre otras. Este trabajo de tesis presenta un sistema multicapa que permite establecer comunicación confiable entre una red de sensores inalámbrica basada en el estándar IEEE 802.15.4 y un sitio web. El sistema tiene como propósito, brindar servicios de monitorización de signos vitales de pacientes que portan dispositivos con sensores, a través del sitio web. Además permite el control de parámetros, como intervalos de muestreo para la toma de signos vitales, activación y desactivación de muestreo y establecimiento de límites en los que los signos vitales de los pacientes se consideran normales, con el fin de emitir alertas en caso ser necesario. La base del diseño para la construcción de la arquitectura del sistema, se encuentra en el modelo de referencia OSI y la estrategia para el desarrollo de los componentes que le dan funcionalidad, retoma conceptos y mecanismos fundamentales tecnologías como IP-MOVIL, Proxy, el protocolo TCP, el protocolo IP y la tecnología DTN. Los conceptos antes mencionados, se implementan en componentes distribuidos a lo largo de las capas, los cuales permiten establecer comunicación confiable entre los dispositivos de la red de sensores y el sistema de monitorización. Tanto para la transmisión de datos como para los mensajes de control, se utiliza SCNAD_paq, un paquete que se compone de un espacio para datos y campos de control que permiten a los componentes interpretarlo de manera correcta y realizar con él las tareas correspondientes.

Los resultados muestran que mediante este sistema, es factible monitorizar signos vitales de pacientes cuyo estado de salud permite que los intervalos de muestreo sean establecidos en 20 segundos o más, lo cual satisface las necesidades de diversas casas habitación y salas de clínicas, sanatorios y hospitales en los que los requerimientos de cuidado de pacientes son menos demandantes que lo antes mencionado.

Terminos Usados:

DTN: *Redes Tolerantes a Retardos*, **IP:** *Protocolo de Internet*, **IP-MOVIL:** *Tecnología que permite a un dispositivo moverse de una red a otra, manteniendo su dirección IP*, **OSI:** *Interconexión de Sistemas Abiertos*, **Proxy:** *Programa, equipo o dispositivo que hace una acción en representación de otro*, **TCP:** *Protocolo de Control de Transmisión.*

Dedicatoria

A Dios, el gran YO SOY, quien ha sido mi sustento y guía cada día, a quien debo la motivación para crecer en todos los aspectos de mi vida, a Él sea la honra y la gloria por siempre.

A mi amada esposa, quien me ha apoyado física, emocional y espiritualmente durante todo el tiempo de estudio de la maestría, compartiendo con amor cada día, las alegrías y los sin sabores que tiene el proceso. Gracias por amarme, por darme fuerzas y por ayudarme a seguir con la misma energía de principio a fin y gracias por tanto tiempo cedido para el trabajo. Te amo Epochita.

A Fernandito, mi amado hijo, quien ha renovado en mi la responsabilidad y las ganas de ser y hacer mejor en la vida, a quien con ayuda Divina, espero transmitirle por precepto y ejemplo los principios que guíen su existencia. Gracias por tantos ¿porqué?, que me ayudan a seguir buscando respuestas, por hacerme reír sin importar el momento o la situación y por aceptar noblemente tantas veces un "ahorita no, tengo que ir a la escuela."

A mis padres, con amor y respeto, quienes pusieron en mí la idea de estudiar y me enseñaron desde muy temprana edad, la responsabilidad, el respeto, la justicia, el honor y la honestidad, valores necesarios para gozar de libertad y vivir en armonía con los seres que nos rodean.

A mi hermano, por confiar en mí y pensar que puedo lograr las cosas que me propongo: "no es tan fácil como crees, pero gracias por la confianza".

Agradecimientos

Agradezco de manera especial el apoyo de la Universidad Linda Vista, por darme las facilidades de tiempo y recursos económicos para estudiar. Especialmente agradezco a mis colegas de la facultad de sistemas computacionales por asumir las responsabilidades de trabajo, que no es poco, con un integrante menos en el equipo, y a la administración de la Universidad y la Unión Mexicana del Sur por la visión de preparar a sus colaboradores, a fin de servir mejor.

Agradezco así mismo a la Universidad Autónoma de Baja California y a CONACYT por el apoyo económico y por otorgarme el uso de sus instalaciones, equipo de laboratorio, biblioteca y todos los recursos puestos a disposición de quienes seguimos construyendo nuestra vida profesional.

A mi director de tesis, Dr. Juan Ivan Nieto Hipólito, quien planteó las directrices iniciales del trabajo y estuvo supervisando y orientando el desarrollo del mismo, de tal manera que éste concluyera en tiempo y forma. A los maestros de las materias, quienes aportaron conocimiento técnico para fortalecer la experiencia profesional y enriquecer éste trabajo de tesis. Sandra y Paco, gracias por sus críticas y comentarios que ayudaron a dar forma al anteproyecto. Mary y Humberto, gracias por sus comentarios, críticas y valiosos aportes técnicos.

A mis compañeros y amigos de la maestría y doctorado, gracias también por el apoyo extra académico y la convivencia que ayudó muchas veces a liberar tensiones.

También agradezco a mi suegra, cuñados y demás familiares de mi esposa por tener abiertas las puertas de su casa para nosotros, y por el apoyo que nos han dado en diversas cosas y circunstancias.

Vienen además a mi mente, familiares y amigos que han contribuido con mi formación en todas las etapas de mi vida, así como otras personas que han tenido simpatía, amabilidad y amistad para un servidor y para mi familia durante nuestra estancia en esta ciudad, ¡Gracias por todo!.

No menciono por nombre a todas las personas a las que deseo agradecer, porque la lista sería muy extensa y lamentaría pasar por alto el nombre de alguien.

Índice general

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción

Las Redes de Sensores Inalámbricas (WSN, siglas en inglés), son redes organizadas en topología Par a Par (P2P), estrella o una combinación de ambas, están construidas por pequeños nodos sensores que poseen cierto nivel de cómputo y memoria para el procesamiento y almacenamiento temporal de datos, los cuales son distribuidos en espacios geográficos diversos para percibir el mundo físico. Los datos que captan del ambiente u objetos observados, son transmitidos a través de un canal inalámbrico de comunicación entre nodo y nodo, teniendo la posibilidad de formar rutas para cubrir grandes superficies hasta llegar a una estación base donde serán almacenados y procesados de tal manera que produzcan información útil.

Hoy se puede encontrar aplicaciones de redes de sensores inalámbricas en diversas áreas, entre las cuales [29] y [13] mencionan las siguientes:

- Monitorización de objetos ubicados en ambientes remotos y hostiles.
- Búsqueda de seres vivos en casos de desastre.
- Proyectos para el cuidado de la salud.
- Vigilancia en el campo de batallas militares.
- Monitorización de tráfico automovilístico.

En los últimos años la aplicación de sistemas basados en redes de sensores se ha extendido en áreas como la preservación y cuidado de patrimonios culturales. Al respecto [22] presenta

una aplicación para vigilancia de reliquias en un museo y [8] describe la manera en que mediante una red de sensores se puede tener información del estado de un edificio (en este caso, la Torre Aquila en Italia) a fin de darle mantenimiento en el momento oportuno. La proliferación de aplicaciones como estas, muestran por qué en el año 2003, [25] listó las WSN entre las 10 tecnologías que cambiarían el mundo en los años futuros.

Este trabajo presenta el desarrollo de un sistema que comunica una red de sensores inalámbrica basada en el estándar IEEE 802.15.4 con un portal web. La red de sensores recaba signos vitales de pacientes y los envía a una base de datos. Por otra parte, las personas encargadas del cuidado de la salud de dichos pacientes, monitorizan los datos através del portal web y establecen parámetros que controlan el comportamiento de los dispositivos en cuanto a intervalos de tiempo para las muestras y verificación de alertas.

1.1. Antecedentes

Las WSN normalmente incluyen en su infraestructura nodos sensores, nodos coordinadores y una estación base que concentra la información de toda la red y que servirá de puente para establecer la comunicación con usuarios finales o con otros sistemas a los que preste servicios. La organización de la red, la forma en que la información se transporta a través de la misma, así como el proceso y tratamiento de los datos, están definidos por los sistemas que residen en los dispositivos antes mencionados. De esta manera, se puede decir que el comportamiento de la red depende de los sistemas que la conforman. Al revisar la literatura se encontró que [21] hace un estudio de los diferentes sistemas que hasta 2006 hicieron posible la implementación de WSN. A partir de ese trabajo, se puede obtener la clasificación que muestra la figura 1.1, la cual se describe a continuación:

Arquitectura: La arquitectura define las tareas que se realizarán en los diversos dispositivos de la red. Las redes con arquitectura jerárquica procesarán tareas completas en cada dispositivo dependiendo de su capacidad y su importancia dentro de la red, por ejemplo, un nodo coordinador porcesará tareas de mayor importancia que un nodo sensor, y la estación base procesará las tareas de mayor importancia dentro de la red, mientras que

Figura 1.1: Clasificación de sistemas para WSN

en la arquitectura distribuida una tarea puede ser procesada por diferentes nodos y la asignación de la misma o parte de ella depende más de la disponibilidad de los nodos que del tipo de nodo que se trate. Por otra parte, en la arquitectura centralizada los nodos sensores se encargan unicamente de transmitir los datos que captan del medio físico, los coordinadores de organizar las celdas que les corresponden y de transmitir los datos que reciben de los nodos sensores y es la estación base quien tiene toda la responsabilidad del procesamiento de los datos.

Funcionalidad Las WSN al igual que todas las redes inalámbricas son proclibes a fallos, lo cual hace que los sistemas, trabajen para satisfacer las necesidades por las cuales fueron creadas estas redes, además de lidiar con solucionar los problemas de congestión, pérdida de enlace y corrupción de los datos, entre otros que se presenten. Cuando los sistemas después de sucedidos los eventos responden para satisfacer las necesidades dependiendo de la naturaleza del evento, se dice que tienen un comportamiento reactivo, si mediante mecanismos implementados en los diferentes dispositivos, se anticipan a corregir los posibles errores antes de que los eventos se disparen, se dice que tienen un comportamieto proactivo y cuando su prioridad es mantener a la red estable, independientemente de como lo logren y el momento en que lo hagan, se consideran sistemas de detección de fallos.

Tipo Los sistemas se clasifican por tipo dependiendo de su prioridad en los servicios que presten. Por ejemplo si un sistema permite interactuar a personas con la red de sensores, será una aplicación de usuario, mientras que si permite a un experto revisar el estado de

la información en cualquier punto de la red que ésta se encuentre, será una herramienta de depuración.

Factor prioritario en el diseño Dependiendo de las necesidades que la red tenga que satisfacer y las condiciones en las que vaya a operar, puede haber uno o más factores que incidan de manera especial en el desarrollo de los sistemas. Por ejemplo, si una red tendrá sus nodos sensores en un ambiente hostil y peligroso para el ser humano, será trascendente cuidar el consumo de energía puesto que el reemplazo de baterías cuando la carga se agota, no es factible. Por otra parte si se sabe que la red puede llegar a crecer demasiado, el diseño debe incluir de manera prioritaria, mecanismos de escalabilidad, etc.

Así mismo, al revisar [19, 27] y [9] se puede identificar que los sistemas de los que venimos hablando, pueden ubicarse en tres grandes áreas: Sistemas operativos, sistemas middleware y sistemas de aplicación de usuario final. Sus características están determinadas por la función que cumplen en el proceso de comunicación de los datos de extremo a extremo y pueden describirse como sigue:

Sistemas Operativos: Ayudan a acceder a los recursos físicos que ofrece el hardware de la red y permiten tener cierto control sobre el mismo.

Sistemas Middleware: Se encargan de transportar información de las capas bajas donde trabajan los sistemas operativos, hacia las capas altas donde trabajan las aplicaciones de usuario final.

Sistemas de usuario final: Ofrecen servicios de presentación de información a manera de listados de datos, emisión de alertas, gráficas, etc.

En la implementación de un sistema que utiliza como base de adquisición de datos una WSN, intervienen varios subsistemas que se ubicarán en alguna clasificación de las mencionadas con anterioridad. Además deben mantener una misma política de comportamiento en cuanto al tratamiento que dan a la información que procesan y transmiten, la cual puede ser:

Data-Centric: El foco principal de atención son los datos, por ejemplo una implementación cuyo interés principal es conocer el promedio de temperatura de una determinada extensión de terreno o de cierto grupo de personas.

Zone-Centric: El foco principal de atención es la zona de donde proviene el dato, así por ejemplo, si se tienen dispositivos diseminados en un área geográfica y se requiere saber la temperatura de esa área, basta con tener la lectura de un dispositivo (la que se obtenga primero) para satisfacer la necesidad del sistema.

Object-Centric: Esta política centra su atención en el individuo, aquí cada objeto observado porta un dispositivo con sensores acoplados y las lecturas que se hagan deben viajar a través de las diferentes capas y dispositivos de la red con un identificador que haga evidente al sistema, el objeto del que proceden los datos. Como ejemplo podemos citar la monitorización de objetos valiosos en exposiciones [22].

People-Centric: Es el nombre que se le da a la política Object-centric cuando los objetos observados son individuos que tienen movilidad. Por ejemplo dispositivos que se acoplen a animales en peligro de extinción para saber en cualquier momento el lugar donde se encuentran, o los dispositivos que se acoplan a pacientes para hacer lecturas de sus signos vitales.

Las características del sistema que constituye este trabajo de tesis son:

- Arquitectura jerárquica puesto que las diversas tareas se ejecutan de manera completa en los diferentes dispositivos a los que han sido asignadas dependiendo de la cantidad de recursos que consumen y la capacidad de memoria y energía con que éstos cuentan.

- Es un sistema reactivo debido a que ejecuta procesos en respuesta a eventos que ocurren en diferentes partes de la red.

- Con base en las clasificaciones que hace [19] se puede decir que se trata de un sistema middleware que se complementa con una aplicación de usuario final como prueba de concepto.

- Con base en el tratamiento que se da a la información, está ubicado dentro de la política people-centric.

1.2. Planteamiento del problema

Como se mencionó en la sección 1, el presente trabajo está enfocado al desarrollo de un sistema que preste servicios de monitorización y gestión a una red de sensores, la cual se implementará como medio de adquisición y transmisión de datos en una infraestructura e-salud semejante a la que ilustra la figura 1.2.

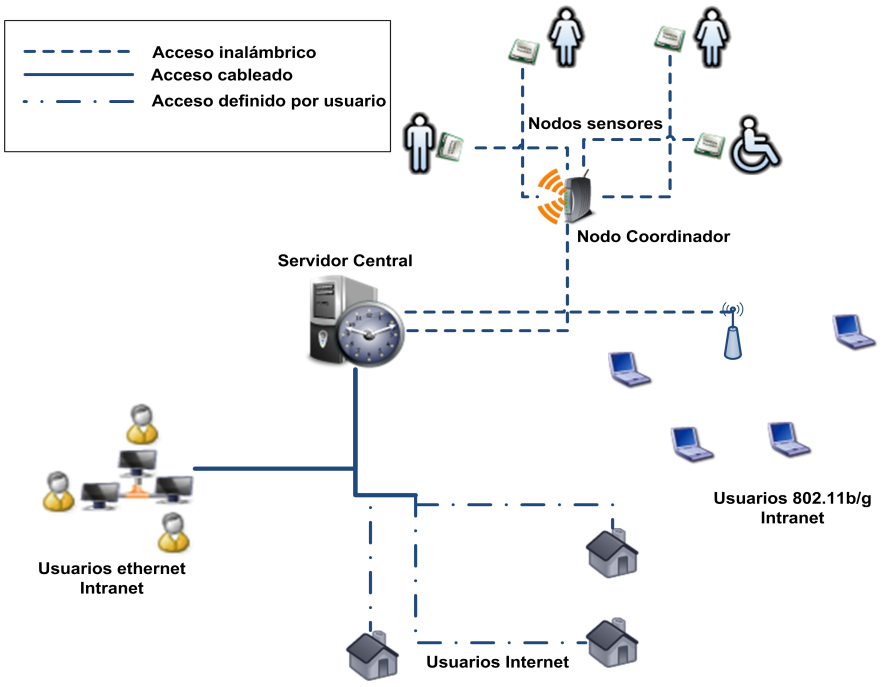


Figura 1.2: Esquema de infraestructura e-salud

Dentro de las características generales que resaltan de la infraestructura, podemos mencionar que: Los datos corresponden a los signos vitales de pacientes y se transmiten en intervalos de tiempo determinados por los usuarios o en el momento que se detectan valores fuera de los rangos establecidos como normales. Estos datos deben transportarse desde los nodos sensores hasta los usuarios finales a través de redes con diferentes estándares de comunicación. Debe considerarse que los usuarios finales esperan confiabilidad en la información que leen, así como

tiempos de entrega consecuentes. Por otra parte, se requiere lectura del estado de los dispositivos que conforman la red (nodos coordinadores y nodos sensores) tomando en cuenta que los nodos sensores pueden tener movilidad[1] en el área de cobertura total de la infraestructura.

1.2.1. Preguntas de investigación

En este marco, surgen las siguientes preguntas de investigación que serán respondidas en el capítulo cinco, tomando como base los resultados de las pruebas que se presentan en el capítulo cuatro.

- ¿Es factible dar seguimiento al estado de salud de pacientes en un hospital, mediante la recolección de sus signos vitales con una WSN basada en el estándar IEEE 802.15.4 y la monitorización de los mismos desde un sitio web?
- ¿Con cuanta confiabilidad[2] se puede realizar la comunicación entre el dispositivo portado por el paciente y el sistema manipulado por el cuidador?

1.2.2. Objetivos

Objetivo general

Desarrollar un sistema de comunicación confiable para monitorización de pacientes, que permita configuración de parámetros y lectura de datos provenientes de dispositivos que forman una WSN bajo el estándar 802.15.4, a través de un sitio web.

Objetivos específicos

1. Identificar los requerimientos que la infraestructura e-salud especifica en cuanto a la monitorización de pacientes y configuración de los dispositivos que recaban la información.
2. Seleccionar las herramientas de desarrollo con base en los requerimientos del sistema y las especificaciones del estándar IEEE 802.15.4.

[1] En el contexto de este trabajo, **movilidad** se refiere a que los nodos no están fijos a un nodo coordinador, sino que pueden estar fluctuando entre diferentes coordinadores de manera no determinística.

[2] **Confiabilidad:** Garantía de entrega de paquetes completos y en el tiempo requerido, de la fuente al destino.

3. Diseñar la arquitectura general de la aplicación.

4. Desarrollar el sistema de comunicación entre la red de sensores y el portal web.

5. Desarrollar el prototipo de aplicación web de usuario final.

1.2.3. Justificación

La realización de esta investigación se justifica por los siguientes motivos:

Desde el punto de vista práctico y debido a que la afirmación hecha en [2] acerca de no haber un estándar que comunique una red 802.15.4 con redes TCP/IP sigue vigente a la fecha que se escribe este documento, aporta un prototipo de aplicación que permitirá la monitorización de datos, la monitorización de estado y la configuración de parámetros de los dispositivos de una WSN mediante una aplicación web, basando su diseño en los requerimientos que debe cumplir la infraestructura e-salud que está desarrollando el cuerpo académico de Telemática de la UABC campus Ensenada.

Desde el punto de vista teórico el estudio es relevante porque aborda el desarrollo de aplicaciones para WSN desde una perspectiva poco considerada en la literatura revisada:

1. Las aplicaciones actuales ponen como prioridad el ahorro de energía y en esto basan su diseño. En este estudio si bien es cierto que se considerará cuidar la energía de las baterías, el foco central es la entrega de información íntegra y en tiempo, razón por la que se retomarán conceptos del Protocolo de Control de Transmisiones (TCP, siglas en inglés) y de la tecnología de Redes Tolerantes a Retardos (DTN, siglas en inglés).

2. El sistema se desarrollará para una red people-centric, por lo que se retomará el concepto básico del Protocolo de Internet (IP, siglas en inglés) y la tecnología IP-Móvil, al asignar una etiqueta que identifique al dispositivo independientemente de la celda a la que pertenezca dentro de la infraestructura.

Por otra parte en la literatura revisada y en el contexto de nuestro conocimiento, no se ha encontrado aplicaciones disponibles que se puedan adaptar a los requerimientos que se deben

cubrir en la infraestructura e-salud de la que se ha venido hablando, lo que hace pensar que [6] tenía razón al afirmar que:

> "La optimización de las propiedades de funcionamiento y operación de las WSN quizá requieren una única solución para cada problema de aplicación."

1.2.4. Delimitación

Con base en las características y clasificaciones de las aplicaciones que se han desarrollado para la interacción con las WSN y de las cuales se ha hecho mención en las partes introductorias de este capítulo, se propone el marco de trabajo en el cual se realiza esta investigación.

1. El proyecto centra su atención en el desarrollo de un sistema de transporte y tratamiento de datos generados por sensores acoplados a dispositivos que forman una WSN, desde que éstos se encuentran en la memoria de los dispositivos, hasta su monitorización en un portal web.

2. Para el prototipo de aplicación web se considera el desarrollo de interfaces para:

   - Configuración de parámetros de los dispositivos de la red, de tal manera que se comporten como los usuarios lo requieren. Por ejemplo, establecer los tiempos de muestreo o establecer valores límite para los signos vitales con el fin de emitir alertas si éstos son excedidos.
   - Monitorización de los signos vitales en tiempo real.
   - Monitorización de alertas.
   - Consulta de signos vitales del registro histórico.

3. Las tareas relacionadas con el control de los sensores y la forma en que éstos capturan los datos quedan fuera del alcance de este proyecto.

4. Las tareas de administración y acceso al hardware de los dispositivos, así como el control de acceso al medio, estará a cargo de una plataforma de software que presta esos servicios con base en los lineamientos del estándar IEEE 802.15.4.

1.3. Metodología

Para cumplir con el objetivo se propone una arquitectura funcional de tipo jerárquica que base sus criterios de comunicación en los conceptos básicos de IP-Movil, TCP, DTN y Proxy, como se describe a continuación:

Arquitectura Funcional Jerárquica:

La figura 1.3 ilustra la proporción de trabajo que realiza cada equipo que integra un sistema con arquitectura jerárquica. La cantidad de trabajo asignada depende de sus recursos de energía, memoria, y capacidad de procesamiento. En este proyecto, las tareas que hacen confiable la comunicación entre los extremos del sistema son realizadas por: Nodos sensores (NS), dispositivos portados por los pacientes; nodos coordinadores (NC), dispositivos que forman redes tipo estrella al asociar cierta cantidad de NS, y un servidor que concentra la información en la base de datos y administra el sitio web.

Figura 1.3: Arquitectura jerárquica

Criterios de comunicación:

Se describe a continuación los conceptos que se considerarán de cada una de las tecnologías tomadas en cuenta para lograr la comunicación entre las redes.

IP-Movil: El concepto que retomamos de IP-Movil es que cada NS posee una dirección única que se mantiene independientemente del NC al que se asocie. El NS puede moverse por un

espacio geográfico con cobertura y saltar de un NC a otro, manteniendo el identificador con el que es reconocido en la base de datos. Para lograr esto, el NS mantiene siempre dos direcciones, una externa asignada al momento de cargarle la aplicación y una interna que adquiere cuando se asocia a un NC.

TCP: Se utilizará el concepto básico de transmisión de mensajes de reconocimiento (ACK, abreviatura en inglés) por cada paquete transmitido, con el fin de asegurar su entrega en el otro extremo. Así mismo se retoma el concepto del "three hand check", (aunque con algunas variantes), cuando se trata de solicitar información en tiempo real de algún dato desde el sitio web a un NS.

DTN: El concepto básico de DTN, tiene que ver con el almacenamiento temporal de la información [9] en los nodos cuando existen problemas para transmitir de un dispositivo de red a otro, de tal manera que los datos se mantienen en el último dispositivo que los recibió, hasta que pueden ser retransmitidos. En la propuesta de este trabajo se considera la realización de esta actividad en los NS y NC de con el propósito de aumentar la confiabilidad en la entrega de los datos.

Proxy: La idea de un proxy es la de ser intermediario entre los proveedores de servicios de información y los consumidores de los mismos. La propuesta incluye un servidor de base de datos donde se almacena la información y un sitio web para acceder a ella, esta parte del sistema es el intermediario entre la red de sensores y el usuario final.

La figura 1.4 muestra el esquema de comunicación según fue descrito en esta sección.

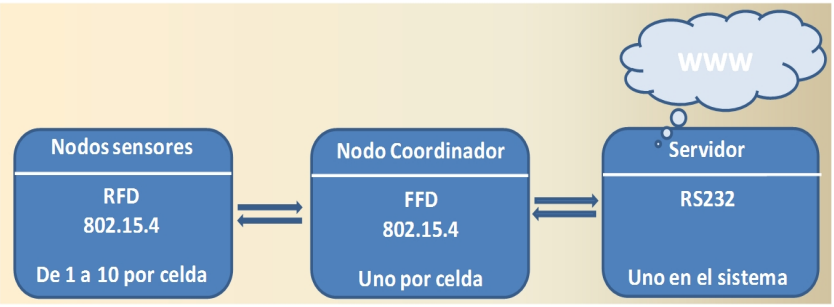


Figura 1.4: Esquema de comunicación

1.3.1. Ruta crítica

El plan que se trazó para medir el avance durante el desarrollo del proyecto está representado en la figura 1.5.

Figura 1.5: Ruta crítica

El proyecto inicia con la revisión de la literatura en tres aspectos que dan el fundamento teórico sobre el que está basado este trabajo:

- Aplicaciones pasadas y presentes, propuestas de trabajo futuro y conceptos generales sobre las redes de sensores inalámbricas.
- Redes de sensores basadas en el estándar IEEE 802.15.4.
- Estado del arte de sistemas desarrollados para redes de sensores inalámbricas.

Aunque el énfasis en el repaso de la literatura se hace en esta primera etapa, las flechas blancas en el esquema indican que este ejercicio se mantiene durante todo el tiempo en el que se desarrolla el proyecto, con el fin de enriquecer el trabajo, durante todo el proceso de desarrollo.

Durante el repaso de la literatura se ubican herramientas de software que están disponibles para usar en el desarrollo de sistemas para redes de sensores inalámbricas, tales como sistemas

operativos, sistemas middleware, plataformas de desarrollo y lenguajes de programación en los que se puede trabajar. Así que como segundo paso se hace la selección de las herramientas que se utilizarán para realizar la aplicación. El tercer paso consiste en realizar la descripción de los requerimientos que debe satisfacer el sistema de comunicación entre los extremos de la aplicación y también se detallan las necesidades que a vista de los usuarios debe suplir el sistema de monitorización del sitio web. Como cuarto paso, se tiene la especificación del diseño funcional y de la arquitectura en la cual estará enmarcado el sistema, para finalmente terminar, empleando una metodología de software que ayude a cumplir con la realización del proyecto en cuatro fases: Análisis, diseño, desarrollo y pruebas en un entorno iterativo.

El seguimiento de esta ruta crítica y su cumplimiento en los tiempos propuestos, da como resultado tener el sistema terminado para cumplir el objetivo general y poder responder a las preguntas de investigación con base en las pruebas realizadas y los resultados obtenidos.

Capítulo 2

Marco Teórico

2.1. Introducción

Las necesidades de cuidado de pacientes en edades avanzadas con padecimientos degenerativos como Alzheimer, Parkinson y otros similares están avanzando de manera considerable alrededor del mundo. Esto hace necesario que cada vez se requieran más personas que se dediquen al cuidado de estos pacientes. Los cuidadores pueden ser profesionales del área de la salud contratados para realizar este trabajo, o lo que sucede en la mayoría de los casos sobre todo en países en vías de desarrollo, el familiar más cercano al paciente se convierte en su cuidador.

Por otra parte, las ciencias computacionales, electrónicas y de telecomunicaciones, en los últimos años han centrado su atención y realizado muchos estudios en el área del cómputo ubicuo o pervasivo, se han hecho sistemas que mediante el uso de dispositivos acoplados a las personas o a las áreas donde éstas se mueven, permiten su monitorización remota y continua. Con esto se da a profesionales de la salud y familiares cercanos la posibilidad de estar informados de la salud del paciente en todo momento y actuar en consecuencia, aun cuando no se encuentren cercanos a ellos de manera física.

La idea del cómputo ubicuo sigue siendo hoy día la visión que Mark Weiser tuvo y que describió en su artículo denominado "La computación para el siglo 21" de septiembre de 1991, donde propone la creación de entornos repletos de computación y capacidad de comunicación mediante tecnología que pase desapersibida para las personas que interactúan en dichos am-

bientes y que sin embargo hagan uso de ésta para el logro de sus propósitos. En el tiempo que Weiser escribió se veía demasiado lejos la realización de esa visión debido en gran parte a la inexistencia del equipo tecnológico que ayudara a cumplirla. Sin embargo, hoy en día, se cuenta con computadoras portátiles muy pequeñas, teléfonos celulares con capacidad de cómputo y comunicación potente, redes inalámbricas formadas por dispositivos con sensores diminutos de costo relativamente bajo, los cuales permiten el desarrollo de aplicaciones que incluyen las características y propiedades que el cómputo ubicuo o pervasivo persigue [4]. Weiser propuso que:

1. La tecnología computacional debe estar integrada a los objetos, tareas y entornos cotidianos.

2. La introducción de computación a estos objetos, tareas y entornos no deben interferir en las actividades para las que son utilizados, además de proporcionar un uso más cómodo, sencillo y útil de los objetos.

3. Los objetos que participen como parte del sistema deben estar dotados de capacidad de cómputo y comunicación, y no solo con el usuario sino con los demás objetos integrados que haya a su alrededor.

4. Los objetos o dispositivos deben tener además, capacidad de memoria y deben poder utilizar esta memoria para una mejor interacción con el resto de dispositivos.

5. Los objetos y dispositivos deben ser sensibles al contexto, es decir, se adaptan a la situación geográfica, a las condiciones de cómputo, comunicación y memoria de otros dispositivos que hay a su alrededor. También deben adaptarse a las necesidades de los usuarios y actuar en congruencia con las demandas del entorno que los rodea.

6. Los dispositivos y objetos deben reaccionar a eventos que perciben del entorno mediante sensores o a través de la interacción con otros dispositivos.

Dentro de las redes inalámbricas que hacen posible la creación de los escenarios pervasivos están las de cobertura amplia, como son las redes de telefonía celular y las redes WiMax, y por

otra parte están las de mediana y corta cobertura como WiFi, Bluetooth, 802.15.4 y UBW que al día de hoy se presentan en la literatura como las más utilizadas y de mayor proyección a futuro para el desarrollo de sistemas pervasivos en el área de la salud.

Este trabajo pertenece al ámbito de las tecnologías de corto alcance, de allí que este capítulo quedará organizado de la siguiente manera: En la siguiente sección se describirá brevemente las tecnologías mencionadas en el párrafo anterior, posteriormente se hará una descripción de los sistemas de software que proveen plataformas y herramientas para el desarrollo de nuevos sistemas que utilizan las tecnologías de corto alcance como medio de comunicación y se terminará con la descripción en detalle de la pila de Control de Acceso al Medio (MAC, siglas en inglés) de Freescale que ha sido elegida como plataforma y herramienta de desarrollo, a partir de la cual se construye la red de sensores inalámbrica que es la base de la infraestructura para este trabajo.

2.2. Tecnologías inalámbricas de corto alcance

2.2.1. Bluetooth

Sus orígenes se remontan al año de 1994 cuando la compañía sueca Ericsson pretende desarrollar una interfaz radio de bajo costo y consumo de energía, principalmente para la interconexión de teléfonos móviles. Más adelante en 1998 se funda el SIG (Grupo de Interés Especial) de Bluetooth que se ocupa desde entonces del desarrollo de esta tecnología y está integrado por más de 9000 miembros que lideran en el mundo las áreas de telecomunicaciones, informática, industria automotriz, música, confección, automatización industrial, y tecnología de redes [26].

Las especificaciones de Bluetooth están definidas en el estándar IEEE 802.15.1 dentro de las que encontramos que la capa física de radio de Bluetooth opera en la banda de 2.4 GHz libre para la banda de frecuencia Industrial, Científica y Médica (ISM, siglas en inglés) y tiene 79 canales de radiofrecuencia (RF) para transmitir. Emplea un transmisor de salto de frecuencia para contrarrestar las interferencias y la pérdida de intensidad y cuenta con un gran número de portadoras de espectro ensanchado por salto de frecuencia (FHSS, siglas en inglés). La

tasa de transferencia de símbolos es de 1 MS/s (mega símbolos por segundo), que admite una velocidad de transmisión de 1 Megabit por segundo (Mbps) en el modo de transferencia básica y una velocidad de transmisión aérea total de 2 a 3 Mbps en el modo de transferencia de datos mejorada. Estos modos se conocen como transferencia básica y transferencia de datos mejorada, respectivamente. Por otra parte el rango de alcance de Bluetooth es de 10 metros con un consumo de 1 mW, aunque dicho rango puede alcanzar los 100 metros si se aumenta la potencia de transmisión a 100 mW.

Las redes formadas por dispositivos Bluetooth se conocen como piconets, las cuales consisten en dos o más dispositivos que ocupan un mismo canal físico (lo que significa que los dispositivos están sincronizados a un mismo reloj y secuencia de saltos). El reloj de la piconet es idéntico al reloj del dispositivo conocido como maestro de la piconet y la secuencia de saltos es derivada del reloj maestro y de la dirección del dispositivo maestro. Los demás dispositivos sincronizados son conocidos como esclavos en la piconet. Estos términos de maestro y esclavo están vigentes únicamente para una piconet en funcionamiento y sirven sólo para describir los roles que cada dispositivo juega. Es posible que exista más de una piconet en un mismo lugar, teniendo cada una, un número de canal independiente, lo que significa un diferente dispositivo maestro con un reloj y una secuencia de saltos independientes. Un dispositivo puede participar como esclavo en en dos o más piconets al mismo tiempo pero no como maestro, debido a que la sincronización del reloj de la piconet depende del reloj del dispositivo maestro. Cuando un dispositivo participa en más de una piconet al mismo tiempo se dice que está envuelto en una scatternet, lo que necesariamente implica funciones de enrutamiento habilitadas en los dispositivos.

2.2.2. WiFi

La tecnología WiFi que a su vez es una marca, está basada en el estándar IEEE 802.11 bajo la dirección de la alianza del mismo nombre. Se ha convertido en la alternativa inalámbrica para las redes de área local. Una red WiFi generalmente está constituida por un punto de acceso inalámbrico (WAP, siglas en inglés) conectado a la red Ethernet por un puerto denominado WAN (*World Area Networks*) y los dispositivos WiFi a los que pretende darles cobertura a

través del medio inalámbrico (aunque también dispone regularmente de una cantidad de puertos ethernet para dar cobertura a dispositivos que así lo requieran). Las redes WiFi trabajan en la banda ISM y sus características particulares de frecuencia y tasas de transferencia entre otras varían dependiendo de la edición del estándar al que pertenezcan. A continuación se describen las características principales de rendimiento de las ediciones más comunes de la norma IEEE 802.11 al momento de escribir este documento:

- 802.11a: Tiene alcance de 30 metros con una tasa de transferencia que puede alcanzar los 54 Mbps a una frecuencia de 5.4 GHz y como técnica de modulación emplea la Multiplexación por División de Frecuencias Orthogonales (OFMD, siglas en inglés).

- 802.11b: Tiene un alcance de aproximadamente 100 metros con una tasa de transferencia de 11 Mbps. Utiliza tres canales de radio en un espacio de 22 MHz los cuales están centrados en los 2412, 2437 y 2462 MHz, lo que restringe a tres, la cantidad de redes que pueden coexistir en un mismo espacio. Utiliza el Espectro Esparcido por Secuencia Directa (DSSS, siglas en inglés) como técnica para evitar las pérdidas por interferencia. No es compatible con 802.11a

- 802.11g: Este estándar es compatible con 802.11b. Trabaja en la banda de los 2.4 GHz y utiliza como técnica de modulación OFMD, teniendo un rango de alcance de 100 metros.

- 802.11n: En teoría el estándar 802.11n eleva la tasa de transferencia a los 600Mbps y hace uso simultaneo de las bandas 2.4 GHz y 5.4 GHz. Esto último lo hace compatible con dispositivos basados en ediciones anteriores del estándar. Se espera que el alcance de operación de las redes con este estándar sea mayor gracias a la tecnología Múltiples Entradas-Múltiples Salidas (MIMO, siglas en inglés), la cual permite el uso de varios canales a la vez para transmitir y recibir datos debido a que permite incorporar varias antenas. Además de la velocidad y compatibilidad que ya pueden considerarse ventajas en esta edición del estándar, ofrece mayor robustez, mejor rendimiento, más seguridad y capacidades de Calidad en el Servicio lo que es atractivo para cualquier administrador y desarrollador de tecnologías de la información [3].

2.2.3. UWB

UWB difiere substancialmente de las tecnologías convencionales de RF y espectro esparcido, tales como Bluetooth y WiFi. UWB usa una banda muy amplia del espectro de radiofrecuencia para transmitir los datos. UWB es capaz de transmitir más datos en un período dado de tiempo que todas las otras tecnologías tradicionales.

UWB hace uso único del nuevo espectro de radio frecuencia recientemente legalizado. UWB puede usar frecuencias de 3.1 GHz a 10.6 GHz (una banda de más de 7 GHz de ancho). Cada canal de radio puede tener un ancho de banda de más de 500 MHz, dependiendo de su frecuencia central. Para permitir el uso de un ancho de banda tan amplio la Comisión Federal de Comunicaciones (FCC, siglas en inglés) establece severas restricciones de potencia en la emisión de la señal. Tomando esto en consideración, los dispositivos UWB pueden hacer uso de un amplio espectro de frecuencia mientras no emitan la suficiente energía para ser notados por los dispositivos cercanos que usan anchos de banda más estrechos. Esta forma de compartir el espectro permite a los dispositivos obtener un alto rendimiento en la transferencia de datos pero dentro de un área muy pequeña (hasta 110 Mbps en un rango de aproximadamente 10 metros). Los bajos requerimientos de poder exigidos por las UWB hacen factible el desarrollo de dispositivos que implementen radios con UWB. Considerando las características de baja energía para las transmisiones, alta tasa de transferencia de datos y rango limitado, UWB se posiciona en el mercado como una tecnología para el desarrollo de Redes Inalámbricas de Área Personal (WPAN, siglas en inglés) de alta velocidad.

2.2.4. Redes bajo el estándar IEEE 802.15.4

Las redes de sensores inalámbricas que operan bajo el estándar IEEE 802.15.4 son una alternativa cada vez más viable para la implementación de sistemas WPAN, debido a la gran cantidad de dispositivos (sensores y tarjetas provistas de unidad de procesamiento, memoria y radio) de bajo costo que se han creado en los últimos años. Estas redes operan en tasas de 250 kbs, 100 kbs, 40 kbs y 20 kbs en 16 canales de la banda de 2.4 Ghz, 10 canales en la banda de 915 Mhz y 1 canal en la banda de 868 Mhz. Se pueden construir redes en topología estrella y p2p

utilizando el mecanismo de Múltiple Acceso por Detección de Portadora Evitando Colisiones (CSMA/CA, siglas en inglés) como mecanismo de acceso al medio. También, permiten el uso opcional de Ranuras de Tiempo Garantizadas (GTS, siglas en inglés) para asegurar períodos libres de contienda a nodos que lo soliciten cuando sea necesario. El rango de alcance del radio puede abarcar hasta 100 metros con línea de vista y en su máxima potencia. Los objetivos que se persiguen para la implementación de esta tecnología son: fácil instalación, transferencia confiable de datos, rango pequeño de operación, bajo costo de recursos y duración razonable de la energía, manteniendo al mismo tiempo un protocolo sencillo y flexible.

2.3. Sistemas para WSN

En la literatura revisada se identifican principalmente tres tipos de sistemas que trabajan de manera interdependiente para colaborar en la gestión eficiente de la información proveniente de una WSN:

- ***Sistemas Operativos:*** Ayudan a acceder a los recursos físicos que ofrece el hardware de los dispositivos diseñados para operar como nodos en una WSN y a tener cierto control sobre el mismo.

- ***Sistemas de Usuario Final:*** Ofrecen servicios de monitorización de la información a través de listados de datos, emisión de alertas, gráficas, etc. y también integran interfaces que permiten la comunicación con los dispositivos a fin de configurarlos o enviar mensajes que produzcan ciertos eventos a los que reaccionen realizando alguna tarea a conveniencia de los usuarios.

- ***Sistemas Middleware:*** Se encargan de comunicar la red de sensores con las aplicaciones de usuario final y dependiendo del tamaño de la red, cantidad de saltos desde el dispositivo más lejano hasta el concentrador de la información, prioridades de la red en cuanto confiabilidad, tiempos de respuesta, etc. Estos sistemas implementarán protocolos de enrutamiento, protocolos de transporte, estrategias de calidad en el servicio (QoS, siglas en

inglés) y muchas otras herramientas que permitan cumplir las características identificadas en los requerimientos funcionales de la WSN.

Las siguiente parte del capítulo describe los sistemas operativos y sistemas middleware con más aceptación entre los desarrolladores de software para infraestructuras basadas en WSN. Además, se presenta las alternativas que Freescale ofrece para el desarrollo de este tipo de sistemas y aclara porqué éstas alternativas se deben considerar como una opción viable para el desarrollo de sistemas para WSN.

2.3.1. Sistemas Operativos

El control de un nodo en una WSN es realizado por el sistema operativo [19], el cual se ejecuta en el CPU implementando servicios que incluyen la programación de tareas, la comunicación entre procesos (IPC, siglas en inglés), el control de la escala de voltaje, el control de activación e inactivación del dispositivo y el manejo de memoria estático [14] para la mayoría de sistemas operativos embebidos. Esto se debe a las medidas de confiablidad que se requieren y la limitación de recursos que existe, lo cual trae como consecuencia tamaños de cola fijos que se refleja en semántica compleja al momento de programar tareas comunes como vaciar el búfer de datos a la pila de protocolos. Los sistemas operativos además, pueden proveer interfaces para acceder y controlar periféricos, las cuales están típicamente asociadas a componentes de software que trabajan por capas dando funcionalidad más sofisticada al dispositivo. A continuación se describen algunos sistemas que son utilizados para desarrollo de WSN.

Tiny-OS

Sistema operativo creado en la universidad de Berkeley especialmente para trabajar con dispositivos micaz, mica2, micadot, iris y telos que forman parte de la familia de motes que la institución ha desarrollado para construir redes de sensores inalámbricas. Es un sistema operativo de código abierto desarrollado en NesC (lenguaje derivado de C). Originalmente diseñado para Unix, puede trabajar en Linux de manera directa y en plataformas Windows sobre la máquina virtual cwing. La programación de los dispositivos se puede hacer a través

de los puertos serial, paralelo y ethernet. Ofrece una arquitectura basada en componentes, que permite la innovación y rápida puesta en práctica de aplicaciones para este tipo de sistemas, mientras reduce al mínimo el tamaño del código. Su funcionamiento interno está basado en eventos y tareas, los eventos se atienden de manera inmediata pudiendo interrumpir a las tareas en ejecución que generalmente son mayores en complejidad y ocupan más tiempo en su realización [16]. La biblioteca de componentes de TinyOS incluye protocolos de red, servicios distribuidos, manejadores de sensores, y herramientas de adquisición de datos, que pueden ser ocupadas directamente, o manipuladas para responder a necesidades más específicas. Aunque hay algunas tareas que son críticas en tiempo, como el manejo de la comunicación a través del radio o el obtener datos de algún sensor, TinyOS no se focaliza en dar garantías estrictas de Tiempo Real [19]. Por ejemplo, habrá eventos que no sea posible atender para no interrumpir una tarea lo que puede traer consigo pérdida de información. TinyOS está programado sobre un lenguaje estático, por lo cual no existen asignaciones de memoria de manera dinámica y los grafos de llamados quedan totalmente definidos en el tiempo de compilación [14]. Usa un modelo de componentes para diseñar aplicaciones para WSN, se puede hacer revisión sintáctica y semántica de la configuración de los componentes en tiempo de compilación, lo que facilita el análisis de programas y la detección de errores. En TinyOS los componentes implementan servicios comunes pero cada dispositivo ejecuta una imagen del sistema estáticamente vinculada, causando con esto la dificultad para ejecutar múltiples aplicaciones o actualizarlas de manera dinámica.

TinyOS no implementa la subcapa MAC del estándar IEEE 802.15.4., en lugar de eso provee su propio protocolo MAC llamado B-MAC, el cual envía paquetes 802.15.4 adaptados.

Contiki

Es un sistema operativo de código abierto escrito en lenguaje C y diseñado para dispositivos de 8 bits que utiliza 2 kilobytes de RAM y 40 kilobytes de ROM en su configuración predeterminada. Es altamente portable, multitareas y usa eficientemente la memoria en sistemas embebidos. Este sistema operativo se ejecuta en una gran variedad de plataformas en el rango de los micro controladores embebidos, tales como el MSP430 y el AVR para computadoras

personales antiguas, otro hardware soportado es SH3 de Hitachi y Zilog Z80.

Dentro de las cualidades que aparecen en su documentación y que le dan gran proyección, es que provee comunicación para Ipv4 e Ipv6 mediante dos pilas de comunicación, uIP y Rime. uIP es una simplificación del acuerdo RFC de la pila TCP/IP, la cual hace posible que Contiki logre comunicación sobre internet con los protocolos TCP, UDP, IP y ARP incluso en micro controladores pequeños de 8 bits. Rime es una pila ligera de comunicación que soporta comunicación de un solo salto y multisalto, diseñada para radios de baja potencia y provee un amplio rango de primitivas de comunicación. Los protocolos o las aplicaciones que se ejecutan sobre la pila Rime pueden implementar protocolos adicionales que no estén en la pila mediante el desarrollo de funciones sobre las primitivas de comunicación de Rime. Contiki usa reasignación de código en tiempo de ejecución para lograr la actualización de módulos de manera dinámica en los dispositivos, de tal manera que tiene la habilidad de cargar y descargar aplicaciones o servicios individuales en tiempo de ejecución. [10].

Mantis

Como describe [5] Mantis es un sistema operativo multihilos de código abierto desarrollado en lenguaje C y diseñado especialmente para trabajar con redes de sensores inalámbricas. El núcleo del sistema operativo ocupa menos de 500 bytes de memoria incluyendo las pilas de red, mientras que la pila para la administración de los hilos es de 128 bytes. Hace eficiente manejo de la energía de la red al poner inactivos los nodos sensores cuando ningún hilo se encuentra realizando una tarea. Provee un módulo de conexión remota para monitorización de la red, estableciendo parámetros de autenticación de usuarios. A las características antes mencionadas [1] agrega que Mantis permite la reprogramación de todo su código o parte de él mediante la descarga de imágenes del programa a la memoria flash a través del EEPROM. Una precaución que debe tomarse al programar sobre Mantis, es que debido la semántica multihilo que maneja, cada programa o módulo debe tener un espacio en la pila de memoria y deben ser implementados mecanismos de bloqueo para lograr la exclusión mutua de variables compartidas.

SOS

En [14], SOS se define como un sistema operativo para redes de sensores que consiste en una carga dinámica de módulos y un núcleo de sistema común. SOS implementa un sistema de mensajes, opera con memoria dinámica y realiza carga y descarga de módulos en tiempo de ejecución, entre otros servicios. Los módulos son programados cooperativamente y no hay protección de memoria. Sin embargo el sistema se protege de errores, usando técnicas tales como supervisión de puntos de escritura, vigilancia de temporizadores y rutinas primitivas de recolección de basura. Los módulos pueden ser añadidos y removidos con una interrupción mínima del sistema. SOS no está limitado a usar el mismo conjunto de módulos o la misma base del núcleo del sistema en todos los nodos dentro de la red y se puede beneficiar de diferentes tipos de protocolos incluyendo aquellos que dan soporte a la transferencia de datos punto a punto o punto a región. Muchos de los desarrollos se hacen en la capa de módulos, incluyendo los desarrollos de controladores, protocolos y componentes de aplicación. La modificación del núcleo será requerida solamente cuando el hardware o las capacidades de administración de recursos tengan que cambiar. La posibilidad de añadir, modificar o eliminar módulos de un dispositivo que está ejecutando el sistema introduce un número potencial de modos de error que no sucede en sistemas estáticos por lo que SOS provee algunos mecanismos para minimizar el impacto potencial de las fallas que pueden resultar. Parte de estos mecanismos tiene que ver con establecer una variable global que mantenga información de fallos cuando un error ocurre, permitiendo a los módulos manejar el error como convenga. La plataforma de SOS es altamente portable soportando los motes mica2, MicaZ y el nodo XYZ. Usa C estándar como lenguaje de programación para el núcleo y para los módulos, lo cual reduce la curva de aprendizaje del programador, de esta forma también saca provecho de muchos compiladores, ambientes de desarrollo, depuradores y otras herramientas diseñadas para el lenguaje C. En resumen, mediante la combinación del núcleo que provee los servicios básicos con el acoplamiento de módulos dinámicos que interactúan con el sistema en forma de aplicaciones, SOS provee una plataforma de propósito general para redes de sensores.

2.3.2. Sistemas Middleware

Aunque el objetivo de un middleware se puede describir como un sistema que implementa un alto nivel de abstracción de las redes de sensores hacia las aplicaciones, pueden clasificarse en tres diferentes categorías dependiendo de su enfoque según [18, 19]:

1. Middlewares que coordinan la asignación de tareas basados en parámetros de confiabilidad, tiempos de entrega oportunos, prioridad de los datos y prioridad de usuarios destino, lo que puede calificarse como sistemas que basan su funcionamiento en términos de QoS.

2. Midlewares que abstraen la red de sensores inalámbricos en una base de datos que soporta procesamiento de consultas.

3. Middlewares de procesamiento de controles basados en el contexto actual del entorno, el cual depende de la ubicación, las personas cercanas, los servidores utilizados, los dispositivos y los cambios hechos a través del tiempo.

A continuación se describen herramientas middleware utilizadas en WSN:

MiLAN

En el estudio que hace [15], podemos encontrar que MiLAN es un middleware diseñado para redes de sensores inalámbricas que operan con el estándar bluetooth y 802.11. Los servicios de este middleware abarcan desde la capa física hasta las capas de transporte y sesión pasando por las intermedias. Es un middleware configurable mediante parámetros para que pueda prestar QoS. Se ha implementado en monitorizaciones médicas, supervisión del medio ambiente y seguridad de oficinas y hogares. MiLAN permite a las aplicaciones de redes de sensores especificar la calidad que necesita y ajusta las características de la red para incrementar el tiempo de vida de la misma siempre y cuando cumpla los requerimientos de calidad especificados.

Específicamente MiLAN recibe información de:

1. Especificaciones de requerimientos de calidad de servicio de las aplicaciones sobre la marcha e información de cómo puede cumplir con estas especificaciones usando diferentes combinaciones de sensores.

2. El contexto del sistema completo acerca de la importancia relativa de las diferentes aplicaciones.

3. La red, en cuanto a sensores disponibles y los recursos con que estos cuentan como capacidad de energía, ancho de banda del canal, etc.

Combinando esta información, MiLAN continuamente adapta la configuración de la red (por ejemplo especificando los sensores que deben enviar datos, los que deben ser enrutadores en redes multi saltos, los sensores que deben tener el rol de cabeceras de celda, etc.) para cumplir las necesidades de las aplicaciones mientras maximiza el tiempo de vida de la red.

Agilla

Es un middleware que permite a los usuarios desarrollar agentes móviles en una red para desempeñar tareas específicas. En Agilla cada nodo sensor puede ser monitorizado por múltiples agentes. Agilla provee un esquema que permite a los agentes moverse por ellos mismos a lugares deseados de acuerdo con las condiciones cambiantes de la red [11]. Cada nodo sensor mantiene un espacio de memoria que contiene un conjunto de descriptores predefinidos acerca del nodo. Este espacio de memoria es compartido por agentes locales, los cuales pueden registrar lo que les sea de interés en eventos particulares mediante la inserción de un modelo de tupla dentro del espacio de memoria. Cuando un nodo detecta eventos de coincidencia, este actualiza su tupla en consecuencia y reporta los eventos al agente. Por esta razón los agentes no necesitan continuamente recabar datos del estado de la red [28].

ActorNet

Cómo puede verse en [20] ActorNet es una plataforma para agentes que provee servicios tales como memoria virtual, cambio de contexto y proceso multitarea para dar soporte a un lenguaje altamente expresivo para WSN. ActorNet soporta coordinación entre agentes en un modelo de comunicación asíncrona, típicamente empleado en WSN debido a la brecha de comunicación y la velocidad de cómputo. Esto sin embargo le presenta un problema serio a resolver si consideramos que un simple mensaje de multidifusión en una WSN de gran tamaño requiere un gran número

de mensajes para realizarse de manera completa, de tal manera que cuando se tenga una comunicación entre dos agentes, se requerirá hacer repetidas multidifusiones, lo cual resulta inviable, dados los recursos limitados de las WSN.

2.3.3. Plataformas Freescale para desarollo de WSN

Freescale ofrece tres plataformas de desarrollo de software para redes de sensores inalámbricas como lo muestra la figura 2.1.

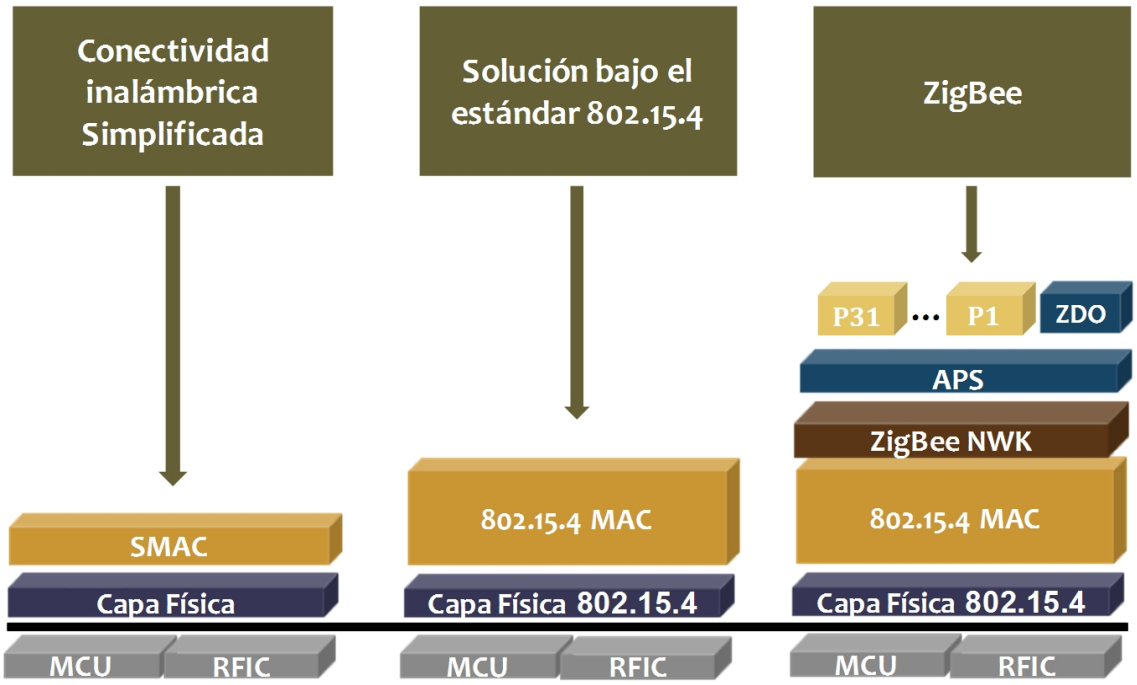


Figura 2.1: Propuesta de Freescale para desarrollo de WSN

SMAC

SMAC (Freescale's Simple MAC) provee una solución simplificada y de bajo costo para establecer una red de sensores inalámbrica. SMAC se basa en la definición de la capa física del estándar IEEE 802.15.4, y provee los comandos necesarios para crear redes tipo estrella y punto a punto. El tamaño pequeño del código que ocupa solamente de 2 a 4 Kbytes de memoria, permite el uso de un procesador de bajo costo para su ejecución y esto combinado con un

tranceptor de radiofrecuencia de semejantes características, hacen que sea una plataforma ideal para el desarrollo de aplicaciones de dimensiones pequeñas que intentan eliminar los cables.

SMAC provee funciones de control e inicialización del transceptor, administración de la energía, funciones de prueba de RF, y operaciones básicas como enviar y recibir paquetes de datos aplicando la norma del ciclo de Valoración de Uso del Canal (CCA, siglas en inglés) y Detección de Energía (ED, siglas en inglés). Estas funciones proveen la construcción básica de módulos y ayudan al usuario a poner en marcha su desarrollo de manera rápida. A pesar de la simplicidad de la plataforma SMAC, posee algunas características que la hacen potente:

- La capacidad de hacer actualización de programas a través del canal inalámbrico.
- La generación de módulos repetidores para ampliar el rango de cobertura.
- La opción de incluir en los desarrollos un módulo de seguridad.

MAC 802.15.4

La pila MAC de Freescale implementa de forma completa la capa física y la capa de control de acceso al medio del estándar IEEE 802.15.4 en un rango de 24 a 27 Kbytes. Dentro de los servicios y características más importantes que el estándar marca y que se pueden ver como ventajas en la pila MAC son:

- Capacidad de construir aplicaciones para redes formadas en topología punto a punto o estrella.
- Administración del canal mediante servicios CSMA/CA.
- Opción de utilizar el mecanismo GTS.
- Utilización de mensajes de reconocimiento para transmisión de datos confiable.
- Opción de implementar seguridad AES-128.
- Posibilidad de extenderse con aplicaciones compatibles con el estándar.

La pila MAC tiene mejores servicios y provee más funcionalidad que lo que ofrece una aplicación de SMAC. El desarrollador no necesita programar servicios como administración de direcciones, mensajes de reconocimiento y administración del acceso al medio para mejorar el rendimiento de la aplicación. Además desde un punto de vista de soporte y crecimiento, existen más desarrolladores utilizando un estándar que una aplicación de propietario, de tal manera que la facilidad de mantenimiento aumentará rápidamente con el tiempo.

La pila MAC se puede ver en desventaja al compararla con ZigBee si se piensa en aplicaciones que requieren la elaboración de un protocolo de red que puede llegar a ser grande y complejo dependiendo de la cantidad y distribución de los nodos de la red. Sin embargo, es una excelente opción cuando la aplicación que se va a realizar no requiere enrutamiento y en cambio se desea tener un protocolo de transporte confiable, ya que se puede implementar transmisiones síncronas mediante redes con señalizaciones de inicio y fin, implementando GTS.

ZigBee

La arquitectura de la pila de ZigBee está compuesta de un conjunto de bloques organizados en capas. Cada capa desempeña un conjunto de servicios para la capa superior, incluyendo una entidad de datos que provee un servicio de transmisión de datos y una entidad de administración que provee todos los demás servicios. Cada entidad de servicio expone una interface a la capa superior a través de un Punto de Acceso al Servicio (SAP, siglas en inglés), y cada SAP ofrece un número determinado de primitivas para lograr la funcionalidad requerida. La arquitectura de ZigBee está basada en el modelo de siete capas del estándar de Interconexión de Sistemas Abiertos (OSI, siglas en inglés) creado por la Organización Internacional para la Estandarización (ISO, siglas en inglés) pero define sólo aquellas capas relevantes para lograr la funcionalidad en el contexto de las aplicaciones de las redes de sensores. El estándar IEEE 802.15.4 define las especificaciones de la capa física y la capa de control de acceso al medio para dar soporte a dispositivos que consumen poca cantidad de energía y típicamente operan en una red de Operación de Espacio Personal (POS, siglas en inglés). La Alianza ZigBee construye sobre esta base para proveer la capa de red (NWK) y la plataforma para la capa de aplicación, la cual incluye la Subcapa de Soporte de Aplicación (APS), el objeto de dispositivos ZigBee (ZDO) y

los objetos de aplicación de soporte a ZigBee definidos por el fabricante de los dispositivos.

Las responsabilidades de la capa de red de ZigBee incluye mecanismos usados para unirse y separarse de la red, para aplicar seguridad a las tramas y para enrutar tramas a los destinatarios. El descubrimiento y mantenimiento de las rutas entre los dispositivos son resueltos en la capa de red. También el descubrimiento de vecinos a un salto y el almacenamiento de la información pertinente a los vecinos son hechos en la capa de red. Además la capa de red de un dispositivo coordinador ZigBee es responsable de iniciar una nueva red cuando es apropiado y asignar direcciones a los nuevos dispositivos que se asocian.

La capa de aplicación de ZigBee consta de la subcapa APS, el componente ZDO y los objetos de aplicación definidos por el fabricante del dispositivo. La subcapa APS es responsable de mantener las tablas de enlace, la cual es responsable de mantener a dos dispositivos unidos basada en sus servicios y necesidades, y enviar mensajes entre ambos dispositivos. Las responsabilidades de ZDO incluyen la definición del rol de un dispositivo dentro de la red, iniciando y respondiendo solicitudes y estableciendo una relación segura entre los dispositivos de la red. Otra responsabilidad de ZDO es descubrir que dispositivos están operando en la POS y que dispositivos están asociados. Los objetos de aplicación de los fabricantes, añaden perfiles definidos dentro de la Alianza ZigBee. Estos implementan las aplicaciones de acuerdo a las descripciones definidas por la Alianza ZigBee. Los perfiles de los dispositivos son una serie de mensajes que permiten a estos realizar las funciones que dan las capacidades principales de descubrimiento, vinculación y gestión de la red ZigBee.

2.3.4. ¿Por qué MAC 802.15.4 implementado por Freescale?

Dentro de las opciones antes mencionadas destaca entre los sistemas operativos Tiny-OS por su popularidad en el desarrollo de sistemas embebidos, no obstante se puede considerar como desventaja para este sistema la curva de aprendizaje de los desarrolladores debido a que los programas deben escribirse en Nes-C, un lenguaje derivado de C pero que se debe entender en un paradigma diferente al de programación estructurada y que a su vez no reúne todas las características para caer en el paradigma de la programación orientada a objetos. Otra

consideración importante es que el sistema operativo no utiliza las convenciones del estándar IEEE 802.15.4 para operar en la capa de control de acceso al medio, sino que utiliza una capa llamada B-MAC, la cual debe estudiarse además del estándar para hacer un desarrollo adecuado. Contiki es una opción bastante atractiva como sistema operativo al ofrecer una pila que provee funcionalidad TCP/IP, lo cual elimina la brecha entre las redes de sensores inalámbricas y las redes convencionales con tecnología TCP/IP, sin embargo esto no sucede de manera automática y transparente, si no que se hace una reducción de los protocolos TCP/IP y del tamaño de los paquetes para que puedan ser manejados a pesar de los limitados recursos de procesamiento y memoria que poseen los dispositivos para formar WSN. Contiki al no haber sido creado originalmente para WSN tampoco utiliza las convenciones del estándar IEEE 802.15.4 para operar. MANTIS y SOS son dos sistemas operativos que tienen aceptación entre un número considerable de desarrolladores debido principalmente al manejo de memoria dinámica y en el caso del primero también por la poca capacidad de recursos que requiere para operar.

Por otra parte, dentro los sistemas middleware los que tienen mejor funcionalidad son los que trabajan con agentes y de manera distribuida, sin embargo los algoritmos para lograr esta funcionalidad son demasiado complejos y por tanto difíciles de modificar si se desea emplearlos en alguna aplicación hecha a la medida. Los que sobresalen por haber sido implementados para más de un desarrollo son MiLAN y Agilla. Dentro de los sistemas middleware revisados se pudo observar que el interés de éstos es, obtener conclusiones a partir de datos que provienen de grupos de nodos que observan ciertas zonas o cierto comportamiento de promedios de datos recabados de diversos dispositivos, lo que presenta inconvenientes al tratar de adaptarlos a un sistema que por su naturaleza debe identificar cada dato con el dispositivo del que proviene.

Las herramientas que ofrece Freescale permiten desarrollar aplicaciones hechas a la medida a partir de una plataforma que implementa el estándar IEEE 802.15.4 de manera parcial, como SMAC que hace la implementación únicamente de la capa física, o total, como MAC que implementa las dos capas definidas por el estándar. La otra opción es ZigBee que implementa la totalidad de la capa física y gran parte de la capa MAC, dejando de lado la opción de GTS y las redes señalizadas, pero agregando la funcionalidad de una capa de red para aplicaciones multisalto e incluyendo el desarrollo de perfiles para hacer los desarrollos compatibles con

diversos fabricantes, entre otras herramientas que ayudan para que los desarrollos sean veloces y con alto grado de confiabilidad y soporte.

Debido a que la red donde el sistema se ejecutará es una infraestructura e-salud se pretende elegir las herramientas que proveen las condiciones de mayor confianza para la implementación. Por esta razón se ha elegido utilizar una de las herramientas de Freescale ya que éstas se basan en el estándar IEEE 802.15.4, con lo que se garantiza soporte para el desarrollo y mantenimiento independiente del desarrollador, así como fácil comprensión de lo propuesto para quienes están familiarizados con el estándar. Otra ventaja al utilizar las herramientas de Freescale, es que las plantillas a partir de las cuales se inicia el desarrollo, están construidas en lenguaje C, por lo que la curva de aprendizaje se reduce considerablemente. Dentro de las tres pilas que ofrece Freescale se seleccionó MAC 802.15.4 debido a que implementa el estándar de manera completa. Además, las características de la red no contemplan comunicación multisalto, de tal manera que la elección de ZigBee habría significado la utilización de mayor cantidad de recursos de los dispositivos de manera innecesaria, así como el pago de una licencia de mayor costo por características que en la práctica no se estarán utilizando para esta infraestructura.

La siguiente sección describe los componentes, los tipos de red, las características y las formas de operación básicas del estándar IEEE 802.15.4, lo cual cubre la forma en que el sistema se comporta en sus capas inferiores, dejando al capítulo 3 la descripción de las capas superiores, siendo éstas el desarrollo central de este trabajo de tesis.

2.4. WSN basadas en el estándar IEEE 802.15.4

2.4.1. Componentes

En las redes de sensores inalámbricas bajo el estándar 802.15.4 intervienen dos tipos de componentes: Los dispositivos con funcionalidad completa (FFD, siglas en inglés) y los dispositivos con funcionalidad limitada (RFD, siglas en inglés).

FFD: Son dispositivos que pueden comunicarse con dispositivos RFD o con otros dispositivos FFD, con frecuencia operan transmitiendo grandes cantidades de datos y generalmente

están conectados a una fuente de energía constante. Dentro de la red pueden desempeñar los siguientes roles:

1. Como coordinador de la Red de Área Personal (PAN, siglas en inglés), identificando, configurando y controlando la red.
2. Como coordinador de celda, provee servicios de sincronización mediante la transmisión de mensajes de señalización de inicio y fin. En este rol debe estar asociado a un coordinador de la red y no crea su propia red.
3. Como dispositivo final, debe asociarse a un coordinador y no puede asociar otros dispositivos a sí mismo.

RFD: Únicamente pueden comunicarse con un FFD al que estarán asociados, son dispositivos que normalmente no envían grandes cantidades de información, se alimentan de baterías y usan al mínimo su capacidad de procesamiento y memoria. Toman siempre el rol de dispositivos finales y son utilizados para realizar tareas simples.

2.4.2. Topologías

El estándar define dos topologías en las que puede operar una red de sensores inalámbricas, la topología entre pares o p2p y la topología estrella, como se muestra en la figura 2.2. El uso de una u otra dependerá de los requerimientos de la aplicación que se desea implementar.

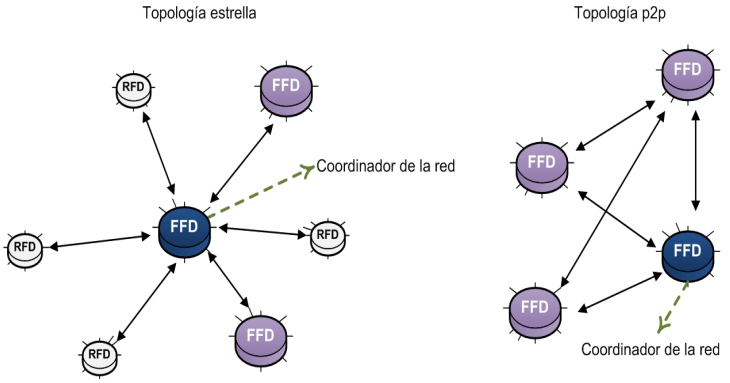


Figura 2.2: Topologías de WSN bajo el estándar IEEE 802.15.4

Topología estrella: Es una topología centralizada en la que existe un dispositivo central de tipo FFD que será el coordinador de la red, el cual una vez que ha sido activado, establece un identificador para su red y permite la asociación a la misma de otros dispositivos FFD y RFD. Los dispositivos FFD que existen en la red además del coordinador, normalmente cumplen funciones de enrutamiento.

Dentro de las aplicaciones que normalmente utilizan este tipo de topología está la automatización de hogares, sistemas de entretenimiento y aplicaciones para el cuidado de la salud.

Topología P2P: Es una topología de red descentralizada y difiere de la topología estrella en que cada dispositivo puede comunicarse con otro dispositivo en su radio de cobertura. El primer dispositivo que establezca un canal de comunicación será denominado coordinador de la red, y a partir de esto la red irá creciendo con cada dispositivo que se una a ella sin mantener una estructura determinada. Una red P2P puede ser ad hoc, auto-organizable y auto-reconfigurable. Las redes p2p necesitan a menudo servicios de enrutamiento de mensajes para comunicar un dispositivo con otro que se encuentra a varios saltos, sin embargo esas tareas son de la capa de red y están fuera del alcance del estándar.

Entre las aplicaciones que utilizan este tipo de topología están las de control y monitorización industrial, seguimiento de inventarios, agricultura inteligente, y seguridad.

Topología Híbrida: La figura 2.3 muestra la tercera estructura de red que resulta de la combinación de las dos topologías anteriores. En esta topología se tiene un nodo coordinador de toda la red y nodos coordinadores de celdas que forman pequeñas estrellas dando como resultado una estructura de árbol en donde los dispositivos FFD tienen capacidad de ser ramas del árbol y asociar otros dispositivos FFD que a su vez puedan hacer lo mismo. Los dispositivos RFD terminan constituyéndose en las hojas del árbol al estar en el último nivel de la estructura.

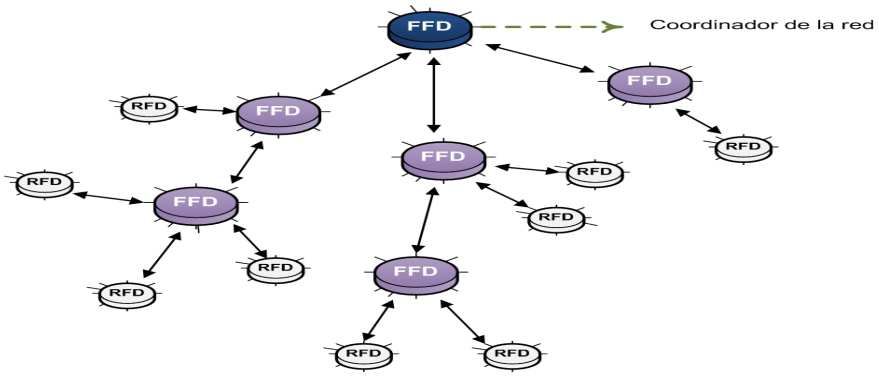


Figura 2.3: Combinación de topologías estrella y p2p

2.4.3. Arquitectura

El estándar IEEE 802.15.4 define la capa física y la capa de acceso al medio dejando a los desarrolladores de aplicaciones la responsabilidad de construir las siguientes capas como lo muestra la figura 2.4. En muchas ocasiones se implementa la capa de red y transporte como una aplicación denominada middleware y la capa superior como su propio nombre lo indica, una aplicación que cumple fines definidos por los requerimientos del sistema al cual prestará servicios.

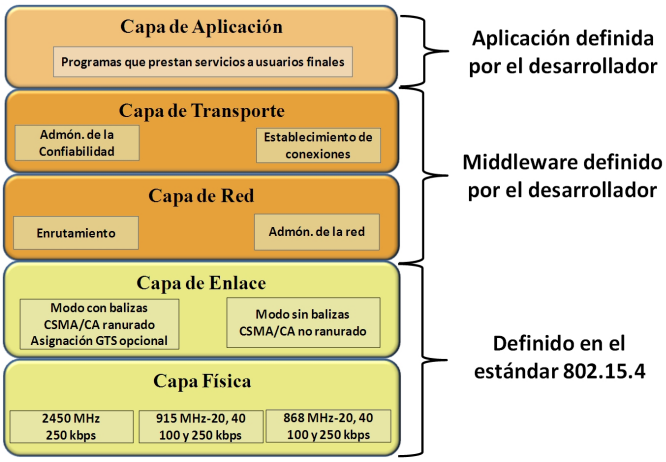


Figura 2.4: Arquitectura de sistema basado en IEEE 802.15.4 rev. 2006

2.4.4. Capa Física

No es de interés para este trabajo hacer una descripción exhaustiva de la capa física, por lo que se tomó únicamente la introducción de la descripción que [23] hace acerca de ésta, con lo que se cubre los conceptos básicos necesarios para interactuar con ella y sacar el mayor provecho de los servicios que provee.

Bandas de frecuencia en las que opera

El principal componente de la capa física es un radio transceptor que opera en una o más bandas representadas en la figura 2.5, donde podemos apreciar la frecuencia de 868 Mhz libre para ser utilizada en Europa, 915 Mhz libre para ser utilizada en Norte América y la de 2450 Mhz que puede utilizarse libremente en cualquier parte del mundo.

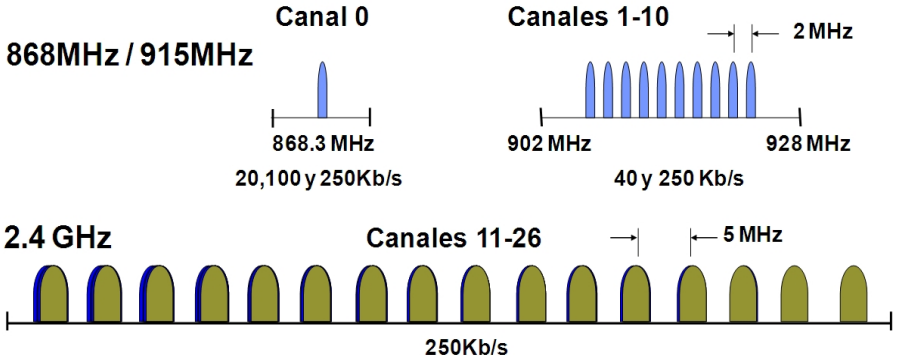


Figura 2.5: Bandas de frecuencia para IEEE 802.15.4 [17]

Tareas que realiza

1. Activación y desactivación del radio transceptor: Apaga y enciende el transceptor como respuesta a las capas superiores.

2. Detección de la energía (ED): Hace la estimación de la energía con la que fue recibida la señal en un determinado canal.

3. Indica la calidad del enlace (LQI): Hace una caracterización de la potencia de la señal con la que fue recibido un paquete, lo cual puede ser implementado utilizando un detector de

energía, haciendo una estimación de la señal a ruido en el radio o una combinación de ambos.

4. Evaluación del canal con menor interferencia (CCA): Determina si el medio está ocupado o disponible para la aplicación del mecanismo CSMA/CA.

5. Selección de la frecuencia del canal: Ubica al transceptor en uno de los 27 canales disponibles a petición de las capas superiores.

6. Transmisión y recepción de datos: Envía y recibe a través del radio los datos que recibe de las capas superiores.

Servicios que ofrece

Provee la interacción entre la capa de control de acceso al medio MAC y el canal de radio físico mediante dos servicios accedidos a través de dos interfaces que se constituyen en los puntos de acceso como lo muestra la figura 2.6.

- **Servicio de datos:** Hace posible la transmisión y recepción del Protocolo de Unidad de Datos de la capa Física (PPDU, siglas en inglés) mediante el Punto de Acceso al Servicio de Datos de la capa Física (PD-SAP, siglas en inglés).

- **Servicio de administración:** Hace posible la interacción con la Entidad de Administración de la capa Física (PLME, siglas en inglés) mediante el Punto de Acceso al Servicio PLME (PLME-SAP, siglas en inglés).

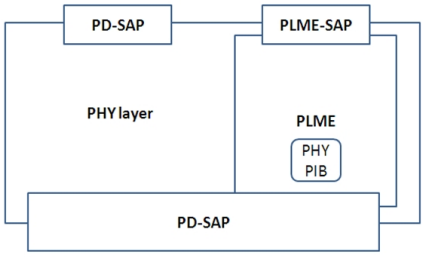


Figura 2.6: Modelo de referencia de la capa física [23]

Conceptualmente la capa física incluye la entidad llamada PLME, la cual provee las interfaces a través de las cuales serán invocadas las funciones que proveen los servicios de administración de la capa, además de ser responsable del mantenimiento de la Base de datos de Información de la PAN (PIB, siglas en inglés), siendo esta la base de datos que contiene los objetos con los atributos requeridos para la administración de la capa.

Atributos y constantes de la PIB

Los atributos que se presentan en la tabla 2.1 son todos los atributos que se requieren para administrar la capa física de un dispositivo que trabaja bajo el estándar IEEE 802.15.4.

Atributos	**Identificador**	**Tipo**	**Rango**	**Descripción**
phyCurrentChannel	0x00	Entero	0-26	Canal de radio frecuencia que será utilizado para transmitir.
•*phyChannelsSupported*	0x01	Arreglo	R x 32 bits y $1 \geq R \leq 32$	El arreglo está compuesto por R renglones cada uno de los cuales es una cadena de bits con las siguientes propiedades: Del $b_{27}, ..., b_{32}$ indica el canal y del $b_0, ..., b_{26}$ indica el estatus (1=disponible, 0=no disponible) por cada uno de los 27 canales válidos.
∗*phyTransmitPower*	0x02	Mapa de bits	0x00-0xbf	Los bits más significativos son de sólo lectura y representan la tolerancia sobre la potencia de la energía: 00 = $\pm 1dB$, 01 = $\pm 3dB$ y 10 = $\pm 6dB$. Los 6 bits menos significativos, representan un entero con signo en formato de complemento a dos correspondiente a la potencia nominal transmitida del dispositivo en decibeles relativa a 1 mW. El valor mas bajo de *phyTransmitPower* se interpreta como menor o igual a $-32dBm$.
phyCCAMode	0x03	Entero	0-3	Representa el modo en el que se dejará configurado el CCA.
phyCurrentPage	0x04	Entero	0-31	Representa la página del canal actual de la capa física Se utiliza en conjunción con *phyCurrentChannel* únicamente para identificar el canal que está siendo utilizado.
•*phyMaxFrameDuration*	0x05	Entero	55,212,266,1064	Es el máximo número de símbolos en una trama: = *phySHRDuration* + *ceiling*([*aMaxPHYPacketSize* + 1]*xphySymbolsPerOctet*).
•*phySHRDuration*	0x06	Entero	3,7,10,40	Representa la duración del encabezado de sincronización en símbolos para la capa física.
•*phySymbolsPerOctet*	0x07	Real	0.4,1.6,2.0,8	Representa el número de símbolos por octeto para la capa física.

Tabla 2.1: Atributos de la capa física.

La viñeta (•) identifica atributos de sólo lectura que se pueden acceder por la capa superior utilizando la primitiva PLME-GET.request. El asterisco (∗) indentifica atributos con bits de sólo lectura que pueden accederse mediante PLME-GET.request, y bits de lectura y escritura que pueden ser accedidos por las primitivas PLME-GET.request y PLME-SET.request respectivamente.

Las constantes presentadas en la tabla 2.2 definen las características de la capa física, son dependientes del hardware y no pueden ser cambiadas durante la operación.

Constante	Descripción	Valor
aMaxPHYPacketSize	Define el máximo tamaño (en octetos) del PSDU que la capa física es capaz de recibir	127
aTurnaroundTime	Es el máximo tiempo (en períodos de símbolos) de ida i vuelta para una transmisión RX-to-TX o Tx-to-RX	12

Tabla 2.2: Constantes de la capa física.

2.4.5. Capa de Control de Acceso al Medio

La capa MAC es la responsable de mantener la comunicación entre los nodos de la red que mantienen una comunicación directa. Provee dos servicios a través de dos interfaces que sirven como puntos de acceso:

- El servicio de datos de la MAC, el cual puede ser accedido a través del Punto de Acceso al Servicio de la Parte Común de la Subcapa MAC (MCPS-SAP, siglas en inglés), y
- El servicio de administración de la MAC que puede ser accedido a través del Punto de Acceso al Servicio de la Entidad de Administración de la capa MAC (MLME-SAP, siglas en inglés).

Estos dos servicios como se muestra en la figura 2.7 proveen la interface de comunicación entre el SSCS (Service-Specific Convergence Sublayer) y la capa física a través de las interfaces PD-SAP y PLME-SAP (ver sección 2.4.4). Además de estas interfaces existe también una interfaz implícita entre la Entidad de Administración de la capa MAC (MLME, siglas en inglés)

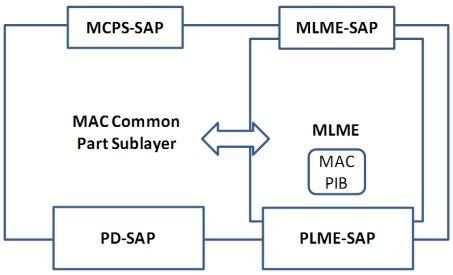


Figura 2.7: Modelo de referencia de la capa MAC [23]

y la Parte Común de la Subcapa MAC (MCPS, siglas en inglés) que permite a la MLME usar los servicios de datos de la MAC.

Conceptualmente la capa MAC incluye una entidad de administración llamada MLME. Esta entidad provee los servicios de interface a través de los cuales pueden ser invocadas las funciones de administración de la capa. La MLME es además responsable del mantenimiento de la PIB de la MAC, siendo esta la base de datos que contiene los objetos con los atributos requeridos para la administración de la capa.

Modos de operación

Existen dos modos de operación del acceso al medio, los cuales se distinguen por que uno usa paquetes de señalización (es decir paquetes que marcan el inicio del período de transmisión definido por un marco en el tiempo) y el otro modo trabaja sin ellos.

- Sin mensajes de señalización (*no-beaconed networks*): Los nodos contienden siempre por el canal utilizando CSMA/CA no ranurado, lo que indica que transmiten inmediatamente después de detectar el canal libre. Es un modelo de operación más fácil de implementar pero menos eficiente.

- Con mensajes de señalización (*beaconed networks*): En este modo de operación, el coordinador transmite paquetes señalizadores (llamados *'beacons'* en el argot computacional) de manera periódica para sincronizar a los nodos y para mantener la red activa mediante la identificación de la PAN en el rango que la señal abarca. El período de tiempo entre dos paquetes de señalización es denominado *'SuperFrame'* (SF, siglas en inglés) y es dentro de

este período que está designado el tiempo para recibir y transmitir tramas desde y hacia los dispositivos. Este espacio de tiempo para intercambio de información puede contener hasta tres divisiones, una donde los nodos acceden al medio por contienda (CAP, siglas en inglés) administrado por el mecanismo CSMA/CA ranurado, otra donde los dispositivos acceden al medio libres de contienda (CFP, siglas en inglés) administrado por el mecanismo GTS (ver 2.4.6) y una tercera parte donde los nodos permanecen inactivos llamado Período Inactivo (PI, siglas en inglés que aparecen en orden inverso para no confundirlo con IP) y por tanto no se realiza ninguna transmisión.

Formas que puede tomar el SF dependiendo de las conveniencias de la aplicación:

1. Utilizar todo el SF como período de acceso por contención, aplicando el mecanismo CSMA/CA ranurado como se muestra en la figura 2.8.

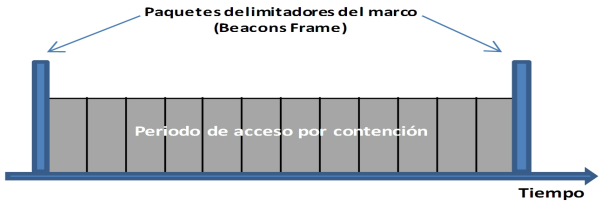


Figura 2.8: SF utilizando únicamente CAP

2. Se configura un CAP y un PI como se ve en la figura 2.9. Durante el PI los nodos no transmiten ni reciben información, lo cual contribuye al ahorro de energía.

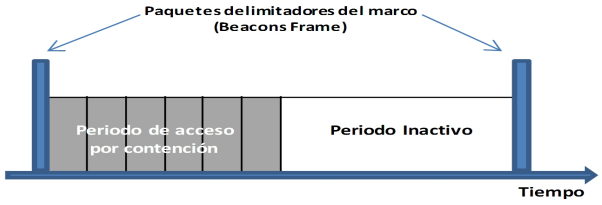


Figura 2.9: SF utilizando CAP y PI

3. La tercera forma de configurar el SF mostrado por la figura 2.10, presenta un CAP y CFP controlado por GTS (ver sección 2.4.6). Los nodos pueden solicitar ranuras libres de contienda para transmitir de manera síncrona. El coordinador es responsable

de aceptar o denegar la solicitud del servicio. Este escenario se utiliza cuando la información de ciertos nodos es de mayor importancia para el sistema, por lo que deben tener prioridad al transmitir.

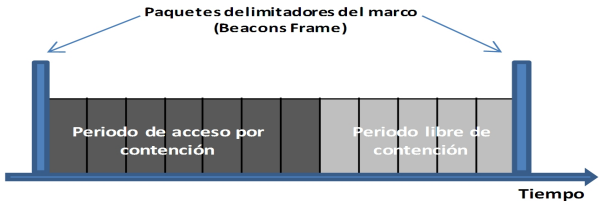


Figura 2.10: SF utilizando CAP y GTS

4. La última forma de configurar el SF se logra agregando a la última configuración un PI, como muestra la figura 2.11.

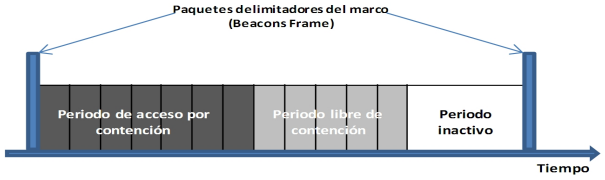


Figura 2.11: SF utilizando CAP, GTS y PI

2.4.6. Mecanismos de la capa MAC

Mecanismo CSMA/CA ranurado

Es el mecanismo utilizado por la capa MAC para acceder al medio. Este mecanismo trata de evitar colisiones durante la transmisión de datos y su funcionamiento se basa en tres estrategias principalmente:

1. IFS (*Interframe Space*): Las colisiones son evitadas mediante el aplazamiento de las transmisiones incluso cuando el canal se encuentra libre. Cuando una estación encuentra el canal libre, no transmite inmediatamente sino que espera por un período de tiempo llamado IFS. Esto se debe a que aunque el canal parezca estar libre es posible que otra estación distante ya esté transmitiendo y la señal no se haya alcanzado a detectar. Si después del

tiempo IFS el canal aun está libre la estación puede transmitir, pero aun necesita esperar un período de contención.

2. Contention Window: Es un período de tiempo dividido en partes llamadas ranuras. Una estación que está lista para transmitir elige aleatoriamente un número de ranura que será su tiempo de espera para transmitir. El número de ranuras en la ventana cambia de acuerdo a la estrategia exponencial binaria de tiempos de espera. Esto significa que después de que es puesto la primera vez para transmitir en una ranura este número se duplica cada vez que la estación no puede detectar el canal libre después del período IFS. Si la estación encuentra el canal ocupado, no reinicia el proceso, sino que detiene el tiempo y lo reinicia cuando el canal es detectado como libre, lo cual da prioridad a las estaciones con los tiempos más largos de espera.

3. Aun con todas estas precauciones, puede haber colisiones y traer la consecuente pérdida de datos, además de que los datos pueden ser corrompidos durante la transmisión. Por lo tanto los ACK y el tiempo de espera agotado pueden ayudar a garantizar que el receptor ha recibido la información.

Los pasos que se dan para aplicar este mecanismo en redes inalámbricas son los siguientes:

1. Antes de enviar una trama, el dispositivo inspecciona el medio para detectar el nivel de energía del canal.

    - El canal usa una estrategia de persistencia mediante tiempos de espera hasta que el canal esté libre.
    - Después de que se encuentra el canal libre, el dispositivo espera un período de tiempo llamado Espacio Distribuido entre Tramas (DIFS, siglas en inglés) y después envía la trama de control que solicita permiso para transmitir (RTS, siglas en inglés).

2. Después de recibir el RTS y esperar un período de tiempo llamado Espacio Corto Entre Tramas (SIFS, siglas en inglés), el dispositivo destino envía una trama de control llamada Libre para Enviar (CTS, siglas en inglés) al dispositivo fuente. Esta trama de control indica que el dispositivo está listo para recibir los datos.

3. El dispositivo fuente envía los datos después de esperar un período SIFS.

4. El dispositivo destino, después de esperar un período de tiempo SIFS, envía un ACK para confirmar que la trama ha sido recibida. Los ACK son necesarios en este protocolo debido a que el dispositivo fuente no tiene ninguna otra manera de saber si la trama que envió llegó a su destino.

Operación de CSMA/CA en IEEE 802.15.4

La figura 2.12 muestra cómo opera el mecanismo CSMA/CA en el estándar 802.15.4 y se puede resumir de la siguiente manera:

1. Inicialización de variables: $NB = 0; CW = 2$; y $BE = Min(2 - macMinBE)$.

   - **NB:** Número de intentos de acceso al canal que un dispositivo lleva acumulados. Cada vez que un dispositivo va a enviar un nuevo paquete, esta variable debe ser inicializada a 0.
   - **CW:** Es la longitud de la ventana de contienda. Representa el número de períodos de espera que el canal debe estar sin actividad para que se considere libre.
   - **BE:** Se emplea para determinar, aleatoriamente, el número de periodos de espera que el dispositivo debe esperar para intentar acceder al canal después de un acceso fallido.

2. Después de asignar el tiempo de espera, el algoritmo hace una pausa equivalente a un número aleatorio de períodos de tiempo antes de intentar acceder al medio.

3. Evalúa el canal para verificar si está libre o no (CCA).

4. Si CCA identifica el canal ocupado, NB se incrementa en 1 y el algoritmo regresa al paso dos.

5. Si CCA identifica el canal libre, CW se decrementa en 1 y cuando llega a ser 0, la trama es transmitida, de otra manera el algoritmo salta al paso tres.

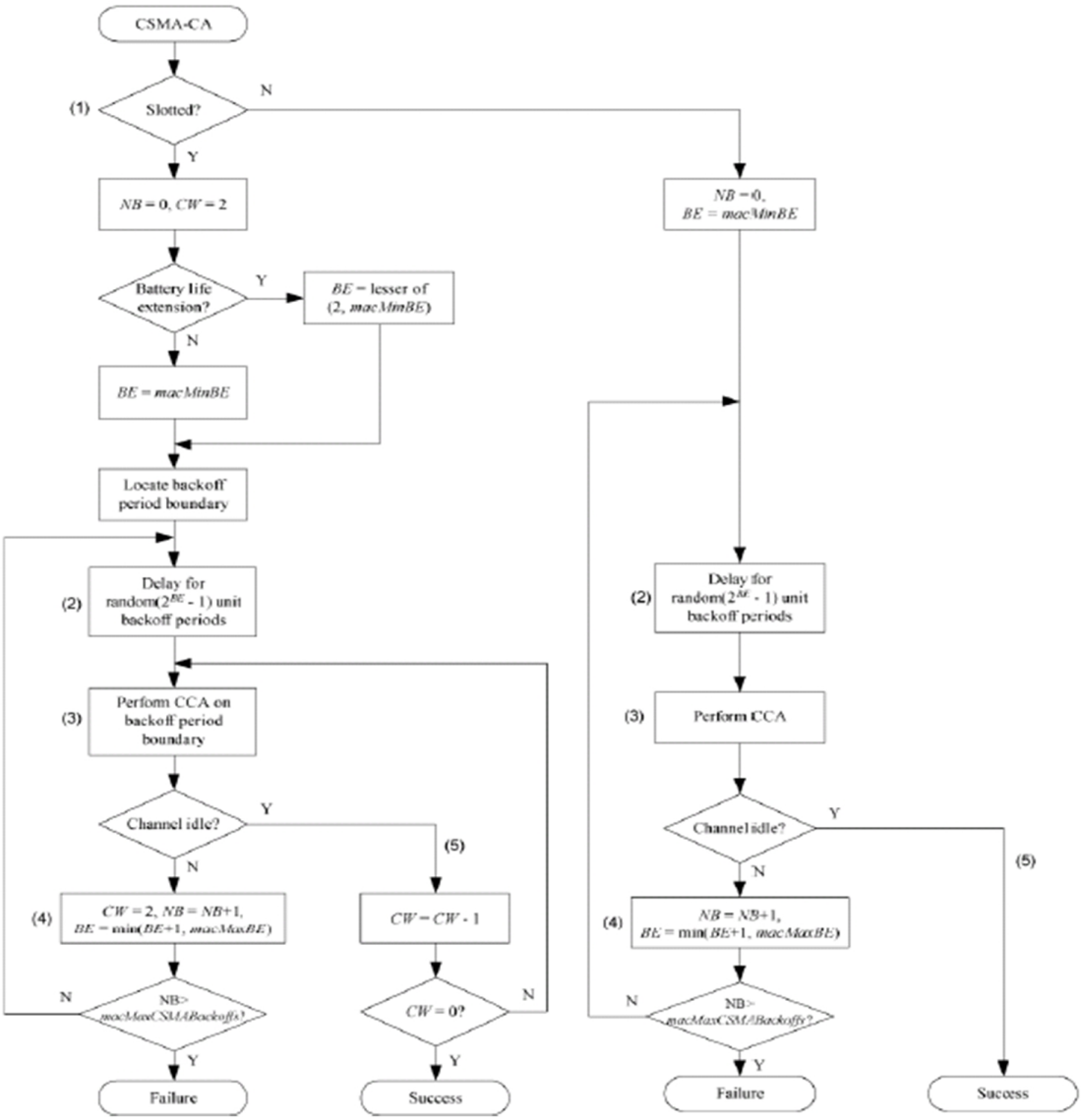


Figura 2.12: Algoritmo CSMA/CA en IEEE 802.15.4 [23]

Los pasos anteriores describen el algoritmo para un CSMA/CA ranurado que es la parte izquierda del diagrama de flujo de la figura 2.12. La parte de la derecha muestra el flujo para la aplicación del mecanismo pero sin ranuras en el que se elimina la verificación de la capacidad de la batería y los períodos de tiempo de espera antes de detectar si el canal está vacío. Por otra parte si el canal es detectado como libre en un mecanismo de tipo CSMA/CA no ranurado, la sub-capa MAC debe iniciar la transmisión de la trama inmediatamente. La aplicación de este mecanismo es menos eficiente, debido a que la probabilidad que las tramas tienen de coincidir en algún momento de la transmisión es más alta, ya que los dispositivos pueden transmitir en

cualquier momento que detecten el canal libre y las tramas que se transmitieron un instante antes pueden ser interceptadas antes de que lleguen a su destino final.

En CSMA/CA ranurado, cuando la extensión de la vida de la batería es 0, el algoritmo se asegura que después del paso 2, las operaciones restantes puedan ser realizadas y las tramas transmitidas antes de que se termine el período de acceso por contienda (CAP). Si el número de períodos de espera es mayor que el tiempo restante del CAP, la subcapa MAC hace una pausa a la cuenta regresiva de períodos de espera al final del CAP y lo aplaza hasta el inicio del siguiente período de transmisión. Si el número de períodos de retardo es menor o igual que el número de períodos restantes de retardos en el CAP, la sub-capa MAC aplica un período de retardo y reevalúa si puede proceder con la transmisión. Si la sub-capa MAC no tiene suficiente tiempo, todo se aplaza hasta el inicio del siguiente período de transmisión y continúa con el paso 3. Si la extensión de la batería es 1, la cuenta regresiva debe ocurrir únicamente durante los primeros seis períodos completos de tiempo de espera, después de la recepción de la trama que indica el inicio del período de contienda.

Mecanismo GTS

El mecanismo GTS permite a un dispositivo acceder al medio sin contención mediante la asignación de ranuras de tiempo del SF exclusivamente dedicadas al dispositivo que las solicita. La prestación de este servicio, se da bajo las siguientes condiciones de operación:

1. El GTS es asignado por el coordinador y se puede utilizar solamente para que un dispositivo se comunique con él.

2. Un dispositivo puede tener asignado sólo un GTS del SF para transmitir o recibir.

3. Cada GTS puede contener una o más ranuras de tiempo y un SF puede contener hasta un máximo de siete GTS.

4. Las ranuras GTS mantienen una dirección, pueden usarse para transmitir del coordinador al dispositivo o viceversa pero no ambos.

El coordinador es responsable del desempeño y la administración del mecanismo y puede retirar los GTS en cualquier momento de manera discrecional. Una de las medidas que el coordinador utiliza para logras este objetivo, es la monitorización de la actividad en el CFP para detectar si un dispositivo ha dejado de utilizar su GTS mediante las siguientes reglas:

- Para un GTS de transmisión, la MLME del coordinador supone que un dispositivo ha dejado de utilizar su GTS si no recibe ninguna trama del dispositivo en el GTS al menos cada $2 * n$ super-marcos (n se define abajo).
- Para un GTS de recepción, la MLME del coordinador supone que un dispositivo ha dejado de utilizar su GTS si no se ha recibido ninguna trama ACK del dispositivo en el GTS al menos cada $2 * n$ super-marcos (n se define abajo). Si las tramas en el GTS no requieren ACK, la MLME del coordinador no puede detectar si un dispositivo continúa utilizando su GTS. La salvedad a este escenario es que el coordinador puede retirar el GTS en el momento que lo decida.

$$\begin{cases} n = 2^{(8-macBeaconOrder)}, & \text{si } 0 \leq macBeaconOrder \leq 8 \\ n = 1, & \text{si } 9 \leq macBeaconOrder \leq 14 \end{cases}$$

Los dispositivos que una vez solicitaron GTS para transmitir también pueden solicitar el retiro de esa asignación de tiempo libre de contienda para transmisión. El coordinador después de haber recibido esta solicitud actualiza el descriptor de GTS, removiendo el GTS de la lista de identificación de GTS asignados y reordenando las ranuras restantes de asignar.

Mecanismo de asociación y desasociación

El procedimiento de asociación como muestra la figura 2.13 inicia cuando un dispositivo envía un comando de solicitud de asociación al coordinador de la PAN. El coordinador al recibir la solicitud de asociación decide si admitir al dispositivo y genera la trama de respuesta de asociación. Cuando una asociación se realizó con éxito, la trama de respuesta incluye una dirección corta que el coordinador asigna al dispositivo para intercambiar información en lo sucesivo y esta dirección es agregada a la tabla de nodos asociados del coordinador. Cuando

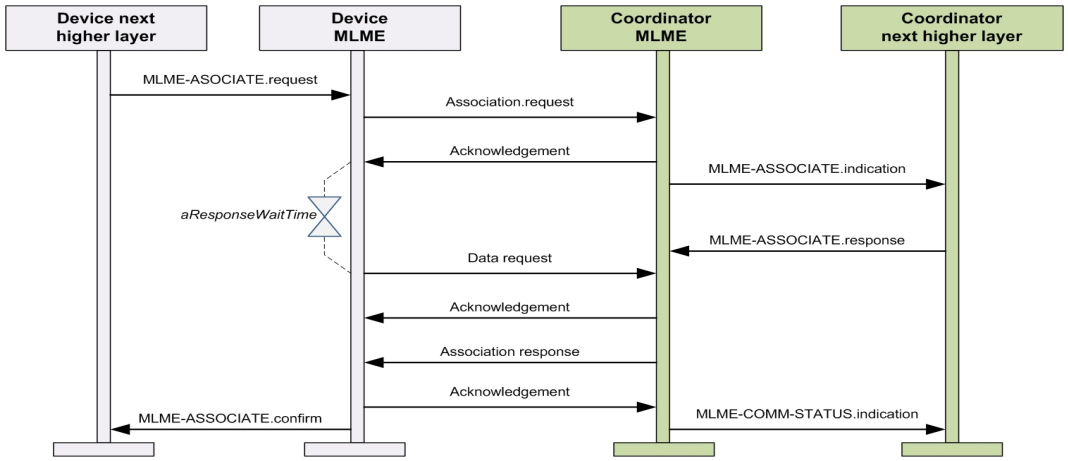


Figura 2.13: Mecanismo de asociación [23]

la asociación no se realizó con éxito, la trama contiene el estatus que indica cual fue la razón por la que la asociación no se logró. La transmisión de la trama de respuesta es indirecta, lo que significa que el coordinador pone la trama de respuesta para transmitir y la dirección del destinatario en la lista de direcciones pendientes. El dispositivo que hizo la solicitud, al recibir la trama con su dirección asignada transmite un comando de solicitud de datos, a lo cual, sigue un mensaje de reconocimiento por parte del coordinador, y después envía la respuesta de asociación exitosa o fallida. La MLME del dispositivo manda un mensaje de reconocimiento al coordinador y después un mensaje de confirmación de la asociación hacia la capa superior al mismo tiempo que la MLME del coordinador envía un mensaje de indicación de estatus a su capa superior y con esto termina la transacción.

Cuando la asociación ha terminado y ha sido realizada con éxito, el dispositivo almacena toda la información concerniente a su nueva PAN mediante la actualización de su MAC-PIB.

2.4.7. Tareas de la capa MAC

Después de la descripción dada arriba acerca de la capa MAC 802.15.4 que a excepción de la explicación del algoritmo CSMA/CA ranurado, ha sido tomada de [23], se listará en términos generales las principales tareas que esta capa realiza:

1. Generación de mensajes de señalización (*beacons*) en el caso de nodos coordinadores.

2. Sincronización de la red mediante mensajes de señalización de inicio y fin de SF.

3. Soporte de asociación y desasociación en un nodo coordinador.

4. Soporte de seguridad.

5. Implementación del mecanismo CSMA/CA para acceso al medio.

6. Administración y mantenimiento del mecanismo GTS.

7. Generación de enlaces confiables entre dos entidades MAC.

2.5. Resumen del capítulo

Después de haber descrito las tecnologías de comunicación inalámbrica que cada vez tienen más y mejores implementaciones en diversos proyectos de la industria, la educación, el gobierno y la medicina entre otros sectores y después hacer el estudio acerca de las herramientas de software que ayudan al desarrollo de sistemas embebidos en dispositivos que forman redes de sensores inalámbricas, se justificó porqué se ha elegido la plataforma MAC de Freescale para el desarrollo del proyecto en lo que respecta a la red de sensores inalámbrica. Cómo fue mencionado anteriormente, MAC de Freescale implementa el estándar IEEE 802.15.4, de allí que se haya hecho la descripción de los conceptos básicos de la capa física y la capa de control de acceso al medio, tal y como lo propone el estándar en la versión 2006. Esto se hace con el propósito de que se comprenda cómo funcionan las dos primeras capas del sistema que se propone en este trabajo. El capítulo 3 describe la forma en que ha sido desarrollado el sistema completo, hace mención de las dos primeras capas y las configuraciones que fueron elegidas en la creación de la plantilla base pero sin entrar en los detalles técnicos de la capa uno y dos que han sido explicados en este capítulo. Se concentra en describir a detalle la forma en que operan las capas superiores del sistema.

Capítulo 3

Construcción del Sistema

3.1. Introducción

Este capítulo presenta el diseño y funcionalidad del sistema que ha sido desarrollado para cumplir con los objetivos propuestos. Su diseño está basado en la arquitectura del modelo OSI, de tal manera que se han desarrollado componentes que por su función, se ubican en algunas de las capas que define como marco de trabajo dicho modelo. La descripción gráfica del sistema puede observarse en la figura 3.1, la cual presenta cinco bloques o niveles dentro de los cuales se encuentran los componentes que realizan las tareas para que el sistema cumpla con las especificaciones requeridas. Cada nivel representa una capa, excepto el primero que contiene la capa física y la capa de control de acceso al medio, las cuales se ha preferido dejar en un solo bloque, debido a que no es desarrollo propio, sino la versión 1.0.1 de MAC 802.15.4 que Freescale pone a disposición de los desarrolladores para que éstos la tomen como plataforma base en el desarrollo de aplicaciones para redes de sensores inalámbricas.

El propósito principal es comunicar una red de sensores inalámbrica basada en el estándar IEEE 802.15.4 con un sistema de monitorización y control en un portal web. Los dispositivos que forman la red inalámbrica se encuentran acoplados a pacientes con movimiento en un área determinada de cobertura y toman los signos vitales de éstos últimos para transmitirlos a una base de datos, de donde las personas que están al cuidado de su salud tienen la posibilidad de leerlos. En términos generales, la red de sensores tiene la responsabilidad de transmitir los

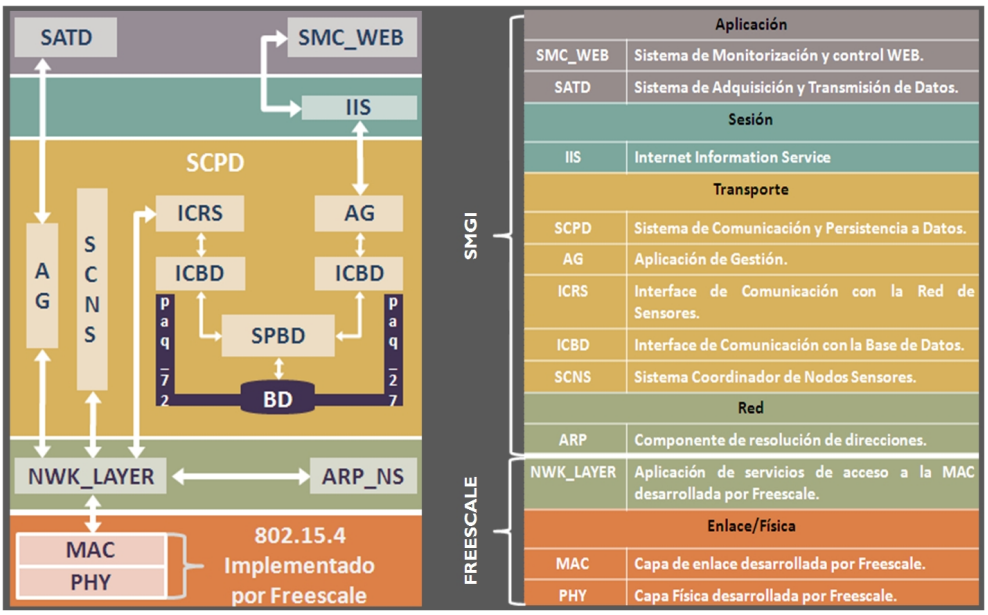


Figura 3.1: Arquitectura multicapa del sistema

signos vitales cada cierto intervalo de tiempo y emitir alertas en caso de ser necesario, mientras que los usuarios del sistema web pueden realizar acciones como:

- Establecimiento de intervalos de muestreo.
- Establecimiento de límites de tolerancia donde los signos vitales no son considerados como anormales.
- Monitorización de los signos vitales en tiempo real.
- Monitorización de los signos vitales que han sido almacenados en la base de datos con anterioridad.
- Gestión de alta, baja y edición de datos de los pacientes, nodos coordinadores y nodos sensores que participan en la red, entre otras.

El diseño del sistema y las estrategias que se siguieron para construirlo obedecen principalmente a tres problemas que se constituyeron en los retos a resolver:

- El establecimiento de comunicación del dispositivo con el sistema de monitorización debido a la inexistencia de un estándar que comunique redes de sensores inalámbricas 802.15.4

con redes 802.11x o Ethernet.

- La confiabilidad en la entrega de datos de extremo a extremo entre los dispositivos y el sistema de monitorización, debido a que son datos de salud sobre los cuales los encargados del cuidado de los pacientes deberán tomar decisiones importantes.

- La asociación correcta de estos datos a los registros de los pacientes en la base de datos del sistema, considerando que los nodos sensores se encuentran organizados por celdas donde cada celda tiene un nodo coordinador, sin embargo, éstos pueden cambiar de celda en cualquier momento, lo que trae con alto grado de probabilidad, el cambio de identificador del dispositivo, ya que cada coordinador lo asigna arbitrariamente cuando los nodos solicitan asociación.

Para lograr la comunicación efectiva entre la WSN y el Sistema de Monitorización y Control Web (SMC_WEB), se diseñó un paquete de datos denominado SCNAD_paq, el cual contiene cabecera, cuerpo y fin de paquete. Así mismo se crearon dos componentes: La Interface de Comunicación con la Red de Sensores (ICRS) y el Sistema de Coordinación de Nodos Sensores (SCNS), que se encargan de leer la estructura del paquete e interpretar la información de manera congruente, para posteriormente darle el formato correcto de tal manera que sea almacenada en la base de datos cuando procede de la red de sensores, o asignarla a las estructuras en el nodo sensor cuando la fuente es el sistema de monitorización.

Por otra parte, para dar la confiabilidad deseada a la entrega de datos, se empleó como estrategia, el diseño de componentes que aplican conceptos básicos de las redes DTN y el protocolo TCP, dos tecnologías orientadas a entrega de datos de manera confiable, mediante diversos mecanismos como se explica en su momento en la sección 3.5. El concepto de TCP es aplicado dentro del sistema, tanto en la comunicación entre los dispositivos de la red de sensores donde se encuentra habilitada la opción de espera de ACK por parte de los emisores en la capa de enlace, como en los componentes de la capa de transporte donde se han empleado mecanismos de confirmación de entrega de paquetes con tiempos de retardo y reenvío de los mismos en caso de ser necesario. Así mismo el concepto DTN se ha empleado en dos componentes: 1) Se configuró la plantilla MAC de Freescale para permitir la funcionalidad de envío de paquetes de forma

indirecta, lo que consiste en separar un espacio de la memoria para dedicarlo al almacenamiento temporal de paquetes que no pueden ser entregados de manera inmediata durante un tiempo indefinido o hasta que el búfer se llene y los paquetes más antiguos sean reemplazados por otros recientes y 2) La base de datos mantiene dos colas donde los paquetes son almacenados y permanecen hasta que los sistemas que requieren la información, los solicitan.

Además de la estrategia de implementar los conceptos TCP y DTN, se eligió utilizar la opción de redes señalizadas, lo cual determina que el mecanismo de acceso al medio será CSMA/CA ranurado con opción a GTS, mecanismo que ha sido habilitado para garantizar la comunicación libre de contienda entre los nodos sensores y su coordinador, de tal manera que haya más probabilidad de éxito en la transmisión de la información.

En cuanto a la estrategia para lograr la asociación correcta entre los datos provenientes de los dispositivos y los registros de los pacientes en la base de datos, aún cuando los nodos cambien de celda y por consecuencia de dirección, se realizaron procedimientos basados en los conceptos de IP-MOVIL. La implementación consiste en un componente embebido en el nodo coordinador que mantiene una tabla de direccionamiento lógico que crea registros para relacionar la dirección asignada por el nodo coordinador al nodo sensor durante su asociación, con la etiqueta del nodo que se mantiene de manera fija en el nodo sensor. El componente consulta la tabla de direccionamiento lógico en cada transmisión de paquetes que pasa por el nodo coordinador y hace el correspondiente intercambio de direcciones de tal manera que el paquete sea enviado a destino correcto cuando proviene del sistema de monitorización y a la tabla que corresponde en la base de datos cuando proviene de un nodo sensor.

A partir de aquí se explica cada capa que presenta la estructura del sistema de la figura 3.1, describiendo a detalle los componentes que la integran y los procedimientos que éstos realizan para llevar a cabo las tareas que logran el funcionamiento adecuado del sistema.

3.2. Capa Física y Enlace

A nivel de capa física y capa de enlace, se utilizó la implementación que Freescale hace del estándar IEEE 802.15.4 sin modificación alguna dentro de su estructura ni funcionamiento. Lo

que se describirá a continuación son las características del hardware donde se va a implementar la aplicación de la red de sensores y las configuraciones que la herramienta Beekit permite establecer, a fin de que la plataforma que provee, satisfaga las necesidades de diseño del hardware y al mismo tiempo se adapte a las características de la red, tales como la topología, la cantidad de espacio en los búferes de salida de paquetes y la forma de acceso al medio que se desea, entre otras opciones que el estándar permite elegir.

3.2.1. Física

El hardware utilizado para la implementación de los dispositivos FFD y RFD es la tarjeta MC1319X de Panasonic, mostrada en la figura 3.2 junto al diagrama de componentes y cuyas características listamos a continuación:

- Transceptor IEEE 802.15.4 (MC13193)
- Banda de trabajo ISM a 2.4 GHz.
- 16 canales con 5 MHz de espacio entre cada canal.
- Velocidad de 250 Kbps.
- Puerto RS-232, convertidor Analógico Digital de 10 bits, dos entradas analógicas y 8 puertos digitales I/O.
- Potencia de salida típica de 0dBm (1 mW).
- Sensitividad del receptor: típicamente -92 dBm al 1
- Antena impresa en la tarjeta, con posibilidad de conectar una antena externa.
- Rango de energía de 2.2 a 3.4 VDC sin RS 232 y de 3 a 3.4 VDC con RS 232.
- Corriente DC en Rx/Tx típica de 35 mA y 5uA en bajo consumo.

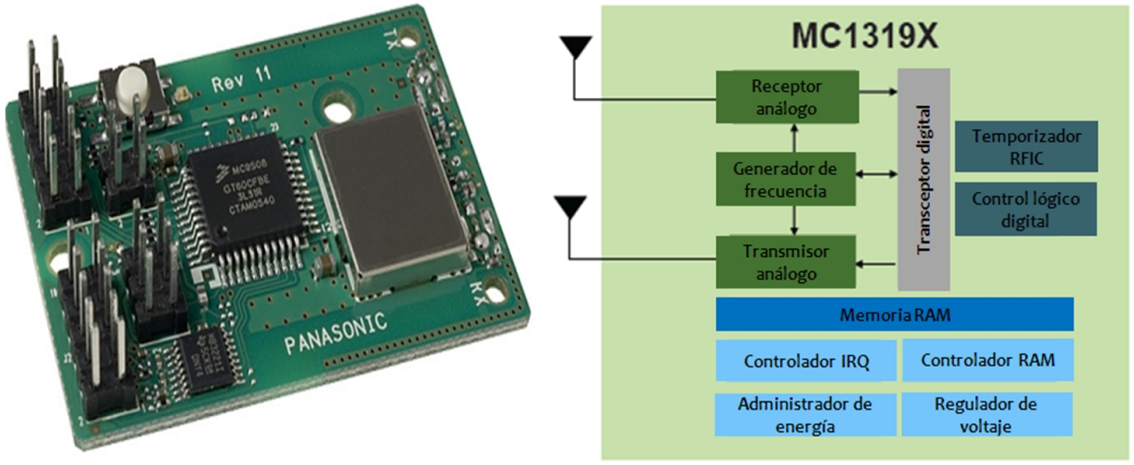


Figura 3.2: Hardware y diagrama de componentes

Características de hardware elegidas en Beekit

- Se eligió la tarjeta MC13192-SARD por ser la tarjeta compatible con el hardware utilizado en este desarrollo.

- Se eligió tener acceso a los LED de la tarjeta que en el caso de la MC13192-SARD son cuatro, mismos que son identificados con LED1, LED2, LED3 y LED4. La tarjeta Panasonic que se usó cuenta con un sólo LED que es utilizado para indicar que el dispositivo está listo para ser activado en el momento que el usuario lo decida, así mismo este LED con la luz intermitente indica al usuario que el dispositivo se ha reiniciado y que debe volver a oprimir el botón indicado para que éste se active y comience a trabajar. El código que se habilita para los LED del dos al cuatro, ha sido eliminado del archivo fuente a fin de disminuir el número de líneas de código y con ello el consiguiente espacio en memoria, lo mismo se ha hecho con el código que se refiere a la pantalla con que cuenta la tarjeta MC13192-SARD pero que no es incluida en la Panasonic que se ha utilizado.

- Otra opción que permite configurar la herramienta, es el trabajo en modo "ahorro de energía", el cual fue activado para la plantilla del módulo destinado al dispositivo RFD. No así con el módulo para el dispositivo FFD, puesto que estará conectado a una fuente de poder.

- La última opción configurada para el hardware del dispositivo FFD que servirá como coordinador de la red, fue la habilitación del puerto serie. La velocidad de transmisión se ha dejado en 19200 bps como dicta el estándar IEEE 802.15.4.

  Es importante acotar que para la tarjeta Panasonic, no basta habilitar la UART en Beekit para que ésta funcione correctamente en la aplicación que se desarrolla. Después de importar con Code Warrior el archivo que genera BeeKit, debe incluirse en el código fuente del archivo App.c dos sentencias como muestra la figura 3.3

  1. PTCDD|=0x40: Establece a uno el bit seis del puerto C de direccionamiento de datos (PTCDD) para indicar al hardware que el puerto va a ser utilizado como un puerto de entrada y salida.
  2. PTCD|=0x40: Establece a uno el bit seis del puerto C de datos (PTCDD) para habilitar el puerto y poder utilizarlo para transmitir y recibir datos.

```
void MApp_init(void)
{

  /* The initial application state */
  gState = stateInit;
  /* Reset number of pending packets */
  mcPendingPackets = 0;

  /* Initialize the MAC 802.15.4 extended address */
  Init_MacExtendedAddress();
  /* register keyboard callback function */
  KBD_Init(App_HandleKeys);
  /* initialize LCD Module */
  LCD_Init();
  /* initialize LED Module */
  LED_Init();
  /* Initialize the LPM module */
  PWRLib_Init();
  /* Initialize the UART so that we can print out status messages */
  UartX_SetBaud(gUartDefaultBaud_c);
  UartX_SetRxCallBack(UartRxCallBack);
/*-------------------------------------------------------------------------------*/
  PTCDD|=0x40; //indica que el puerto C de Direccionamiento de Datos será utilizado para I/O.
  PTCD|=0x40;  //Habilita el puerto C de Datos para ser utilizado como I/O.
/*-------------------------------------------------------------------------------*/
```

Figura 3.3: Configurar y habilitar el puerto serie como I/O en CW

3.2.2. Enlace

La capa de enlace, así como la física, es la implementación que Freescale hace del estándar IEEE 802.15.4. Esta capa se generó con la herramienta Beekit con las siguientes configuraciones:

Topología de red estrella: Esta es la forma que conservará la estructura de la red, teniendo diversos nodos coordinadores diseminados en las salas y cuartos donde pueden moverse los pacientes dentro de un hospital, y nodos sensores que se asocian al coordinador más cercano o con mejor conectividad.

Mecanismo de acceso al medio: En este entorno donde la confiabilidad en la entrega de los paquetes es una de las restricciones de diseño, es importante configurar el acceso al medio de manera que se optimice al máximo y los dispositivos tengan la oportunidad de transmitir información al nodo coordinador de la manera más efectiva y oportuna posible. La forma de lograr esto con el estándar IEEE 802.15.4 tomando en cuenta que las celdas no tendrán una gran cantidad de nodos, es habilitar una red controlada por señalizadores y establecer un SF con GTS habilitado. Esta configuración permite utilizar CSMA/CA ranurado en el período de acceso al medio por contienda en lugar de CSMA/CA no ranurado que es el mecanismo que utilizan las redes sin señalizadores, siendo este último el de menor desempeño [12]. Por otra parte al habilitar la opción GTS se está dando a los dispositivos, la capacidad de solicitar al coordinador la asignación de ranuras de tiempo libres de contienda para transmitir información relevante de manera síncrona, característica que impacta directamente en el desempeño confiable de la red.

Servicio de ACK habilitado: Esta configuración obliga a que por cada transmisión de un paquete entre un nodo coordinador y un nodo sensor, exista la confirmación respectiva de su recepción, de lo contrario la retransmisión del mismo se realiza cuando el tiempo de espera ha transcurrido.

Modo de direccionamiento de 16 bits: Esta configuración permite que el intercambio de paquetes una vez que un nodo ha sido asociado al coordinador, sea con un identificador de 16 bits, mientras que la dirección de 64 bits que mantienen los dispositivos sea utilizada únicamente para realizar la asociación cuando un nodo ha quedado desasociado e intenta asociarse al coordinador que le ofrece mayor conectividad en su área de cobertura.

3.3. Paquetes de datos

La comunicación entre la aplicación de usuario final y los dispositivos de la red de sensores se lleva a cabo con un paquete denominado SCNAD_paq. Este paquete toma como alias "paq_72" cuando su fuente es el Sistema de Monitorización y Control Web, y es llamado "paq_27" cuando su fuente es el Sistema de Adquisición y Transmisión de Datos (SATD) embebido en los dispositivos de la red de sensores.

Dentro de la red de sensores el paquete contiene dos bytes de encabezado y una carga útil de 32 bytes y va encapsulado en la carga útil del paquete 802.15.4 como lo muestra la figura 3.4.

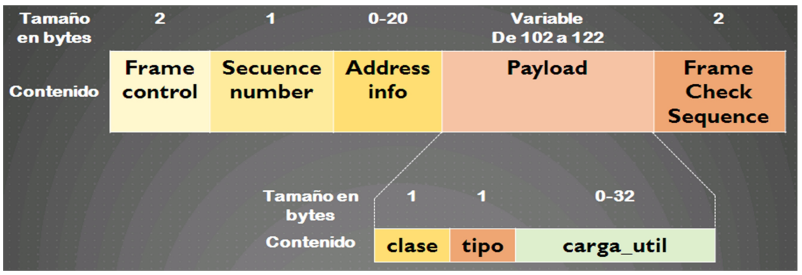


Figura 3.4: SCNAD_paq

Los dos primeros octetos de encabezado permiten la identificación del paquete en las diferentes instancias por donde este transita a través de la red, el primero es la clase a la que el paquete pertenece y el segundo es el tipo de paquete dentro de la clase. La tabla 3.1 presenta las clases, tipos, códigos y descripción de cada uno de los paquetes que forman parte del protocolo de comunicación entre las capas.

Cuando el paquete se encuentra en la ICRS, ya se han agregado, cuatro campos más en el encabezado y fin de la trama como se muestra en la figura 3.5 y se describe a continuación:

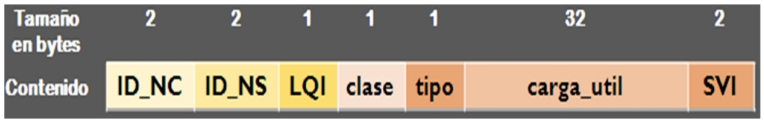


Figura 3.5: SCNAD_paq en ICRS

Clase	Tipo	Código	Descripción
E: Error Código: 0	NSNE	5	NS no encontrado
	ESVNR	7	Establecimiento de signos vitales no realizada
	AIMNER	8	Actualización de intervalos de muestreo no realizada
	LSV	15	Lectura de signos vitales no realizada
	ADMNR	17	Activación-Desactivación no realizada
MC: Mensaje de Confirmación Código: 1	RSV	2	Rangos de signos vitales establecidos
	IMA	3	Intervalos de muestreo actualizados
	SVR	4	Signos vitales recibidos
	AR	5	Alerta recibida
	ERR	6	Estado de red recibido
	CNSA	7	Confirmación de NS asociado
	ARE	8	Asociación realizada con éxito
	ADMR	11	Activación-Desactivación de monitorización realizada
	LSVR	12	Lectura de signos vitales recibida
	AIM	13	Actualización de intervalos de muestreo
	NCLR	14	Nodo Coordinador listo
MD: Mensaje de Datos Código: 2	SV	1	Signos vitales
	EA	2	Emisión de alerta
	ERNC	3	Estado de red de Nodo Coordinador
MSC: Mensaje de Solicitud de Configuración Código: 3	ERSV	1	Establecer rangos a signos vitales
	AIM	5	Actualización de intervalos de muestreo
	ADM	6	Activación-Desactivación de monitorización
MSI: Mensaje de Solicitud de Información Código: 4	ER	1	Estado de red
	LSV	2	Lectura de signos vitales

Tabla 3.1: Descripción de paquetes

- ID_NC que es el identificador del NC que encaminará el paquete hacia el NS o a la ICRS,
- ID_NS como identificador del NS al que se dirige o del que proviene el paquete,
- LQI que es el campo donde se guarda la calidad de enlace con la que el paquete llegó del NS al nodo coordinador, por lo que estará en cero cuando el paquete provenga del SMC_WEB o servidor de la base de datos y
- SVI (Suma de Verificación de Integridad) que contiene la suma de los valores asccii de los caracteres que forman el paquete, de tal manera que cuando éste llegue a la ICRS se haga un proceso de Comprobación de Redundancia Cíclia (CRC, siglas en inglés) de capa cuatro que permita verificar que el paquete no ha sufrido alteración, sino que se conserva tal y como fue generado desde el principio de la transmisión.

Los campos agregados al paquete como lo muestra su descripción arriba tienen el propósito de permitir control en cuanto a integridad, calidad del enlace y direccionamiento en esta

instancia del recorrido a través de las capas.

3.4. Capa de Red

La capa de red en este proyecto se limita a prestar servicios de enlace lógico. Contiene entre otros, dos componentes que permiten la asociación de un dispositivo a un nodo coordinador, al mismo tiempo que mantiene la identificación del dispositivo con el paciente al que ha sido asignado en la base de datos.

App_SendAssociateResponse: Componente desarrollado por Freescale que ofrece el servicio de asociación de un dispositivo a un nodo coordinador mediante la asignación de una dirección de 16 bits. Si la asociación es realizada con éxito, esta dirección de 16 bits será utilizada en lo sucesivo para enviar y recibir los paquetes que se transmitan entre ambos nodos.

ARP_NS: Este componente ayuda a mantener la comunicación estable entre el sistema de monitorización, la base de datos y el dispositivo sensor que portan los pacientes. Esto se hace mediante la actualización de una tabla de direccionamiento lógico en los nodos coordinadores. La figura 3.6 muestra como se realiza este proceso.

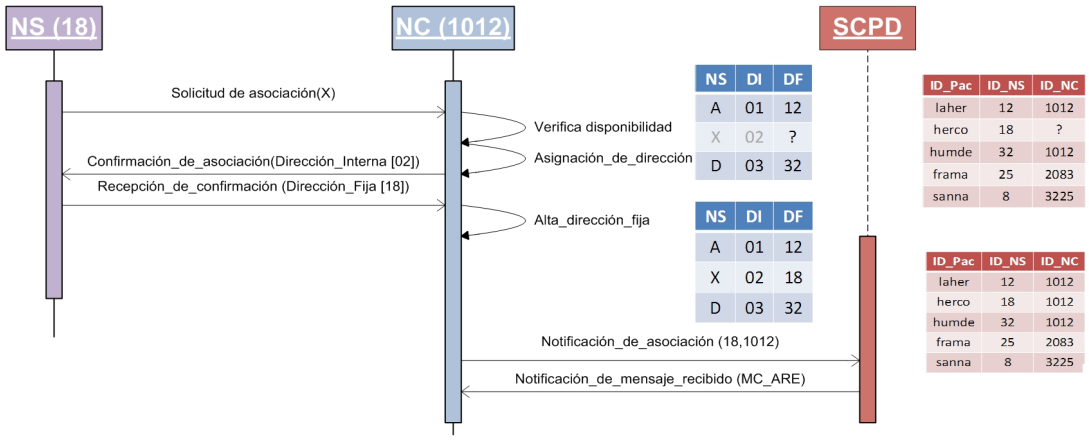


Figura 3.6: Mecanismo de asociación implementando ARP_NS

1. Inmediatamente después de haberse completado la asociación del dispositivo sensor

al nodo coordinador, el componente interviene generando los registros de la tabla de direccionamiento. Lo que hace, es asociar la dirección que ha sido asignada por el nodo coordinador al dispositivo a una etiqueta fija que tiene el dispositivo, la cual a su vez está asociada con un paciente en la base de datos. La figura 3.6 muestra que el dispositivo con etiqueta 18 hace una solicitud de asociación al nodo coordinador con identificador 1012, este verifica en su tabla de direccionamiento y encuentra la dirección 02 disponible para ser asignada a un dispositivo que la solicite, de esta manera el nodo coordinador confirma la asociación enviando la dirección 02 al dispositivo. Cuando el dispositivo la recibe envía una confirmación con su etiqueta para que el nodo coordinador mediante el componente ARP_NS la agregue a su tabla de direccionamiento, asociándola con la dirección interna 02. Después, el nodo coordinador manda el mensaje al sistema de comunicación y persistencia de datos (SCPD) para notificar que el dispositivo con etiqueta 18 ha sido asociado a él, el (SCPD) a su vez, hace la actualización en la base de datos y de esta manera, paciente, dispositivo y nodo coordinador quedan vinculados; como fin del proceso el SCPD envía un mensaje de confirmación de recepción del mensaje de asociación realizada con éxito y la transacción se da por terminada. Este proceso se repite cuando el dispositivo cambia de nodo coordinador y de esta manera se asegura que los paquetes de configuración sean recibidos por el dispositivo correcto y los paquetes de datos sean relacionados en la base de datos con el paciente al que corresponden independientemente del nodo coordinador al que se encuentren asociados.

2. Después de hecha la asociación, el componente ARP_NS participa en cada transmisión de paquetes que vaya desde los dispositivos que forman la red de sensores hacia el sistema de monitorización y viceversa. Cada paquete al pasar por el nodo coordinador sufrirá un cambio en su dirección destino o fuente según corresponda. Este cambio será determinado por la tabla de direccionamiento lógico de la siguiente manera: Cuando el paquete sea transmitido hacia la red de sensores, el componente cambiará la dirección destino del paquete, la etiqueta del dispositivo será sustituida

por la dirección de 16 bits que el nodo coordinador le asignó como dirección local, mientras que cuando el paquete vaya en dirección al sistema de monitorización, la operación será inversa, el paquete traerá la dirección de 16 bits que le asignó el nodo coordinador al dispositivo como dirección fuente, y esta será cambiada por la etiqueta del dispositivo de tal manera que cuando llegue a la base de datos, sea identificado correctamente con un paciente y la información llegue de manera congruente al sistema de monitorización.

3.5. Capa de Transporte

La capa de transporte es la capa que cuenta con el mayor número de componentes, esta capa tiene la responsabilidad de proporcionar confiabilidad a la entrega de información entre el componente de la capa de aplicación de la red de sensores y su similar del lado del servidor web. Para lograr la confiabilidad deseada la capa de transporte se diseñó tomando en cuenta los conceptos básicos de operación de las redes DTN y el protocolo TCP. A continuación se explica cómo se aplican estos conceptos y en que componentes se implementan.

3.5.1. Redes Tolerantes a Retardos (DTN)

Las DTN, son aquellas redes que se les provee de un capa llamada "bundle" para resolver los problemas de transporte relacionados con la pérdida de paquetes debido a retardos en la comunicación por congestión, por caídas de la red, alto grado de bit-error y tasas de transferencia asimétricas [9]. Para solucionar el problema de pérdida de paquetes o reenvíos innecesarios, las redes DTN utilizan nodos con capacidad para almacenar datos y enviarlos en el momento que la conexión al siguiente salto es restablecida. La estrategia es tener uno o más nodos DTN en cada región de la red para hacer las tareas de retransmisión de la información, tan pronto como sea posible.

El sistema cuenta con tres componentes que incluyen procesos para el resguardo de paquetes y envíos posteriores como en las redes DTN. Los componentes están distribuidos entre los nodos sensores, los nodos coordinadores y la base de datos, siendo este último el que da más soporte al

resguardo de paquetes debido a que los datos pueden estar almacenados por un largo período de tiempo y el espacio con que se cuenta es suficiente como para esperar por la lectura de paquetes el tiempo que sea necesario, sin el correr el riesgo de saturación o desbordamiento del búfer.

Procesos en la WSN:

Durante la configuración de las plantillas que se usaron como base del desarrollo de las aplicaciones para los nodos coordinadores y sensores, se especificó que el programa incluya la técnica de transmisión de manera indirecta. Esta técnica consiste en hacer un espacio de memoria suficiente para el almacenamiento de una determinada cantidad de paquetes cuando estos no pueden ser enviados de manera inmediata a través del radio, lo cual puede ser causado por caída de la red o por congestionamiento de la misma. Para este sistema se determinó que el espacio de memoria de resguardo sea equivalente a cuatro paquetes, con esto se logra que los nodos sensores y coordinadores tengan tolerancia a retardos, lo que se traduce en confiabilidad en la entrega de paquetes aplicando el concepto básico de las redes DTN.

Colas de paquetes en la BD:

Cuando los paquetes están fuera de la red de sensores, se mantienen en dos colas que residen en la base de datos. Los paquetes que van del nodo sensor a la aplicación web, pasan por la cola llamada paq_27 y los paquetes que van de la aplicación web al nodo sensor pasan por la cola paq_72 y se mantienen en este lugar hasta que pueden ser leídos por los respectivos componentes que finalmente los conducirán a su destino. La retención de los paquetes en estas colas persistentes y su permanencia hasta que sean leídos, garantiza también la entrega confiable de datos y completa la funcionalidad de tolerancia a retardos como fue la propuesta para el sistema.

3.5.2. Protocolo de Control de Transmisión (TCP)

El protocolo TCP ha sido diseñado para usarse como protocolo de gran fiabilidad entre máquinas que se conectan para formar redes de comunicación de computadoras por conmutación de paquetes. Es un protocolo extremo a extremo orientado a conexión, proporcionando

comunicación interprocesos fiable entre pares de procesos de computadoras conectadas a redes informáticas de comunicación distintas pero interconectadas [24]. Los conceptos más importantes de TCP que ayudan a garantizar la entrega de paquetes entre pares de manera confiable son:

1. La utilización de números de secuencia en cada octeto transmitido exigiendo una confirmación positiva (ACK) por parte del protocolo TCP receptor. Si no se recibe la confirmación ACK dentro de un intervalo de temporización, los datos se retransmiten. En el receptor los números de secuencia se utilizan para ordenar aquellos segmentos que puedan haberse recibido de forma desordenada y para eliminar los duplicados.

2. La verificación de daños en los datos al añadir una suma de comprobación a cada segmento transmitido, comprobando esta en el receptor y descartando los segmentos dañados.

3. El establecimiento de conexiones ente pares que desean transmitir información mediante un mecanismo de negociación inicial con números de secuencia temporizados, para evitar la inicialización errónea de conexiones.

Este sistema implementa como en TCP, el concepto de reconocimiento de paquetes y la verificación de daños o integridad de los mismos. Esta funcionalidad se logra mediante el uso de dos componentes que se describen a continuación:

Aplicación de Gestión (AG)

Este componente denominado Aplicación de Gestión reside en dos partes dentro del sistema. Del lado del servidor web se encuentra por debajo de la capa de sesión y del lado de la red de sensores se encuentra por debajo de la aplicación denominada SATD. En ambas partes del sistema tiene como tarea fundamental, asegurarse de que la información ha llegado al destino deseado y enviar el mensaje de confirmación correspondiente acorde con la definición que se presenta en la tabla 3.1 respecto a los diversos mensajes que pueden ser empleados en el sistema.

La figura 3.7 muestra como el componente AG que trabaja del lado de la red de sensores envía un paquete de alerta que debe ser visto en el monitor de alertas de la aplicación web. El

paquete pasa por tres componentes (SCNS, ICRS y AG_SCPD) antes de llegar a la cola de paquetes en la base de datos donde se mantendrá por tiempo indeterminado, hasta ser leído por la interface de comunicación que le dará salida al monitor de alertas en la aplicación web. Sin embargo, cuando el paquete alcanza la base de datos, el componente AG_SCPD transmite un mensaje de confirmación al nodo sensor que emitió el paquete, para que éste sepa que el paquete ha sido recibido y de por terminada la transacción de manera satisfactoria.

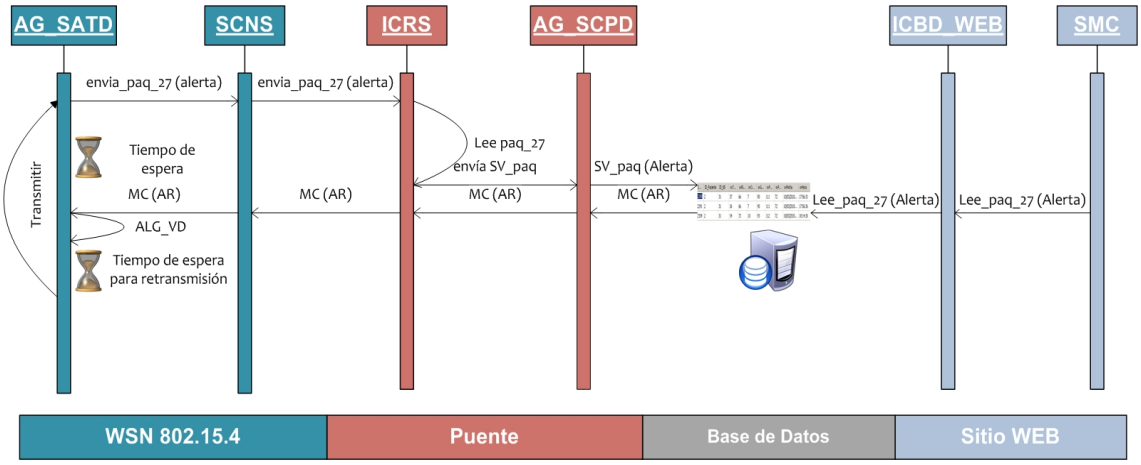


Figura 3.7: Mecanismo de envío de paq_27

El paquete de confirmación debe ser leído antes de que el tiempo de espera definido para este proceso se haya agotado, de lo contrario el dispositivo ejecuta el algoritmo de ventana deslizante (VD_SMGI) que como muestra la figura 3.8, controla los tiempos de espera en los que se realizan los reenvíos necesarios con la esperanza de tener éxito en la entrega del paquete. El tiempo de espera máximo que implementa el componente, fue determinado a partir del promedio de tiempo que se tarda un paquete en ir y regresar de extremo a extremo del sistema.

El algoritmo inicia después de agotado el tiempo de espera, asignando un valor $k = 0$ que servirá para calcular el tiempo de espera $TE = 2^k$ que deberá darse para el reenvío del paquete. Una vez que el paquete fue reenviado (como se puede observar, el primer intento se hace un segundo después de que se ha agotado el tiempo de espera), se espera nuevamente el mensaje de confirmación y si este no se recibe antes de que finalice el tiempo indicado, k se incrementa en uno y el proceso vuelve a repetirse hasta que el valor de k sea igual 3 o se haya recibido el mensaje de confirmación indicando que la entrega del paquete se ha realizado con éxito.

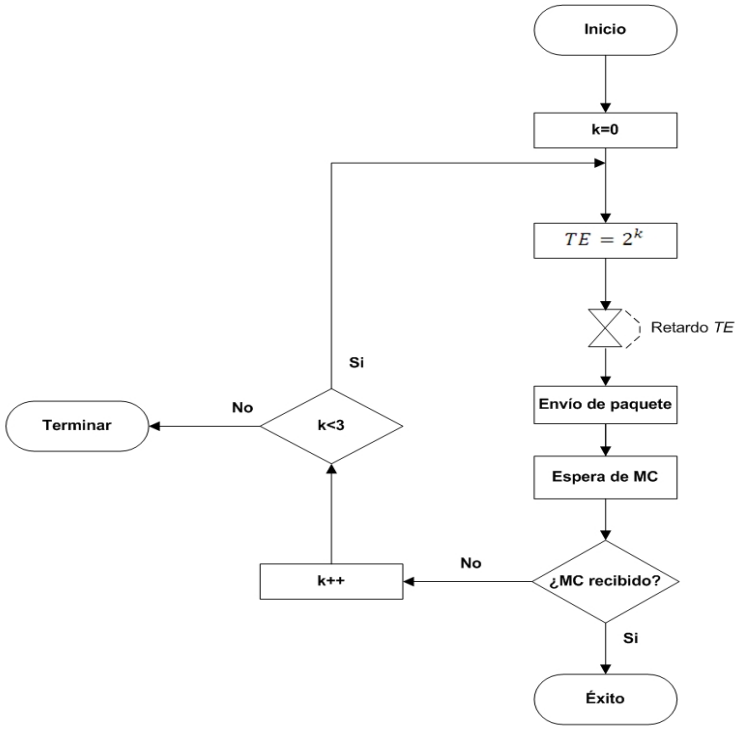


Figura 3.8: Algoritmo de ventana deslizante de SMGI

Del lado del servidor se implementa un mecanismo similar para asegurar la confiabilidad en la transmisión de la información. Un usuario puede enviar un mensaje desde del SMC_WEB hasta el SATD con el fin de solicitar información de los signos vitales de un determinado paciente o bien solicitando el estado de la red, también puede enviar mensajes de configuración de parámetros que regulan el comportamiento del nodo sensor en lo que respecta a intervalos de muestreo, límites en los que los signos vitales de un paciente se consideran estables o puede mandar una orden de control como activar o desactivar la monitorización de un paciente. En todos los casos antes mencionados el componente AG inicia un temporizador para determinar un tiempo de espera en el que se debe recibir un mensaje de confirmación de recepción de paquete por parte del nodo sensor, si este llega, se envía una notificación al usuario de que los datos fueron actualizados de manera exitosa o de lo contrario el mensaje dará la descripción de por qué la instrucción no ha tenido éxito, en este último caso el usuario tendrá la decisión de intentarlo nuevamente o abortar la operación. Por otra parte, la base de datos mantiene una copia de los valores con los que los parámetros del nodo sensor han sido configurados de tal

manera que cuando un usuario requiere saber o utilizar estos valores, no es necesario hacer la solicitud de la información al dispositivo, la consulta se hace directamente a la base de datos, lo que reduce el tráfico y el consumo de energía en la red de sensores. Debido a esta última consideración, cuando el mensaje que se envía es de configuración o control, la actualización de la base de datos respecto al nuevo estado del nodo sensor en cuestión se hace una vez que el mensaje de confirmación ha llegado, de tal manera que se mantenga la congruencia entre los datos que el nodo sensor contiene y los que se tienen como copia fiel de estos en la base de datos.

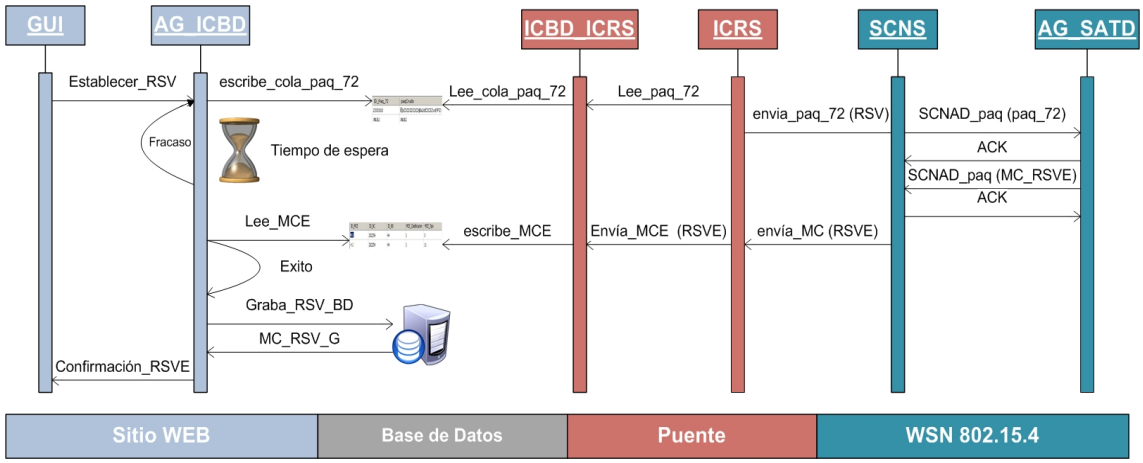


Figura 3.9: Mecanismo de envío de paq_72

La figura 3.9 muestra cómo se hace el envío de un paquete de configuración de rangos de signos vitales para un determinado paciente. El paquete es enviado desde el sistema de monitorización y control hasta el dispositivo que es portado por el paciente al que se desea aplicar el control. El paquete pasa del sistema que monitoriza y controla el usuario al componente AG_ICBD y este lo manda a la cola de paquetes paq_72, al mismo tiempo que pone en marcha el tiempo de espera de confirmación de paquete. Del otro lado de la cola paq_72, se muestra como la Interface de Comunicación con la Red de Sensores (ICRS) envía el paquete al dispositivo a través del nodo coordinador correspondiente, el cual a su vez hace la transmisión al nodo sensor, utilizando el mecanismo de intercambio de paquetes que determina el estándar IEEE 802.15.4. Cuando el dispositivo lo recibe envía el mensaje de confirmación de recepción de paquete que debe llegar a la base de datos para ser leído por el componente AG_ICBD. Si

el paquete alcanzó este destino antes de que el tiempo de espera haya concluido se enviarán dos mensajes: El primero, será un paquete a la base de datos con los nuevos parámetros establecidos a fin de que los datos de ésta armonicen con los que para ese momento el nodo sensor ya tiene establecidos, el segundo mensaje será al SMC_WEB para comunicarle al usuario que la operación realizada ha sido exitosa. Por otra parte, si el tiempo de espera se agotó, la operación se aborta enviando un mensaje al SMC_WEB especificando que la nueva configuración no se realizó con éxito y otro mensaje al nodo sensor para que restablezca los parámetros anteriores a fin de conservar la congruencia en la información.

Sistema Coordinador de Nodos Sensores (SCNS)

Este sistema mostrado en la figura 3.10, está encargado de interpretar los paquetes que llegan de la red de sensores y ponerlos en el formato correcto para su envío al sistema de monitorización y control y viceversa.

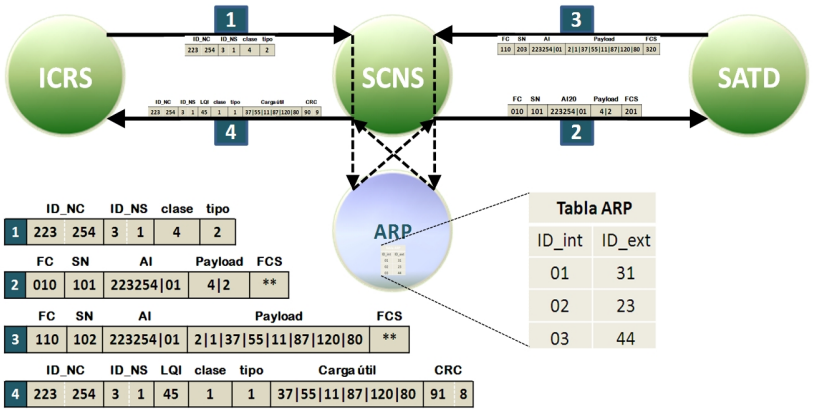


Figura 3.10: Gestión de paquetes en SCNS

La especificación de las tareas que realiza el componente puede describirse de la siguiente manera:

- Cuando el paquete procede de un nodo sensor y se dirige al sistema de monitorización, el componente lee la trama encapsulada en la carga útil del paquete 802.15.4 y agrega los campos correspondientes de encabezado y fin de paquete de tal manera que la trama que reciba la ICRS, cuente con los elementos necesarios para darle el tratamiento adecuado

antes de ser enviado a la base de datos.

- Cuando el paquete procede del sistema de monitorización y control, lo convierte en un paquete 802.15.4 y lo envía al nodo sensor.
- En ambos casos, solicita la resolución de direcciones al componente ARP_NS descrito en la sección 3.4 de este capítulo.

Interface de Comunicación con la Red de Sensores (ICRS)

Esta interface se encuentra ubicada del lado del servidor. Es un componente que está constantemente verificando la actividad en la red de sensores y en la aplicación web, de esta manera, cuando una de las dos entidades envía paquetes a la otra, el componente los captura y hace la gestión correspondiente para que estos alcancen su destino. Para realizar esta tarea mantiene un temporizador que activa un proceso cada cierto intervalo de tiempo, el cual se encarga de leer, por un lado la cola paq_72 y por otro, la actividad que se manifiesta en el puerto de la computadora por donde se reciben los paquetes desde la red de sensores.

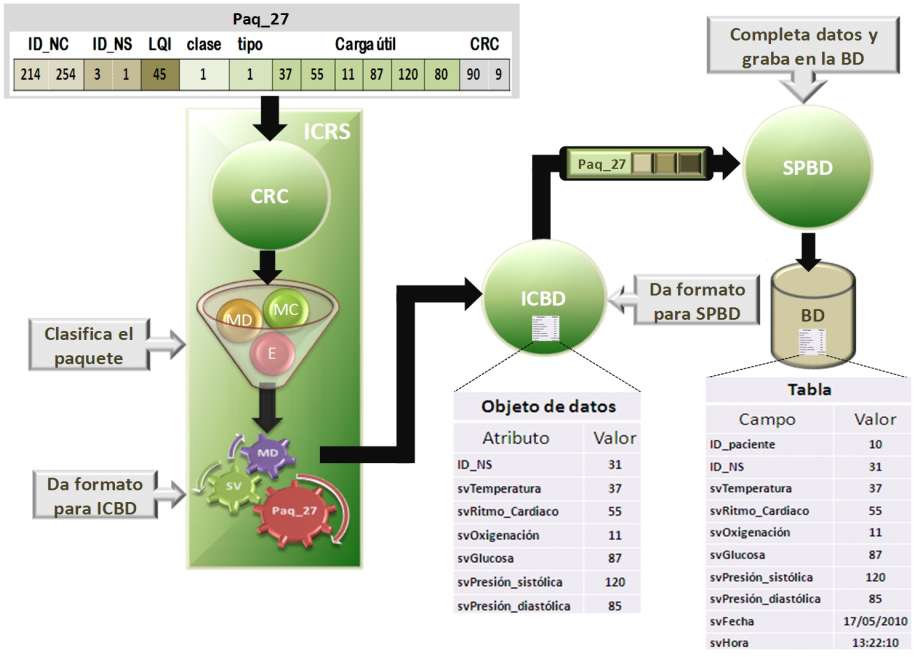


Figura 3.11: Proceso de Pac_27 en ICRS

La figura 3.11 muestra cómo un paquete que es recibido desde la red de sensores llega como

una trama de bytes y es leída por la ICRS convirtiéndola inmediatamente en una cadena y pasando esta última a un proceso que verifica la integridad de la información aplicando un algoritmo CRC de capa cuatro. Después de verificada la integridad de la información, la trama pasa a un filtro donde se lee la cabecera de la misma para determinar la clase y tipo de paquete, con estos datos se elige el objeto de datos adecuado para depositar la información y acto seguido se procede a llenar cada atributo del objeto con el dato de la trama que le corresponda. Cuando esta tarea se ha completado, el objeto de datos pasa a la Interface de Comunicación con la Base de Datos (ICBD) y ésta, da el formato necesario para enviarlo a la base de datos mediante el Sistema de Procedimientos Almacenados en la Base de Datos (SPBD), donde finalmente se agregará la fecha y hora en que se ha recibido, para proceder a registrarlo en la tabla correspondiente.

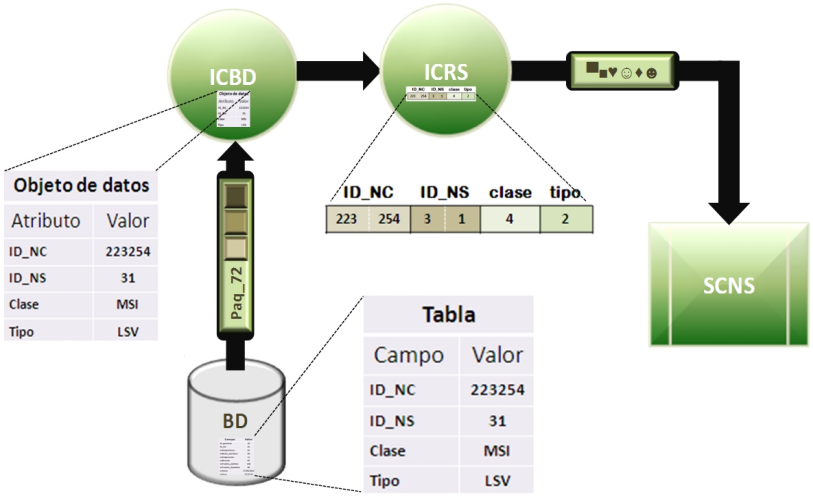


Figura 3.12: Proceso de Pac_72 en ICRS

En la figura 3.12 se muestra como un paquete es tomado de la cola paq_72 por petición de la ICRS mediante la ICBD. Al entrar al componente ICBD, éste lee su encabezado y pone cada campo en su atributo equivalente dentro del objeto de datos que le corresponde, después es enviado a la ICRS, la cual toma los atributos del objeto para ordenarlos y formar una trama que será enviada en formato asccii a través del puerto serie de la computadora a la red de sensores. En la red de sensores un nodo coordinador recibe la trama y siguiendo el orden de los campos, interpreta el mensaje y lo reenvía al dispositivo correspondiente si está asociado a él,

de otra manera regresa un mensaje con la indicación de que el nodo no ha sido encontrado.

3.6. Capa de Sesión

La capa de sesión se deja a cargo de la plataforma ASP.NET y del Servicio de Información de Internet (IIS, siglas en inglés) que es el servidor de sitios de Internet de Microsoft. Como lo explica [7], las aplicaciones ASP.NET se implementan a través de un mecanismo que permite a un navegador o cliente, acceder a una página Web a través de una URL. Todas las páginas y demás archivos que componen una aplicación ASP.NET son puestas a disposición de un usuario a través del IIS siguiendo el flujo de procesos que muestra la figura 3.13

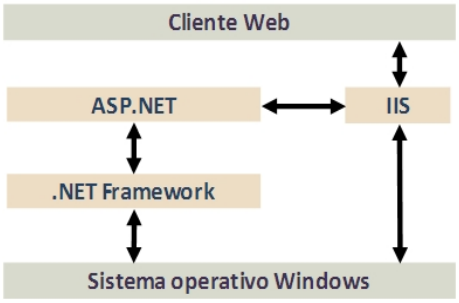


Figura 3.13: Flujo de procesos ASP.NET sobre IIS

Cuando una página de una aplicación ASP.NET implementa código, lo ejecuta a través del Framework .NET, y utilizará IIS para proporcionar el resultado como código HTML en la sesión que mantiene el cliente web; sino, será el IIS quien accederá directamente al sistema operativo para proporcionar el resultado. En cualquiera de los dos casos es el servidor IIS quien termina proporcionando el resultado al cliente.

3.7. Capa de Aplicación

La capa de aplicación en el Sistema de Monitorización y Gestión de la Información SMGI, está integrada por dos componentes que se ubican en ambos extremos del sistema, uno del lado de la red de sensores denominado Sistema de Adquisición y transmisión de Datos (SATD) y

otro denominado Sistema de Monitorización y Control Web (SMC_WEB) que provee la interfaz gráfica de usuario y se encuentra del lado del servidor.

3.7.1. Sistema de Adquisición y Transmisión de Datos (SATD)

Está embebido en el NS, es un componente programado en lenguaje C dentro de la plataforma CW (Code Warrior) de Freescale. Utiliza como base de su desarrollo la aplicación MyApp que genera Beekit e importa CW, la cual provee interfaces que permiten tener acceso a los recursos de la capa de enlace y física, así como las estructuras de la PIB (PAN Information Base) y la estructura que define la trama del paquete 802.15.4. El SATD es un componente que tiene su estructura dividida en dos partes, en la primera se conservan los atributos que pueden ser configurados desde la aplicación web, por ejemplo, el intervalo de tiempo de muestreo por parámetro, los límites donde los signos vitales se consideran estables y el estado de la monitorización que puede ser activo o inactivo, y la segunda parte contiene los métodos que proveen la funcionalidad que se describe a continuación:

1. Ejecución recursiva de una función controlada por temporizador que verifica en cada ciclo el estado activo-inactivo del dispositivo y tiene el control de los contadores de tiempo que determinan el momento de las muestras para los diferentes signos vitales que se desean monitorizar.

2. Obtención de los signos vitales cada cuarto de ciclo normal con el propósito de compararlos con los valores de los límites en los que estos se consideran estables, de tal manera que se envíen alertas si es necesario.

3. Obtención y envío de los signos vitales cuando se cumple el tiempo de muestreo designado por el usuario.

Para probar el sistema se generan números aleatorios con las restricciones necesarias que proveen datos congruentes con lo que serían mediciones de signos vitales reales. Estos números aleatorios llegan al SATD desde la AG del lado del servidor en cada mensaje de confirmación de recepción del paquete de datos y se guardan en los atributos que eventualmente recibirán

los datos que sean adquiridos por los sensores reales. Las pruebas del sistema se muestran en el capítulo 4.

3.7.2. Sistema de Monitorización y Control WEB (SMC_WEB)

Este sistema trabaja sobre el servidor IIS, es un sistema desarrollado en ASP.NET que permite evaluar el comportamiento del protocolo de transporte que se desarrolló para comunicar la red de sensores con una aplicación del lado de un servidor que implementa servicios web. Cuenta con cuatro subcapas que colaboran entre sí para prestar los servicios que el usuario final espera al interactuar con la aplicación. Está desarrollado en el paradigma de programación orientada a objetos por lo que cada subcapa se compone de diversos objetos que se comunican mediante mensajes.

Subcapa de presentación

La subcapa de presentación contiene todos los elementos que constituyen la interfaz gráfica con el usuario, en esta fueron desarrollados los formularios que presentan la información de los signos vitales de los pacientes, la información del estado de la red y los formularios que permiten al usuario interactuar con la base de datos y la red de sensores. Cada formulario fue desarrollado para satisfacer uno o más requerimientos de usuario final como se describe en la tabla 3.2

	Pantalla	**Requerimiento cumplido**
1	frmGestión_NC	Alta, consulta, modificación y eliminación de un NC.
2	frmGestión_NS	Alta, consulta, modificación y eliminación de un NS.
3	frmGestión_Pacientes	Alta, consulta, modificación y eliminación de pacientes.
4	frmMonitor_de_Datos	Monitorizar los signos vitales de los pacientes.
5	frmMonitor_de_Alertas	Monitorizar alertas de signos vitales fuera de rango.
6	frmMonitor_de_Red	Monitorizar el estado de la red.
7	frmRangos_SV	Establecer rangos en los que los signos vitales se consideran estables.
8	frmIntervalos_Muestreo_SV	Establecer tiempos de muestreo para recabar signos vitales.
9	frmActiva_Desactiva_M	Activar y desactivar la monitorización de un paciente.
10	frmLectura_SVTR	Leer los signos vitales de un paciente en tiempo real.

Tabla 3.2: Pantallas y requerimientos

Los puntos del uno al tres son formularios que hacen gestiones comunes de alta, baja y

modificación de registros con la base de datos como el que muestra la figura 3.14. En éstos formularios es donde los usuarios dan de alta todas las entidades que participarán en la red de sensores de tal manera que la información que se capture tenga orden y congruencia, por ejemplo, un nodo coordinador tendrá una ubicación y una dirección para identificarlo dentro de la red, los nodos sensores tendrán también una dirección única que los identificará y estarán asociados a un paciente mediante el identificador del paciente, de tal manera que los datos que lleguen del sensor puedan ser relacionados con el paciente al cual está asociado en la base de datos.

Figura 3.14: Formulario par gestión de datos de un paciente

Del cuatro al seis son formularios que permiten la consulta de información procedente de las muestras tomadas de los pacientes por la red de sensores y almacenada en la base de datos. La consulta de esta información puede ser realizada mediante filtros que permiten seleccionar a los pacientes, los signos vitales que desean monitorizarse y el período de tiempo en el que estos han sido tomados. La visualización de esta información puede ser de forma tabular o gráfica. En el formulario tabular se muestran de manera ordenada todos los datos que se han recabado de un paciente determinado, entre las fechas que el usuario ha seleccionado como filtro para la consulta. La tabla que muestra los datos tiene una vista para cinco registros que siempre serán los últimos que se han reportado entre las fechas especificadas, sin embargo, si la cantidad de datos es más extensa que eso y el usuario desea ver más datos, se pone a su disposición un índice en la parte inferior de la tabla por medio del cual puede moverse entre las páginas que

contienen todos los registros. Un ejemplo de esta página que presenta la tabla con los registros y los índices se puede observar en la figura 3.15.

Figura 3.15: Formulario para vista tabular de datos

Cuando se solicita la información en modo gráfico los registros se representan en una curva formada por cinco puntos distribuidos a lo largo de los registros incluyendo el primero y el último como muestra la figura 3.16.

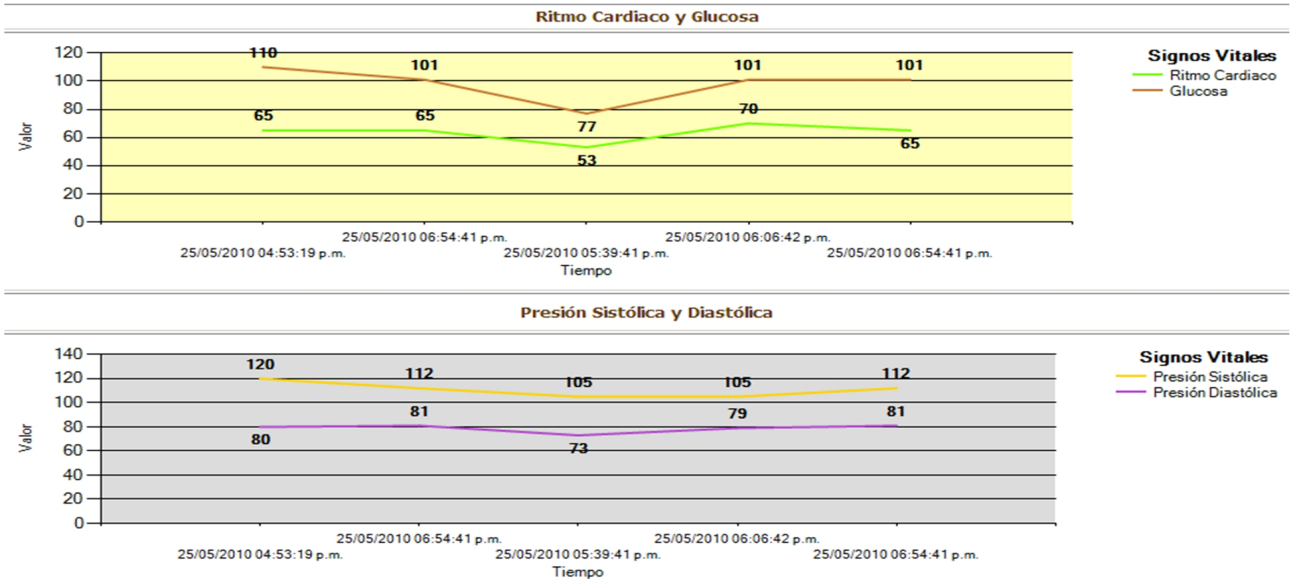


Figura 3.16: Formulario para vista gráfica de datos

Los puntos del siete al diez son formularios como el que muestra la figura 3.17, los cuales

permiten al usuario interactuar con la red de sensores para configurar algunos parámetros de control, entre los cuales se encuentran los intervalos de muestreo de signos vitales para un paciente determinado, la activación o desactivación de monitorización y el establecimiento de límites dentro de los cuales los signos vitales se consideran normales al momento de hacer la lectura, de tal manera que se emita una alerta si uno de ellos se identifica fuera del rango.

Figura 3.17: Formulario para establecer rangos de signos vitales

Finalmente para terminar con esta sección se enumeran otras tareas de aspecto general que la subcapa de presentación realiza:

1. La prestación de servicios de acceso a los diversos formularios de la aplicación, que se logra mediante la implementación de un formulario maestro al que se agregó un menú de hipervínculos que se mantiene persistente en cada uno de los formularios que estén bajo el dominio del formulario maestro.

2. Un alto porcentaje de la validación sobre los datos que el usuario captura está desarrollado sobre la subcapa de presentación mediante controles que ofrece la plataforma .NET, por ejemplo, el control RequiereFieldValidator que no permite el envío de datos al servidor mientras no se haya capturado cierto dato en un control asociado a él, el control RegularExpressionValidator que permite configurar el formato que debe cumplir cierto dato (fechas, código postal, correo electrónico, etc.) antes de ser procesado o el control Requie-

reFieldValidator que no permite enviar datos al servidor hasta que el dato sea capturado en el control al cual ha sido asociado.

3. La persistencia de datos mediante el uso de variables de sesión o cookies para no solicitarlos periódicamente al servidor y tenerlos a disposición en el momento que se requieran.

Subcapa de procesos de aplicación

Esta subcapa, en las aplicaciones multicapa de desarrollo de software tradicionales recibe el nombre de subcapa de lógica de negocio y se amolda al comportamiento del sistema, basándose en los datos provistos por la subcapa de datos, además de actualizarlos según sea necesario. En este sistema, la subcapa de procesos de aplicación cumple con los mismos propósitos, el dar tratamiento a los datos que reciben desde las subcapas de presentación y de datos respectivamente, para que cobren significado al pasar a la subcapa a la que se dirigen. Las tareas más sobresalientes que realiza esta subcapa para colaborar con el funcionamiento de esta aplicación se listan a continuación:

1. Realiza el proceso de validación que no ha sido posible cubrir con los controles que se tienen para este fin en la subcapa de presentación.

2. Interpreta los mensajes de error que regresan desde la base de datos o la red sensores cuando una operación no se realizó de manera exitosa para la subcapa de presentación.

3. Selecciona los datos que deben tomarse en cuenta para crear las gráficas cuando el usuario solicita este tipo de monitorización de los signos vitales.

4. Cuando se despliega el monitor de alertas, identifica los signos vitales que están fuera de los límites establecidos, de tal manera que puedan ser marcados de color diferente para hacerlos evidentes al usuario.

La importancia de esta subcapa dentro de los desarrollos de software, reside por una parte, en la seguridad que otorga al sistema al no hacer evidente el código que muestra los nombres de los objetos de la base de datos en la subcapa de presentación, siendo la más expuesta y

vulnerable a los ataques de usuarios maliciosos, y por otro lado, permite dar mantenimiento de manera más fácil al sistema, cuando este requiere cambios en la lógica de negocio o la forma de tratamiento e interpretación de los datos.

Subcapa de datos

En esta subcapa se encuentran los mecanismos para el acceso y almacenamiento de la información en la base de datos, mantiene la persistencia de los datos mediante transacciones que deben realizarse con consistencia, de tal manera que tanto los datos que ingresan como los que proceden de la base datos sean precisos. En esta subcapa se definen todas las consultas para la generación de reportes y se construyen todas las sentencias que ayudan a realizar las actualizaciones de los datos así como el alta de nuevos registros.

En este sistema la subcapa de datos está dividida en dos partes, una desarrollada dentro de la aplicación de ASP llamada ICBD (Interface de Comunicación con la Base de Datos, de la misma manera que se denomina el componente que trabaja para la subcapa de transporte) y otra desarrollada dentro de la base de datos llamada SPBD (Procedimientos almacenados en la base de datos) que tanto aquí como para la subcapa de transporte, cumple con el propósito de ejecutar todas las sentencias SQL para realizar las transacciones con la base de datos.

Interface de Comunicación con la Base de Datos (ICBD): Es el componente que recibe las peticiones de consulta y actualización de la base de datos por parte de la subcapa de procesos de aplicación. La subcapa de procesos de aplicación envía mensajes a la subcapa de datos pasando como parámetros los objetos que contienen los datos para actualización o los parámetros que se tomarán en cuenta para realizar la consulta solicitada. El objeto que recibe el mensaje, lee los datos y los ordena en el formato correcto para que sirvan como parámetros de los procedimientos almacenados en la base de datos, de tal manera que realicen las transacciones correspondientes.

Procedimientos Almacenados en la Base de Datos (SPBD): Componente embebido en la base de datos que cuenta con los procedimientos que se ejecutarán cuando reciban la solicitud por parte de la ICBD. Los procedimientos son llamados por nombre y reciben

los datos que utilizarán para la transacción en forma de parámetros. Estos procedimientos pueden contener lógica de programación como ciclos, estructuras de decisión, estructuras de selección, operaciones matemáticas, etc., las cuales permiten manipular los datos que serán almacenados o extraídos de la base de datos. Además ofrecen ciertas características que los hacen recomendables y que se listan a continuación:

1. Ayudan a mantener la seguridad de la base de datos, al no exponer en la aplicación las sentencias que realizan las transacciones en la base de datos y que revelan los nombres de los campos, tablas y vistas, entre otros objetos.
2. Ejecutan con rapidez las operaciones debido a que los procedimientos son compilados una vez y después se ejecutan sin compilación cada vez que son invocados desde la aplicación, a diferencia de las transacciones que se construyen en las aplicaciones y cuya sintaxis correcta es comprobada por el compilador de la base de datos cada vez que intentan leer o escribir datos en ésta.
3. Al permitir la manipulación de los datos mediante programación, se pueden realizar operaciones que de otra manera tendrían que programarse en la aplicación, lo que se traduciría en consumo de recursos que pueden ser utilizados para atender otras peticiones que los clientes realizan al servidor WEB.

Subcapa de paquetes de datos

Esta subcapa es transversal a las otras tres y contiene objetos que servirán para transportar los datos entre las primeras. Por ejemplo, cuando la subcapa de presentación desea enviar información a la subcapa de datos, ésta hace uso de un objeto de la subcapa de paquetes de datos, copia los valores que desea enviar a los atributos del objeto, mediante los métodos definidos para estos fines y después envía el objeto como parámetro del método de la subcapa de procesos de aplicación que utiliza, la subcapa de procesos de aplicación hace uso del objeto de datos si es necesario y después lo envía a la subcapa de datos mediante el mismo procedimiento que utilizó la subcapa de presentación.

3.8. Conclusión

Este capítulo describió la forma en que ha sido construido el sistema, a fin de satisfacer los requerimientos inherentes a la naturaleza de la infraestructura en la que debe ejecutarse. Se ha mostrado que con base en conceptos de tecnologías existentes como TCP, DTN e IP-MOVIL y mecanismos aplicados a sistemas con condiciones similares, se han seguido estrategias que permiten transportar datos de manera confiable entre aplicaciones que residen en los extremos de redes 802.15.4 y redes WiFi o Ethernet. Así mismo, se puede garantizar la correcta asociación de los signos vitales de los pacientes a los registros pertinentes en la base de datos, aún cuando los nodos pueden moverse y cambiar de coordinador en cualquier moemento. Para lograr la implementación de las estrategias y mecanismos, se aprovecharon las características de comunicación que ofrece el estándar IEEE 802.15.4 implementado de manera completa en la plataforma MAC 802.15.4 de Freescale que se utilizó como base del desarrollo y a partir de allí se desarrollaron componentes que usan conceptos basados en técnicas que han sido base de muchos desarrollos tecnológicos, como el Protocolo de Resolución de Direcciones (ARP, siglas en inglés), el algoritmo CRC, el algoritmo de backoff de CSMA/CA, entre otros.

Con esto se dará paso al siguiente capítulo, donde se hace la descripción de los resultados obtenidos al realizar las pruebas del sistema.

Capítulo 4

Pruebas y Resultados

Este capítulo presenta los resultados que se obtuvieron después de hacer las pruebas que permitieron medir el desempeño del sistema. Se hicieron dos tipos de mediciones que permitieron hacer ajustes en los parámetros que controlan los temporizadores dentro de los componentes, y a su vez son útiles para establecer los límites del sistema en cuanto a cantidad de nodos por celda e intervalos de tiempo de muestreo para la adquisición de los signos vitales, de tal manera que el sistema se mantenga estable. Las pruebas se realizaron con la capa MAC configurada en CSMA/CA ranurado con GTS.

1. La primera prueba fue: Medir el tiempo que toma enviar un paquete del servidor que mantiene la base de datos a un nodo sensor y recibir confirmación de que el paquete ha sido recibido con éxito.

2. La segunda prueba fue: Medir la cantidad de paquetes que son entregados de manera exitosa a medida que la cantidad de nodos aumenta.

4.1. Tiempo de envío y confirmación de paquete

Esta medición permitió establecer el tiempo de espera máximo para la confirmación de un paquete cuando éste es enviado del lado del servidor hacia los nodos sensores. También permitió establecer el tamaño de ventana en la primera iteración del algoritmo de ventana deslizante que

se presentó en el capítulo tres, como técnica de retransmisión de paquetes desde los nodos sensores cuando no se tiene respuesta de éxito en la entrega por parte del servidor.

4.1.1. Procedimiento

1. Se hicieron diez pruebas, enviando 100 paquetes con espera de confirmación desde el servidor que mantiene la base de datos, hasta un nodo en la red de sensores. Para cada paquete enviado se marcó el instante de envío del paquete y el instante de llegada de confirmación. La diferencia entre estos tiempos se guardó como el intervalo de tiempo que tardó cada transacción.

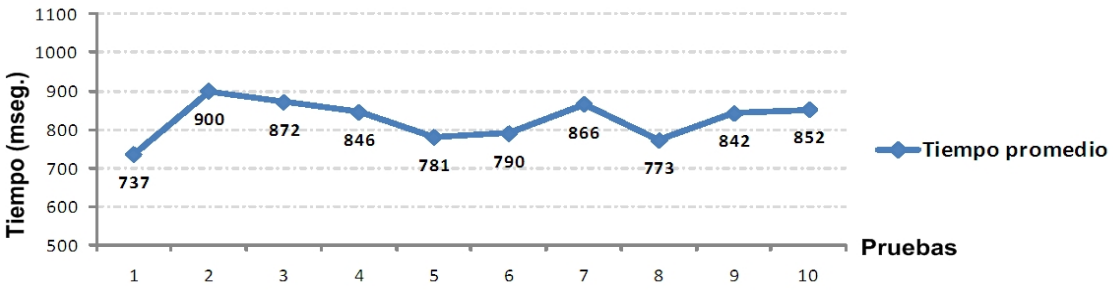
2. Se calculó las medias y la desviación estándar de los intervalos de tiempo para cada una de las diez pruebas.

3. Se calculó la media de las diez medias obtenidas con el proceso anterior y de allí se obtuvo la estimación del tiempo que tarda un paquete en llegar a su destino y recibir la confirmación de éxito.

4. Finalmente se hizo el cálculo de un intervalo de tiempo con 95 % de confiabilidad, para el tiempo que tarda un paquete en ir y regresar del servidor a la red de sensores.

4.1.2. Resultados

La gráfica 4.1 muestra los tiempos de retardo promedio en cada una de las pruebas.



Gráfica 4.1: Tiempos de respuesta del servidor a un nodo.

El tiempo promedio P está dado por:

$$P = \frac{1}{n} \sum_{k=1}^{n=10} p_k$$

donde

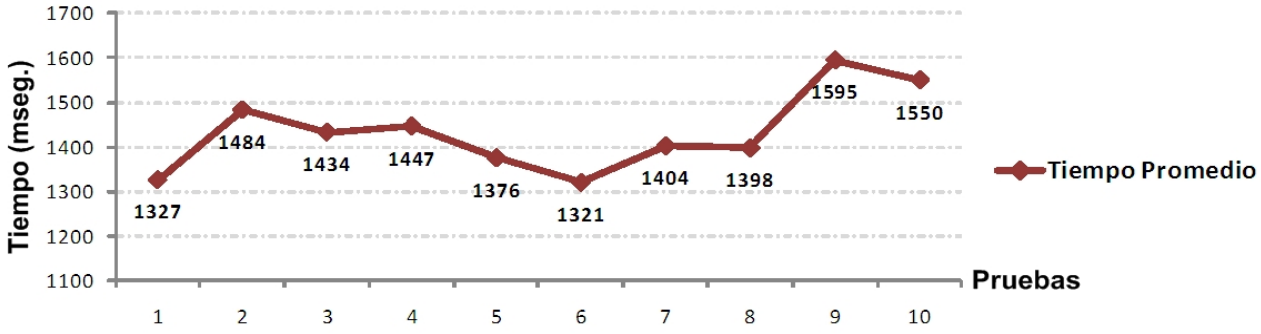
$$p_k = \frac{1}{m} \sum_{i=1}^{m=100} t_i$$

En las ecuaciones anteriores, t_i representa el tiempo de envío y recepción de confirmación de cada paquete en cada prueba y p_k es el promedio de tiempo de cada prueba. Con estas ecuaciones y aplicando las fórmulas correspondientes para una población con distribución uniforme se obtienen los resultados que se presentan en la tabla 4.1.

Dato	**Tiempo (mseg.)**
Tiempo promedio máximo:	900
Tiempo promedio mínimo:	737
Media:	826
Desviación estándar promedio:	367
Intervalo de confianza al 95 %:	598 - 1053

Tabla 4.1: Tiempos de retardo del servidor a la red de sensores

El mismo procedimiento fue utilizado para calcular el promedio de tiempo que tarda el envío y recepción de confirmación de un paquete que se envía a un nodo de la red de sensores, desde el SMC_WEB en una computadora ubicada a un salto del servidor que mantiene el sitio y la base de datos. Los resultados se pueden ver en la gráfica 4.2 y tabla 4.2 respectivamente.



Gráfica 4.2: Tiempos de respuesta del SMC_WEB a un nodo.

Dato	**Tiempo (mseg.)**
Tiempo promedio máximo:	1595
Tiempo promedio mínimo:	1327
Media:	1434
Desviación estándar promedio:	453
Intervalo de confianza al 95 %:	1153 - 1714

Tabla 4.2: Tiempos de retardo del sitio WEB a la red de sensores

Para fines prácticos, el tiempo de espera en el servidor que mantiene la base de datos se dejó en un segundo, mientras que el tiempo de espera de respuesta de un cliente del sitio web se dejó en tres segundos debido a que se pretende servir a clientes remotos, donde no se tiene control de la cantidad de saltos ni del tráfico de red que haya en los puntos de acceso a donde éstos clientes estarán conectados.

4.2. Eficiencia en la entrega de paquetes

La confiabilidad en la entrega de paquetes desde los nodos sensores hacia el servidor de base de datos donde la información se considera segura y persistente, fue la restricción de diseño de mayor importancia en el desarrollo del sistema. Los resultados que a continuación se presentan, proporcionan los datos que permiten establecer los límites en los que el sistema se comporta de manera estable y permiten hacer una evaluación del logro de los objetivos propuestos.

4.2.1. Escenario y Procedimiento

1. Se configuraron ocho nodos para que enviaran un total de 100 paquetes a la base de datos donde fueron asociados a registros de pacientes diferentes.

2. Los nodos se distribuyeron de manera aleatoria en un área de aproximadamente $80m^2$. El espacio donde se hizo este experimento es cerrado y cuenta con computadoras, muebles de madera y metal, y una red con terminales Ethernet y WiFi.

3. Se estableció una ventana de tiempo de 500 segundos para el envío de los paquetes, la cual permaneció constante durante todas las pruebas del experimento. Esta ventana de tiempo

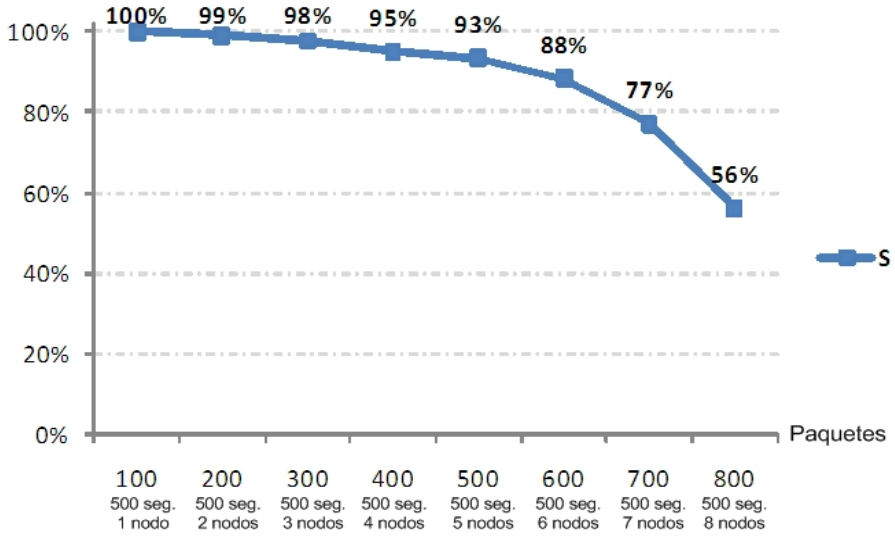
corresponde al envío de un paquete por nodo cada cinco segundos. Con este intervalo de tiempo se garantiza que los temporizadores del servidor que mantiene la base de datos y el temporizador del SMC_WEB activen de manera oportuna los procedimientos que hacen las lecturas del puerto serie y base de datos respectivamente y envíen la confirmación correspondiente a la red de sensores.

4. Las pruebas se realizaron empezando con un nodo enviando 100 paquetes en la ventana de tiempo dada, la segunda prueba incluyó dos nodos enviando 100 paquetes cada uno en la misma ventana de tiempo hasta completar ocho nodos y 800 paquetes transmitidos.

5. Durante cada prueba se registró la cantidad de paquetes entregados, la cantidad de retransmisiones que se hicieron y la cantidad de paquetes que no llegaron a destino.

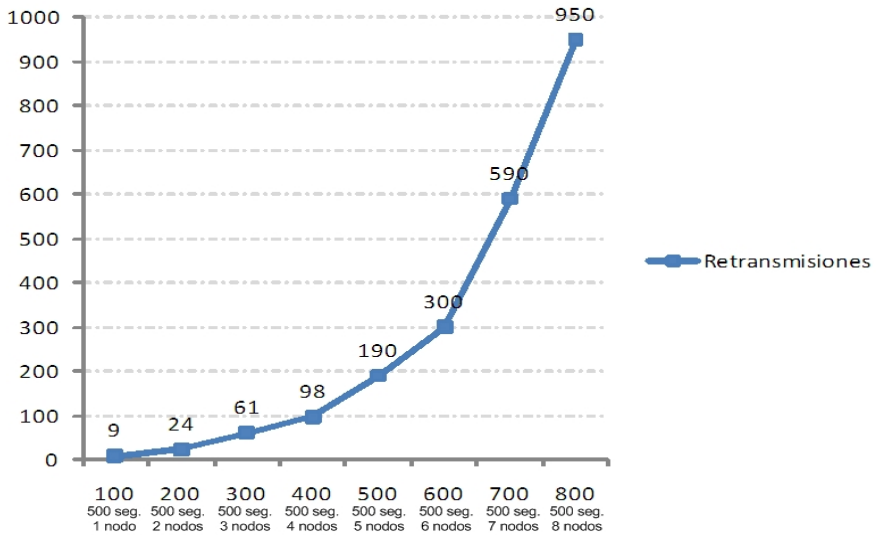
4.2.2. Resultados

La gráfica 4.3 presenta la eficiencia S en la entrega de paquetes. En este caso $S = \frac{pn+pr}{pt}$ donde pn es la cantidad de paquetes nuevos que se recibieron, pr es la cantidad de paquetes que se retransmitieron y llegaron con éxito a su destino, y pt es la cantidad total de paquetes transmitidos.



Gráfica 4.3: Eficiencia del sistema en la entrega de paquetes.

Por otra parte, la gráfica 4.4 muestra la cantidad de retransmisiones que se hacen durante la transmisión de los paquetes. Un paquete se reenvía utilizando el algoritmo de ventana deslizante explicado en la sección 3.5.2 cuando tras su primer envío, el nodo no recibe el mensaje de confirmación correspondiente y se pueden tener un total de ocho intentos. Cuando se alcanza este número de intentos sin tener éxito, el nodo pasa a estado inactivo hasta un nuevo mensaje del SMC_WEB indicándole que puede volver a transmitir.



Gráfica 4.4: Retransmisiones realizadas en el proceso.

4.3. Discusión de resultados

El sistema ha sido desarrollado para servir de intermediario entre una red d sensores inalámbrica que toma signos vitales de pacientes y un sistema de monitorización web de los signos vitales, que además permite el control de ciertos parámetros como tiempos de muestreo y rangos en los que los signos vitales se consideran normales, a fin de que cuando éstos se salgan de los límites establecidos, se emitan alertas. La tarea primaria del sistema es garantizar que los datos recabados por los nodos sensores llegarán a la base de datos con un alto grado de confiabilidad. Para estos fines se emplearon diversos mecanismos que fueron explicados en el capítulo tres, además de aprovechar las características que ofrece el estándar IEEE 802.15.4 para establecer transmisiones confiables entre los dispositivos que forman una red de sensores.

Los resultados que arrojaron las pruebas sobre el tiempo que tarda un paquete desde que es enviado del servidor a un nodo de la red de sensores y se recibe la confirmación de éxito y el tiempo que tarda el mismo proceso desde el SMC_WEB a un nodo de la red de sensores (826 mseg. y 1434 mseg. en promedio respectivamente), permitió establecer el intervalo de tiempo de lectura al puerto serie de la computadora por parte del servidor, y el intervalo de tiempo para lectura de las colas de paquetes en la base de datos por parte del sistema web. Los intervalos de tiempo fueron establecidos en 1 segundo para la lectura al puerto serie y 3 segundos para la lectura a las colas de paquetes en la base de datos, lo que da un amplio margen de recepción y proceso de paquetes con el fin de garantizar la integridad de los mismos.

Las consideraciones anteriores, dieron la pauta para establecer en 5 segundos el intervalo de tiempo que debe existir entre una transmisión y otra de paquetes procedentes de un mismo nodo, de tal manera que el sistema fuera probado cerca de su límite de tolerancia de recepción de paquetes.

Puesto que la verificación de alertas se hace en un cuarto del tiempo establecido como intervalo de muestreo y tomando en cuenta que no se tiene control sobre la cantidad de alertas que serán emitidas, si se establece un intervalo de tiempo de muestreo de 20 segundos, podemos afirmar que:

1. El sistema ofrece al menos 90 % de confiabilidad en la entrega de paquetes atendiendo hasta cinco pacientes.

2. Entre 80 y 90 por ciento de confiabilidad en la entrega de paquetes está garantizada, atendiendo a seis pacientes.

3. La confiabilidad cae de manera exponencial después de siete pacientes, con lo cual el sistema se hace inoperable.

4. La cantidad de retransmisiones en este escenario crece de manera exponencial al incrementar el número de nodos, lo que concuerda con la caída exponencial de la confiabilidad.

Los resultados aquí mostrados muestran el límite de operabilidad del sistema, sin embargo, se puede afirmar con base en pruebas realizadas, que a medida que la ventana de tiempo de los

intervalos de muestreo se hace más grande, la entrega de paquetes tiende a 100 % de eficiencia, con una cantidad pequeña de retransmisiones, relativa al número de paquetes transmitidos. Por ejemplo, las pruebas realizadas con cinco nodos enviando 100 paquetes en una ventana de tiempo de 45 segundos entre transmisiones procedentes de un mismo nodo, dio como resultado la entrega de 500 paquetes con 25 retransmisiones. Esto significa que se puede atender a cinco pacientes con intervalos de muestreo de tres minutos o más para la toma de signos vitales, logrando el 100 % de confiabilidad en la entrega de paquetes.

Los resultados se consideran satisfactorios puesto que el sistema fue desarrollado con la idea de ser implementado en casas habitación, salas de hospitales o clínicas donde los pacientes se consideran estables y para los cuales los intervalos de muestreo de los signos vitales (a decir del personal médico entrevistado para establecer los requerimientos del sistema) están en el orden de horas o fracciones de hora en casos especiales.

Capítulo 5

Conclusiones y Trabajo Futuro

Este trabajo de investigación se ha dado por terminado después de haber hecho las pruebas de desempeño del sistema y haber obtenido los resultados que permiten entender y expresar la manera en que se ha logrado el cumplimiento de los objetivos. Con los resultados también es posible dar respuesta a las preguntas de investigación y hacer recomendaciones para que trabajos posteriores, mejoren y caractericen lo que esta tesis presenta.

5.1. Conclusiones

Las conclusiones de este trabajo están formuladas en función de los objetivos específicos, el objetivo general y las preguntas de investigación que se presentaron en el capítulo uno.

5.1.1. Del cumplimiento de los objetivos

1. Durante la búsqueda de requerimientos de la infraestructura e-salud para la que se ha desarrollado este sistema, se identificaron las necesidades más importantes que rigieron el diseño de la arquitectura y el desarrollo de la funcionalidad del sistema. El sistema de monitorización se requiere para un entorno con bajo estrés, es decir, los intervalos de tiempo entre las muestras de signos vitales están en el orden de horas o fracciones de éstas en casos extremos. Los casos de pacientes en estado crítico, se prefiere que sean atendidos en salas donde existen equipos que utilizan infraestructura cableada, y la mo-

nitorización se hace de manera continua por personas que permanecen de manera física cerca de los pacientes. Lo antes mencionado dio la posibilidad de pensar en el desarrollo de un protocolo de comunicación con alto grado de confiabilidad, debido a que cada dato recabado de un paciente debe formar parte de su expediente médico a fin de dar el mejor seguimiento posible a lo largo de su vida. Por otra parte, los intervalos de muestreo en los rangos de tiempo antes mencionados, dio la posibilidad de incluir la verificación de que los signos vitales permanecen dentro de los límites establecidos por los usuarios de manera personalizada para cada paciente. Esta verificación se hace en un intervalo de tiempo equivalente a un cuarto de tiempo del intervalo de muestreo y hace posible emitir alertas cuando las condiciones lo ameritan.

2. Se hizo revisión de la literatura respecto a los sistemas que han sido desarrollados como herramientas de soporte para la creación de redes de sensores inalámbricas, y se estudió el estándar IEEE 802.15.4. Se identificaron sistemas operativos, sistemas middleware, sistemas de aplicación y plataformas de desarrollo de entre los cuales resaltan Tiny-OS como sistema operativo, por la gran cantidad de aplicaciones que lo han implementado, otros sistemas operativos importantes son: Contiki, Mantis y SOS, el primero por su tendencia a utilizar TCP/IP en las redes de sensores, y los dos últimos, por utilizar memoria dinámica y requerir pocos recursos de hardware para operar. Entre los sistemas middleware se pueden considerar como los más importantes MiLAN, Agilla y ActorNet por la cantidad de implementaciones que tienen y las prestaciones de servicios que ofrecen (ver sección 2.3.2).

   De los sistemas arriba mencionados incluyendo ambos (sistemas operativos y middleware), el sistema que más se apega a los lineamientos del estándar IEEE 802.15.4, es Tiny-OS. Este sistema operativo hace una implementación completa de la capa física, sin embargo hace modificaciones en la capa MAC y en lugar de utilizar la pila tal y como lo indica el estándar, implementa la pila BMAC.

   Por otra parte, existe MAC 802.15.4 que es una implementación que Freescale hace del estándar IEEE 802.15.4. Freescale ofrece dos herramientas de desarrollo, Beekit y Code-

Warrior que permiten al desarrollador generar plantillas y compilar programas respectivamente. El desarrollador puede configurar las plantillas al momento de generarlas con Beekit, además puede editarlas y agregarles código en CodeWarrior para lograr que el sistema se comporte de acuerdo a los requerimientos. CodeWarrior ofrece al desarrollador, un entorno de programación que permite compilación, depuración en tiempo de ejecución y almacenamiento de programa en la memoria flash del dispositivo.

MAC 802.15.4 de Freescale fue la herramienta seleccionada para este trabajo debido a que ofrece flexibilidad y potencia en los desarrollos, al mismo tiempo que garantiza estar bajo las convenciones del estándar IEEE 802.15.4.

3. El diseño de la arquitectura del sistema está basado en el modelo de capas OSI. Se construyó un conjunto de componentes que por su función, se pueden ubicar en las diferentes capas del modelo y que trabajan con el cometido de transportar información de manera confiable entre los dos sistemas que operan en la capa de aplicación. La comunicación entre los componentes es posible gracias a la definición de un paquete que además de la carga útil, cuenta con campos de control en la cabecera y fin del mismo. Se utilizaron dos lenguajes de programación, C# del lado del servidor que mantiene la base de datos y C del lado de la red de sensores. Aunque éste último es para programación estructurada, el sistema en su visión general, fue diseñado teniendo en mente el paradigma de programación orientado a objetos.

4. Se utilizó como se mencionó en el punto dos, la implementación MAC 802.15.4 para las primeras dos capas, se desarrolló el componente ARP_NS para la capa de red que trabaja sobre la capa MAC y mantiene el control del direccionamiento lógico entre los nodos sensores y los registros de pacientes en la base de datos, se construyó la capa de transporte integrada por diversos componentes que aseguran la entrega de paquetes de datos desde los nodos sensores hasta la base de datos y la entrega de paquetes de control desde el SMC_WEB hasta los nodos sensores, se implementó el IIS como sistema gestor de sesiones para el acceso a usuarios y finalmente se desarrollaron los sistemas de la capa de aplicación: la interfaz gráfica del lado del servidor para monitorización e interacción

con la base de datos y la red de sensores, y el sistema de adquisición y transmisión de datos del lado de la red de sensores para transmitir los signos vitales de los pacientes.

5. El prototipo de aplicación web fue construido en C# y permite la interacción de los usuarios con la base de datos para consultas, inserción, edición y eliminación de pacientes, nodos sensores y nodos coordinadores. Además, permite establecer control sobre parámetros que cambian el comportamiento de la red de sensores como son: el establecimiento de intervalos de muestreo para los signos vitales y la asignación de rangos en que los signos vitales se consideran estables para cada paciente. También permite la monitorización de los datos históricos recabados por los nodos, brindando servicios de filtrado por pacientes y fechas, la monitorización de los signos vitales en tiempo real y la monitorización de la tabla de alertas emitidas.

Cada punto anterior, describe la manera en que fueron cubiertas las diferentes etapas de la construcción del sistema y cómo se dio respuesta a los requerimientos que demandaban los objetivos específicos. Esto condujo a completar la realización del sistema de comunicación entre la red de sensores y el prototipo de sistema web para la monitorización de datos y control de parámetros de la red, lo cual da la posibilidad de decir que el objetivo general ha sido cubierto de manera satisfactoria.

5.1.2. De las preguntas de investigación

1. La primera pregunta está orientada a saber si es factible monitorizar a través de un sitio web, los signos vitales de pacientes tomados por dispositivos inalámbricos que forman una red basada en el estándar IEEE 802.15.4. El problema técnico consiste en la inexistencia de un protocolo que comunique este tipo de redes con las redes TCP/IP que son la base de la comunicación en Internet.

   Después de construir y probar el sistema, se puede responder que es factible establecer dicha comunicación, mediante la implementación de procesos que retoman los conceptos de mecanismos que originalmente fueron pensados para otro tipo de tecnología como

TCP/IP, DTN e IP-MOVIL, pero que se pueden adaptar tomando en cuenta las limitantes de energía, capacidad de memoria y capacidad de procesamiento de los dispositivos.

2. La segunda pregunta gira en torno a la confiabilidad que se puede garantizar en las transmisiones, respecto a que todos o un alto porcentaje de datos recabados de los pacientes, por los nodos sensores, lleguen íntegros a la base de datos para su monitorización en tiempo real o diferido.

   Con base en las pruebas realizadas y resultados obtenidos, se puede decir que el sistema es altamente efectivo en un entorno donde las exigencias de intervalos de tiempo para las muestras de signos vitales se realizan cada 180 segundos o más, con un máximo de 10 pacientes por nodo coordinador y llega a su límite de operatividad estable, cuando los signos vitales deben tomarse cada 20 segundos en un nodo coordinador que atiende cinco o más pacientes.

5.2. Trabajo Futuro

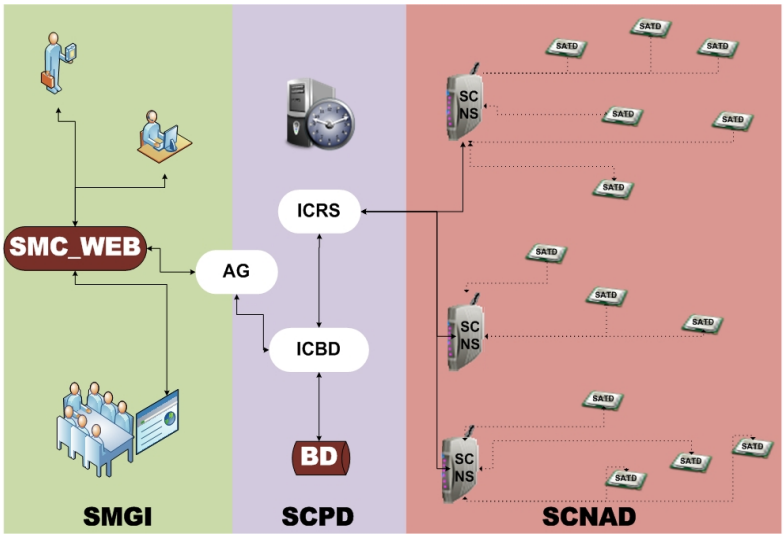
Con la construcción de este sistema, se ha probado la factibilidad de monitorizar pacientes de manera remota mediante una red de sensores inalámbricos y un sitio web. Se han dado con base en los resultados, las condiciones en que el sistema puede funcionar de manera óptima y se ha precisado el momento cuando el sistema se acerca a su límite de inoperatividad, al reducir los intervalos de muestreo de signos vitales y aumentar el número de nodos asociados a un coordinador. La aplicación centra su atención en los componentes que integran el protocolo de transporte, los cuales se encuentran de manera embebida en los nodos sensores y el Sistema de Comunicación y Persistencia a Datos del lado del servidor. En el nodo coordinador se desarrollaron componentes que administran la asociación de los nodos y mantienen la tabla de direccionamiento lógico y también se construyó el componente que interpreta los mensajes provenientes de ambos extremos del sistema y los retransmite a su destino final en el formato correspondiente. En lo que respecta a mecanismos que garanticen la confiabilidad en la entrega de información dentro del nodo coordinador, se configuró la transmisión de datos de mane-

ra indirecta, lo cual permite el almacenamiento de hasta cuatro paquetes para retransmitirlos posteriormente por el radio cuando en el primer intento se detectan colisiones o la red no está operando de manera correcta.

Se recomienda como trabajo futuro, construir un nodo coordinador que además de los componentes arriba mencionados, incluya los componentes que sean necesarios para garantizar la transmisión indirecta de paquetes del lado del puerto serie, lo cual incrementaría el nivel de eficiencia del sistema, permitiendo atender a más pacientes y hacer lectura de los signos vitales en intervalos más pequeños.

También se recomienda construir del lado del servidor, una interface TCP/IP análoga al componente ICRS que en este sistema presta servicios de lectura y escritura hacia la red de sensores. Esto permitirá utilizar un dispositivo más pequeño que una computadora como intermediario entre los nodos coordinadores y el servidor que mantiene la base de datos.

Finalmente también se deja como trabajo futuro la creación del sitio web para dispositivos móviles de tal manera que los servicios de monitorización y control estén disponibles en una mayor cantidad de equipos que por su tamaño y versatilidad pueden ser portados en todo momento y en cualquier lugar, lo cual es ilustrado en la figura 5.1, que muestra un esquema del sistema y sus diversos puntos de interacción.



Gráfica 5.1: Representación general del sistema.

Glosario

A

ACK (*ACKNOWLEDGEMENT*).
Acuse o reconocimiento. Por su uso en diversos sistemas de comunicación, formando parte de protocolos de capa de transporte y acceso al medio, este término tiene diferentes definiciones de las cuales se eligieron dos que se aplican adecuadamente en el contexto de este trabajo:

1. Señal de respuesta, que hace de acuse de recibo de que la comunicación se ha establecido con éxito.
2. Mensaje de control utilizado por un nodo de red dentro de un proceso de comunicación, para indicar a otro nodo que el paquete de datos ha sido recibido correctamente.

.

AES-128 (*Advanced Encryption Standard 128*).
Estándar de Encriptación Avanzado de 128 bits. Esquema de cifrado por bloques adoptado como estándar de cifrado por el gobierno de los Estados Unidos. AES es uno de los algoritmos más populares usados en criptografía simétrica.

AG (*Aplicación de Gestión*).
Componente responsable de confirmar la recepción exitosa de paquetes y aplicar los algoritmos de tiempos de espera y reenvío de paquetes. Presta servicios a la capa de aplicación.

ARP (*A*ddress Resolution Protocol).
Protocolo de Resolución de Direcciones. Es un protocolo de nivel de red responsable de encontrar la dirección hardware (Ethernet MAC) que corresponde a una determinada dirección IP.

ARP_NS (*Protocolo de Resolución de Direcciones para Nodos Sensores*).
Protocolo en la capa de red del sistema SMGI, responsable de encontrar la dirección de un nodo sensor que corresponde a una determinada etiqueta relacionada con un paciente en la base de datos.

B

BD (*B*ase de Datos).
Aunque existen varias definiciones para base de datos, se consideraron dos que parecen la más apropiadas para este trabajo.

1. Conjunto organizado e integrado de datos almacenados en computadora, con el fin de facilitar su uso para aplicaciones con múltiples finalidades.
2. Conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

.

C

CAP (*Contention Access Period*).
Periodo de Acceso por Contienda. Intervalo de tiempo en el que los nodos de una red basada en el estándar 802.15.4 con señalizadores, compiten entre ellos para acceder al medio de comunicación y transmitir información. Este intervalo de tiempo puede o no estar dividido en intervalos más pequeños llamados ranuras.

CCA (*C*lear Channel Assessment).
Valoración de Uso del Canal. Función lógica de capa física, la cual determina el estado actual de medio de comunicación inalámbrica.

CFP (*Contention Free Period*).
Periodo Libre de Contienda. Espacio de tiempo en el que los nodos de una red basada en el estándar IEEE 802.15.4 con señalizadores y GTS habilitado pueden transmitir de manera síncrona, sin competir con otros nodos por el acceso al medio de comunicación.

CPU (*Central Processing Unit*).
Unidad Central de Procesamiento. La unidad central de procesamiento y control en un sistema computacional controla la interpretación y ejecución de instrucciones. Contiene la unidad aritmética y lógica (ALU) que es la encargada de realizar los cálculos matemáticos, y la unidad de control que es la encargada de recibir las instrucciones de la memoria y aceptar los cálculos de la ALU y ejecutar las instrucciones.

CRC (*CyClic Redundancy Check*).
Comprobación de Redundancia Cíclica. Mecanismo usado comunmente para la detección de errores. Valida la integridad de un conjunto de datos contenidos en un paquete o trama mediante un proceso de muestra estadística y un polinomio matemático.

CSMA/CA (*Carrier Sense Multiple Access/Collision Avoidance*).
Múltiple Acceso por Detección de Portadora Evitando Colisiones. Protocolo de acceso al medio de capa MAC que incluye un esquema de prioridades para garantizar los

privilegios de transmisión de estaciones con mayor importancia. En redes inalámbricas como 802.11 y 802.15.4 se usa un mecanismo de reconocimiento positivo (ACK) antes de que una estación haga la transmisión para determininar la disponibilidad de medio y garantizar la no colisión.

CTS (*Clear To Send*).
Libre Para Transmitir. Es un mensaje enviado por una estación como respuesta de un mensaje (RTS) para indicar que el medio está libre para transmitir.

D

DIFS (*Distributed Inter-Frame Space*).
Espacio Inter-Frame Distribuido. Es un período de tiempo que una estación espera para transmitir después de haber detectado que el canal está libre. Por ejemplo en las redes basadas en los estándares IEEE 802.11 y 802.15.4, las estaciones utulizan una estrategia persistente con retardos aleatorios para detectar si el medio no está ocupado, cuando finalmente lo detectan libre, la estación espera por un periodo de tiempo (DIFS) y después envía el mensaje RTS y espera el mensaje CTS que garantice la transmisión sin colisiones.

DSSS (*Direct Sequence Spread Spectrum*).
Espectro Esparcido por Secuencia Directa. Es una técnica de modulación que utiliza un código de pseudorruido para modular directamente una portadora, de tal forma que aumente el ancho de banda de la transmisión y reduzca la densidad de potencia espectral (es decir, el nivel de potencia en cualquier frecuencia dada). La señal resultante tiene un espectro muy parecido al del ruido, de tal forma que a todos los radiorreceptores les parecerá ruido menos al que va dirigida la señal.

DTN (*Delay Tolerant Networks*).
Redes Tolerantes a Retardos. Enfoque para arquitectura de redes de computadoras que busca solucionar los problemas de transmisión en redes heterogéneas que carecen de continuidad en la conectividad. Ejemplo de estas redes son aquellas que operan bajo ambientes de mobilidad o con factores de interferencia extrema y redes interespaciales.

E

ED (*Energy Detection*).
Detección de Energía. Característica de funcionalidad de la capa física en la norma IEEE 802.15.4 que permite al dispositivo detectar la cantidad de energía en el canal inalámbrico.

EEPROM (*Erasable Programmable Read Only Memory*).
Memoria Programable y Borrable de Sólo Lectura. Memoria no volátil que puede ser programada, borrada y reprogramada eléctricamente.

F

FCC (*Federal Communication Commission*).
Comisión Federal de Comunicaciones. Es una agencia estatal independiente de Estados Unidos, bajo responsabilidad directa del Congreso. Junto con la Ley de Comunicaciones, es la encargada de la regulación (incluyendo censura) de telecomunicaciones interestatales e internacionales por radio, televisión, redes inalámbricas, satélite y cable. La FCC otorga licencias a las estaciones transmisoras de radio y televisión, asigna frecuencias de radio y vela por el cumplimiento de las reglas creadas para garantizar que las tarifas de los servicios por cable sean razonables.

FFD (*Full Function Device*).
Dispositivo con Funcionalidad Completa. Dispositivo que tiene capacidad física y lógica para fungir como nodo, enrutador o coordinador en una red de sensores inalámbrica basada la norma IEEE 802.15.4.

FHSS (*Frecuency-Hopping Spread Spectrum*).
Salto en Frecuencia de Espectro Ensanchado. Técnica de modulación empleada en comunicaciones de radio, la cual transmite ráfagas cortas de datos sobre un rango de canales de frecuencia dentro del ancho de banda de una portadora.

G

GTS (*Guaranteed Time Slot*).
Ranura de Tiempo Garantizada. Mecanismo que garantiza transmisiones síncronas entre un nodo y su nodo coordinador en redes de sensores inalámbricas con señalización basadas en el estándar IEEE 802.15.4.

I

ICBD (*Interface de Comunicación con la Base de Datos*).
Componente en el SMGI, responsable de la interacción con la base de datos. Provee todos los métodos de acceso para consulta, inserción, actualización y eliminación de datos.

ICRS (*Interface de Comunicación con la Red de Sensores*).
Componente en el SMGI, responsable de la comunicación entre el sistema de monitorización, base de datos y red de sensores, utiliza el puerto serie de la computadora como medio de transmisión y recepción.

IEEE (*Institute of Electrical and Electronics Engineers*).
Instituto de Ingenieros Electricos y Electronicos. Es la mayor asociación internacional sin ánimo de lucro formada por profesionales de las nuevas tecnologías, como ingenieros electricistas, ingenieros en electrónica, científicos de la computación, ingenieros en informática, ingenieros en biomédica, ingenieros en telecomunicación e ingenieros en mecatrónica. Trabaja con el objetivo de desarrollar estándares técnicos, prácticas recomendadas y guías para el desarrollo e implementación de tecnología en

ingeniería computacional, biomédica y aeroespacial, así como en las áreas de energía eléctrica, control, telecomunicaciones y electrónica de consumo, entre otras.

IFS (*InterFrame Spaces*).
Espacios entre Tramas. Período de tiempo que existe entre la transmisión de dos tramas consecutivas de la capa física a la capa MAC. Sirve para que después de recibida una trama, la capa MAC se prepare y pueda leer de manera correcta la segunda trama.

IIS (*Internet Information Service*).
Servicios de Información de Internet. Conjunto de servicios que convierten una computadora en un servidor de internet o intranet con las herramientas y funciones necesarias para administración del mismo.

IP (*Internet Protocol*).
Protocolo de Internet. En el contexto del modelo de referencia OSI, es un protocolo sin conexión de capa tres para el enrutamiento de datagramas a través de puentes que conectan redes y sub redes, sin garantía de entrega, secuencia y sin aplicar mecanismos de detección y correccción de errores. El protocolo IP procesa datagramas de IP de manera independiente definiendo su representación, ruta y envío.

IPC (*InterProcess Communication*).
Comunicación Entre Procesos. Mecanismo para que dos o más procesos (se usa mayormente para procesos que administra un sistema operativo) se puedan comunicar entre si y sincronizar sus acciones.

ISM (*Industrial Scientific and Medical*).
Industrial, Científica y Médica. Son bandas reservadas internacionalmente para uso no comercial de radiofrecuencia electromagnética en áreas industrial, científica y médica.

ISO (*International Organization for Standardization*).
Organización Internacíonal para la Estandarización. Organismo encargado de promover el desarrollo de normas internacionales para industrias, comercios y comunicaciones.

L

LQI (*Link Quality Indicator*).
Indicador de Calidad de Enlace. Mide la calidad del enlace en la transmisión de un paquete desde un nodo a su coordinador en una red de sensores inalámbrica basada en el estándar IEEE 802.15.4.

M

MCPS (*MAC Common Part Sublayer*).
Parte Común de la Subcapa MAC. Entidad encargada de proveer el servicio de datos entre la capa MAC y la aplicación desarrollada por un el usuario.

MCPS-SAP (*MAC Common Part Sublayer Service Access Point*).
Servicio de Acceso a la Parte Común de la Subcapa MAC. Interface responsable de proveer acceso al servicio de datos entre la aplicación desarrollada por usuario y la capa MAC.

MIMO (*Multiple Input-Multiple Output*).
Multiples Entradas-Múltiples Salidas. Se refiere específicamente a la forma como son manejadas las ondas de transmisión y recepción en antenas para dispositivos inalámbricos. MIMO aumenta la eficiencia espectral de un sistema de comunicación inalámbrica por medio de la utilización del dominio espacial.

MLME (*MAC Layer Managemment Entity*).
Capa MAC de Administración de Entidades. Provee el servicio de interfaces a través de las cuales, las funciones de administración de la capa pueden ser invocadas. También es responsable del mantenimiento de la PIB.

MLME-SAP (*MAC Layer Managemment Entity Service Acces Point*).
Punto de Acceso al Servicio de la Entidad de Administración de la Capa MAC. Interface responsable de proveer acceso para el servicio de administración de la capa MAC.

N

NC (*Nodo Coordinador*).
Dispositivo FFD, utilizado en la red de sensores del sistema SMGI para coordinar una celda de la red.

NS (*Nodo Sensor*).
Dispositivo RFD utilizado en la red de sensores del sistema SMGI, el cual tiene sensores acoplados para tomar las muestras de los signos vitales y transmitirlas a la base de datos a través de un NC.

O

OFMD (*Orthogonal Frequency Division Multiplexing*).
Multiplexación por División de Frecuencias Orthogonales. Consiste en enviar un conjunto de ondas portadoras de diferentes frecuencias, donde cada una transporta información, la cual es modulada en QAM o en PSK.

OSI (*Open System Interconnection*).
Modelo de Referencia de Interconexión de Sistemas Abiertos. Modelo de red Creado por la ISO para definir arquitecturas de interconexión de sistemas de comunicación. Proporciona una visión abstracta de como funciona las redes, desde el cableado que conecta las computadoras hasta los programas que se usan para la comunicación. Los componentes fundamentales del modelo OSI se constituyen en siete niveles, que son: físico, enlace de datos, red, transporte, sesión, presentación y aplicación.

P

P2P (*Peer to Peer*).
Par a Par. Red de computadoras en la que todos o algunos aspectos de ésta funcionan sin clientes ni servidores fijos.

PAN (*Personal Area Network*).
Red de Área Personal. Una red de comunicación de datos que conecta, entre sí o con otras computadoras, a los dispositivos que están próximos a una persona (por ejemplo, un PDA o un teléfono móvil).

paq_27 (*Paquete de datos 27*).
Paquete de datos transmitido desde el sistema de adquisición y transmisión de datos (SATD) en la red de sensores hacia el sistema de monitorización y control web (SMC_WEB).

paq_72 (*Paquete de datos 72*).
Paquete de datos transmitido desde el sistema de monitorización y control web (SMC_WEB) hacia el sistema de adquisición y transmisión de datos (SATD) en la red de sensores.

PD-SAP (*PHY Data Service Access Point*).
Punto de Servicios de Acceso de Datos de la Capa Física. Responsable de proveer el servicio de acceso a los datos de la capa física desde la capa MAC.

PI (*Inactive Period*).
Periodo Inactivo. Período de tiempo dentro de un SF donde los nodos no pueden transmitir.

PIB (*PAN Information Base*).
Base de Información de la PAN. Es la estructura de información donde se encuentran todos los atributos referentes a la configuración de la red de área personal. Tanto la capa física como la capa de control de acceso al medio cuentan con una PIB.

POS (*Personal Operation Space*).
Espacio de operación Personal. Región virtualmente esférica de aproximadamente 10 mts. de radio, al rededor de un dispositivo inalámbrico digital operado por una persona.

PPDU (*PHY Protocolo Data Unit*).
Unidad de Datos del Protocolo de la Capa Física. Contiene las cabeceras y la carga útil del paquete de datos de la capa física en las redes basadas en la norma IEEE 802.15.4.

Q

QoS (*Quality of Service*).
Calidad en el Servicio. Conjunto de tecnologías que garantizan la transmisión de cierta cantidad de información en un tiempo determinado. Los parámetros de QoS

incluyen prioridad en el acceso, disponibilidad de ancho de banda, latencia, retardos y pérdida de paquetes.

R

RF (*Radio Frecuency*).
Radiofrecuencia. Frecuencias del espectro electromagnético asociadas a ondas de radio tales como electricidad, luz, rayos X, rayos gama o rayos cósmicos.

RFC (*Request For Comments*).
Comentarios para discusión. Son memorandos publicados por el grupo Internet Engineering Task Force (IETF) para describir métodos, comportamientos, investigaciones o innovaciones aplicables al desarrollo de internet y sistemas interconectados. Después de revisión y discusión de los memorandos dentro de la comunidad de científicos del IETF, pueden llegar a convertirse en normas o estándares para el desarrollo de Internet.

RFD (*Reduce Function Device*).
Dispositivo con Capacidad Limitada. Dispositivo que tiene capacidad lógica limitada, toma el rol de un dispositivo final en una red de sensores inalámbrica basada la norma IEEE 802.15.4. No puede utilizarse como enrutador o coordinador.

RTS (*Request To Send*).
Solicitud Para Envío. Mensaje enviado por un nodo dentro de una red inalámbrica con el fin de confirmar la idoneidad del canal para realizar una transmisión. El nodo espera como respuesta al RTS un mensaje CTS.

S

SAP (*Service Access Point*).
Punto de Acceso al Servicio. Unidad conceptual en una arquitectura con base en el modelo OSI donde una capa puede solicitar los servicios de otra capa contigua.

SATD (*Sistema de Adquisición y Transmisión de Datos*).
Sistema embebido en los nodos sensores, es responsable entre otras cosas de tomar los signos vitales recabados por los sensores, verificar la existencia de alertas y armar el paquete paq_27 y enviarlo a su nodo coordinador para retransmisión a la base de datos.

SCNAD_paq (*Paquete de Datos del Sistema de Coordinación de Nodos y Adquisición de Datos*).
Con este nombre se denomina a los paquetes que se transmiten dentro de la red de sensores.

SCNS (*Sistema de Coordinación de Nodos Sensores*).
Sistema embebido en los nodos coordinadores con el fin de servir de pasarela de información, haciendo uso del mecanismo ARP_NS para la localización de direcciones y etiquetas de los nodos y registros destino respectivamente.

SCPD (*Sistema de Comunicación y Persistencia de Datos*).
Contiene todos los componentes que forman el protocolo de transporte dentro del SMGI.

SF (*SuperFrame*).
Super Frame. Espacio delimitado por dos señalizadores de principio y fin, en el cual se permite la transmisión de paquetes entre los dispositivos RFD y el dispositivo FFD en una red de sensores. Este período puede ser configurado con un periodo de acceso por contienda (CAP), otro periodo libre de contienda (CFP) y un tercer período inactivo (PI). Los dos últimos son opcionales.

SIFS (*Short InterFrame Space*).
Espacio Corto Entre Frames. IFS más pequeños que se pueden tener en el envío de tramas.

SMC_WEB (*Sistema de Monitorización y Control Web*).
Sistema de usuario final que contiene las pantallas de interacción para monitorizar los datos de los pacientes recabados por la red de sensores y para establecer ciertos parámetros como intervalos de muestreo y rangos de signos vitales que dan control sobre el comportamiento de los nodos sensores.

SMGI (*Sistema de Monitorización y Gestión de la Información*).
Nombre que se da al sistema de comunicación completo. Incluye el sistema de monitorización y control web, el servidor con las interfaces de comunicación haci la base de datos y la red de sensores, la base de datos y las aplicaciones en los dispositivos de la red de sensores.

SPBD (*Procedimientos Almacenados de Base de Datos*).
Son funciones programadas dentro de la base de datos que permiten ejecutar acciones como consultas, actualizaciones, inserción y eliminación de registros mediante su invocación desde un lenguaje de programación. El código es precompilado por lo que su uso reduce el tiempo de las transacciones con la base de datos.

SQL (*Structured Query Language*).
Lenguaje Estructurado de Consultas. Es un lenguaje formal declarativo, estandarizado ISO, para manipular información en la base de datos.

SVI (*Suma de Verificación de Integridad*).
Algoritmo que se aplica dentro del componente ICRS para verificar que los datos hayan llegado de manera correcta.

T

TCP (*Transmision Control Protocol*).
Protocolo de Control de Transmisión. Protocolo del conjunto de protocolos TCP/IP que trabaja en la capa de transporte. Es un protocolo orientado a conexión diseñado para proveer transmisiones confiables en redes tales como Internet que utilizan protocolos poco fiables como IP.

TCP/IP (*Transmision Control Protocol/Internet Protocol*).
Protocolo de Control de Transmisión/Protocolo de Internet. Es un conjunto de protocolos que proporcionan las bases de todo lo que ha llegado a ser Internet y por tanto a menudo se le referencía como el Conjunto de protocolos de Internet. Este conjunto de protocolos se ha organizado en cinco capas que se asemejan a las del modelo de OSI. Las capas TCP/IP son: Física, Enlace de Datos, Red, Transporte y Aplicación. Dentro de el conjunto de protocolos TCP/IP encontramos los protocolos FTP, SMTP, SNMP, TELNET, TCP UDP, ARP, ICMP, IP, entre otros.

U

UART (*Universal Asynchronous Receiver-Transmiter*).
Transmisor-Receptor Asíncrono Universal. Controla los puertos y dispositivos serie. se encuentra integrado en la placa base o en la tarjeta adaptadora del dispositivo.

UDP (*User Datagram Protocol*).
Protocolo de Datagrama de Usuario. Protocolo de capa de transporte basado en el intercambio de datagramas. Permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento en su cabecera.

URL (*Uniform Resource Locator*).
Localizador Uniforme de Recursos. Secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos en Internet para su localización o identificación, como por ejemplo documentos textuales, imágenes, videos, presentaciones digitales, etc.

V

VD_SMGI (*Ventana Deslizante SMGI*).
Algoritmo que se aplica para el reenvío de paquetes del SATD al SMC_WEB cuando el tiempo de espera inicial se agotó sin recibir como respuesta de éxito, el mensaje de confirmación correspondiente.

W

WAP (*Wireless Access Point*).
Punto de Acceso Inalámbrico. Dispositivo que interconecta equipos de comunicación inalámbrica para formar una red. Normalmente este dispositivo también se puede conectar a una red cableada y permitir la transmisión de información entre ambas redes.

WPAN (*Wireless Personal Area Network*).
Red Inalámbrica de Área Personal. Red formada por dispositivos que utilizan un canal inalámbrico para comunicarse dentro de un área de algunas decenas de metros.

WSN (*Wireless Sensor Networks*).
Redes de Sensores Inalámbrica. Red formada por dispositivos autónomos distribuidos dentro de un área determinada que ayudan a persibir el mundo físico mediante sensores y se comunican mediante un canal inalámbrico.

Bibliografía

[1] H. Abrach, S. Bhatti, J. Carlson, H. Dai, J. Rose, A. Sheth, B. Shucker, J. Deng, and R. Han. Mantis: system support for multimodal networks of in-situ sensors. In *WSNA '03: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 50–59, New York, NY, USA, 2003. ACM.

[2] Ian F. Akyildiz, Xudong Wang, and Weilin Wang. Wireless mesh networks: a survey. *Computer Networks*, 47(4):445–487, March 2005.

[3] WiFi Aliance. Wi-fi. In *Knowledge Center. Disponible en:*. www.wi-fi.org, Accedido en Junio de 2010.

[4] Agustín A. Araya. Questioning ubiquitous computing. In *In Proceedings of the 1995 ACM 23rd annual conference on Computer science*, pages 230–237. ACM Press, 1995.

[5] Shah Bhatti, James Carlson, Hui Dai, Jing Deng, Jeff Rose, Anmol Sheth, Brian Shucker, Charles Gruenwald, Adam Torgerson, and Richard Han. Mantis os: an embedded multithreaded operating system for wireless micro sensor platforms. *Mob. Netw. Appl.*, 10(4):563–579, 2005.

[6] Athanassios Boulis, Chih-Chieh Han, and Mani B. Srivastava. Design and implementation of a framework for efficient and programmable sensor networks. In *MobiSys '03: Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 187–200, New York, NY, USA, 2003. ACM.

[7] Fco. Javier Ceballos. *Enciclopedia de Microsoft Visual C Sharp*. Alfaomega Ra-Ma, Madrid, 2006.

[8] Matteo Ceriotti, Luca Mottola, Gian Pietro Picco, Amy L. Murphy, Stefan Guna, Michele Corra, Matteo Pozzi, Daniele Zonta, and Paolo Zanon. Monitoring heritage buildings with wireless sensor networks: The torre aquila deployment. In *IPSN '09: Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*, pages 277–288, Washington, DC, USA, 2009. IEEE Computer Society.

[9] Adam Dunkels and Juan Alonso. Connecting Wireless Sensornets with TCP/IP Networks. pages 153—-152, 2004.

[10] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *LCN '04: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, pages 455–462, Washington, DC, USA, 2004. IEEE Computer Society.

[11] Chien-Liang Fok, Gruia-Catalin Roman, and Chenyang Lu. Rapid development and flexible deployment of adaptive wireless sensor network applications. In *ICDCS '05: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, pages 653–662, Washington, DC, USA, 2005. IEEE Computer Society.

[12] Sophia Chung Fegan Forouzan, Behrouz A. *Data Communications and NetWorking.* McGraw-Hill, fourth edition edition, 2007.

[13] Tia Gao, C. Pesto, L. Selavo, Yin Chen, Jeong G. Ko, Jong H. Lim, A. Terzis, A. Watt, J. Jeng, Bor-Rong Chen, K. Lorincz, and M. Welsh. Wireless medical sensor networks in emergency response: Implementation and pilot results. In *Technologies for Homeland Security, 2008 IEEE Conference on*, pages 187–192, 2008.

[14] Chih-Chieh Han, Ram Kumar, Roy Shea, Eddie Kohler, and Mani Srivastava. A dynamic operating system for sensor nodes. In *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 163–176, New York, NY, USA, 2005. ACM.

[15] Wendi Beth Heinzelman, Amy L. Murphy, Hervaldo S. Carvalho, and Mark A. Perillo. Middleware to support sensor network applications. *IEEE Network*, 18(1):6–14, 2004.

[16] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for networked sensors. *SIGPLAN Not.*, 35(11):93–104, 2000.

[17] IEEE-TG15.4. *Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs).* IEEE standard for Information Technology, 2003.

[18] Chaiporn Jaikaeo, Chavalit Srisathapornphat, and Chien-Chung Shen. Querying and tasking in sensor networks. In Raja Suresh and Homer H. Pien, editors, -, volume 4037-1, pages 184–194. SPIE, 2000.

[19] Mauri Kuorilehto, Marko Hännikäinen, and Timo D. Hämäläinen. A survey of application distribution in wireless sensor networks. *EURASIP J. Wirel. Commun. Netw.*, 2005(5):774–788, 2005.

[20] YoungMin Kwon, Sameer Sundresh, Kirill Mechitov, and Gul Agha. Actornet: an actor platform for wireless sensor networks. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1297–1300, New York, NY, USA, 2006. ACM.

[21] Winnie L. Lee, Amitava Datta, and Rachel Cardell-Oliver. *Network Management in Wireless Sensor Networks in book Mobile Ad Hoc and Pervasive Communications*, chapter 2. School of Computer Science and Software Engineering The University of Western Australia, 35 Stirling Highway Crawley WA 6009 Australia, 2006.

[22] Dong Li, Wei Liu, Ze Zhao, and Li Cui. Demonstration of a wsn application in relic protection and an optimized system deployment tool. In *IPSN '08: Proceedings of the 7th international conference on Information processing in sensor networks*, pages 541–542, Washington, DC, USA, 2008. IEEE Computer Society.

[23] L A N Man, Standards Committee, and Ieee Computer. IEEE Standard for Information technology- Telecommunications and information exchange between systems- Local and metropolitan area networks- Specific requirements–Part 15.4: Wireless MAC and PHY Specifications for Low-Rate WPANs. *Control*, 2006(September), 2006.

[24] John Ray. *Edición Especial TCP/IP*. Prentice Hall, Madrid, 1999.

[25] Wade Rush. 10 emerging technologies that will change the world. *Tecnology Review*, pages 33–56, Febrero 2003.

[26] SIG. Bluetooth. In *Aspectos técnicos. Disponible en:*. www.bluetooth.com, Accedido en Junio de 2010.

[27] K. Sohraby and M. Daneshmand. A survey of transport protocols for wireless sensor networks. *IEEE Network*, 20(3):34–40, Mayo 2006.

[28] Shunichiro Suenaga and Shinichi Honiden. Name-based location service for mobile agents in wireless sensor networks. In *MOBILWARE '08: Proceedings of the 1st international conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications*, pages 1–6, ICST, Brussels, Belgium, Belgium, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[29] Sing-Yiu Cheung Varaiya and Pravin. Traffic surveillance by wireless sensor networks: Final report for path to 5301. Technical report, Berkeley, 2007.

Anexo I (Análisis y diseño del sistema)

Este documento incluye el detalle del análisis que se realizó para desarrollar tres de los procesos más importantes del sistema. Presenta la descripción de los requerimientos, los casos de uso, los diagramas de clases, los diagramas de secuencia y el diseño detallado de los procesos. Además se incluye el diseño detallado de los componentes que integran el Sistema de Comunicación y Persistencia a Datos (SCPD), que es la parte central del trabajo de tesis, y finalmente, se agrega el diagrama de la base de datos y la descripción de las tablas y campos de la misma. El volumen de páginas que contiene la documentación, no hace posible incluirla de manera completa de forma impresa, sin embargo el lector la puede consultar dentro del CD que se encuentra en la contraportada de este documento.

Contenido

Requerimientos

Maestría en Ingeniería

Universidad Autónoma de Baja California

Gestión de pacientes

Nombre	Establecer rangos a los signos vitales.
Fecha	**6 de Octubre de 2009**
Descripción	El usuario podrá especificar el valor en los que cada signo vital de un paciente se considera dentro de un rango normal. Para comodidad del usuario podrá elegir un grupo de pacientes o aplicarlo a un solo paciente.
Precondiciones	• Ninguna
Importancia	Alta

Gestión de la Monitorización

Nombre	Activar-desactivar la monitorización.
Fecha	**6 de Octubre de 2009**
Descripción	Se presentará una pantalla donde el usuario podrá ver todos los parámetros establecidos para el paciente. El usuario puede: • Activar la monitorización, si no está iniciada y está de acuerdo con los datos mostrados. • Desactivar la monitorización, si ya no desea seguir recibiendo datos del paciente.
Precondiciones	• Los siguientes datos deben estar capturados para dar la posibilidad de activar la monitorización: o La asociación del paciente con el nodo. o Los límites inferior y superior para cada signo vital que se monitorizará. o Los tiempos de muestreo para cada signo vital a monitorizar
Importancia	Alta

Nombre	Lectura de signos vitales de pacientes en tiempo real
Fecha	**6 de Octubre de 2009**
Descripción	El usuario tendrá la posibilidad de solicitar a la red la medición de una o más veces de cualquier signo vital que le interese, ya sea de un paciente o de un grupo de los mismos.
Precondiciones	• Ninguna
Importancia	Alta

Análisis y Diseño

Maestría en Ingeniería

Universidad Autónoma de Baja California

Establecer rangos de signos vitales

Casos de uso

Componente:	**Sistema de Monitorización y control web**	
Fecha:	14 de Enero de 2010	
Descripción:	El usuario podrá especificar el valor en los que cada signo vital de un paciente se considera dentro de un rango normal. Para comodidad del usuario podrá elegir un grupo de pacientes o aplicarlo a un solo paciente.	
Actores:	Usuarios finales (UF), SCNS	
Precondiciones:		
Flujo normal:	**Acción del Actor** 1.- (UF) Abre la pantalla de gestión de rangos de signos vitales. 3. (UF) Selecciona los pacientes a los que desea establecer los parámetros. Puede ser uno, varios o una clasificación, Captura la información, hace clic en el botón **GRABAR**. 6. (SCNS) Envía Mensaje de confirmación de rango de signo vital establecido. 8.- Hace clic en el botón **SALIR.**	**Respuesta del Sistema** 2. Despliega los pacientes por clasificación. 4.- Verifica que los datos estén capturados correctamente. 5.- Manda la información a **SCNS.** 7.- Graba la información a la base de datos. 9.- Cierra la ventana de **GESTION DE SIGNOS VITALES**.
Flujo alternativo:	Línea 4: Si los datos no han sido capturados correctamente. 4.1 Regresa el mensaje de datos incorrectos, ir al paso 3. Línea 5: Si no se puede enviar información a **SCNS**. 5.1 Regresa el mensaje de correspondiente, ir al paso 3. Línea 7: Si la información no se puede registrar en la BD: 7.1 Regresa el mensaje que indica que la base de datos no puede ser accedida, ir al paso 4.	
Postcondiciones:	Los rangos de los signos vitales de los pacientes seleccionados han sido modificados en la base de datos.	

Componente:	**Sistema coordinador de nodos sensores**	
Fecha:	14 de enero de 2010	
Descripción:	Atender un mensaje **MSC** de tipo **ERSV** (Establecer rangos a signos vitales), Enviar el mensaje al **NS** implicado. Atender **MC** (Mensaje de Confirmación) **RSVE** (Rangos de signos vitales establecidos) y enviar la respuesta al **SCPD** (Sistema de Comunicación y Persistencia a Datos)**.**	
Actores:	SCPD (Sistema de Comunicación y Persistencia a Datos), NS.	
Precondiciones:	Haber dado de alta los nodos al sistema y a la red.	
Flujo normal:	**Acción del Actor** 1. El **SCPD** envía mensaje **MSC_ ERSV**. 4.- El **NS** envía un mensaje **MC_ RSVE.**	**Respuesta del Sistema** 2. Verifica si el nodo está asociado a él. 3.-Envía al **NS**, el **MSC_ ERSV**. 5.- Recibe **MC_ RSVE**, del **NS** 6.- Envía al **SCPD** el **MC_ RSVE.**
Flujo alternativo:	Línea 2: El nodo no está asociado a él:	

	2.1 Envía **E_NSNE** al **SCPD**. Línea 5: Recibe **E_ESVNR** o no recibe nada en 2 segundos**:** 5.1 Envía **E_ESVNR** al **SCPD**.
Pos condiciones:	
Comentarios:	Es llamado por el caso de uso de SMGI CUP5 (Establecer rangos a signos vitales)

Componente:	**Sistema de adquisición y transmisión de datos**	
Fecha:	14 de enero de 2010	
Descripción:	Atender un mensaje **MSC_ERSV**, realizar los cambios correspondientes y regresar el mensaje **MC_RSVE**, al **SCNS**.	
Actores:	SCNS (Sistema de Coordinación de Nodos Sensores)	
Precondiciones:		
Flujo normal:	**Acción del Actor** 1.- Envía mensaje **MSC_ERSV**.	**Respuesta del Sistema** 2.- Lee los datos para la actualización de los parámetros y actualiza uno por uno**.** 3.- Envía **MC_RSVE**, al **SCNC**.
Flujo alternativo:	Línea 3: No se llevó a cabo correctamente la actualización de los rangos de los signos vitales: 3.1.- Envía **E_ESVNR**.	
Pos condiciones:	Los rangos de los signos vitales para el nodo han sido actualizados	
Comentarios:	**Es llamado desde el caso de uso de SCNS CUP5 (Establecer rangos a signos vitales)**	

Diagrama de clases: **Establecer rangos a signos vitales**

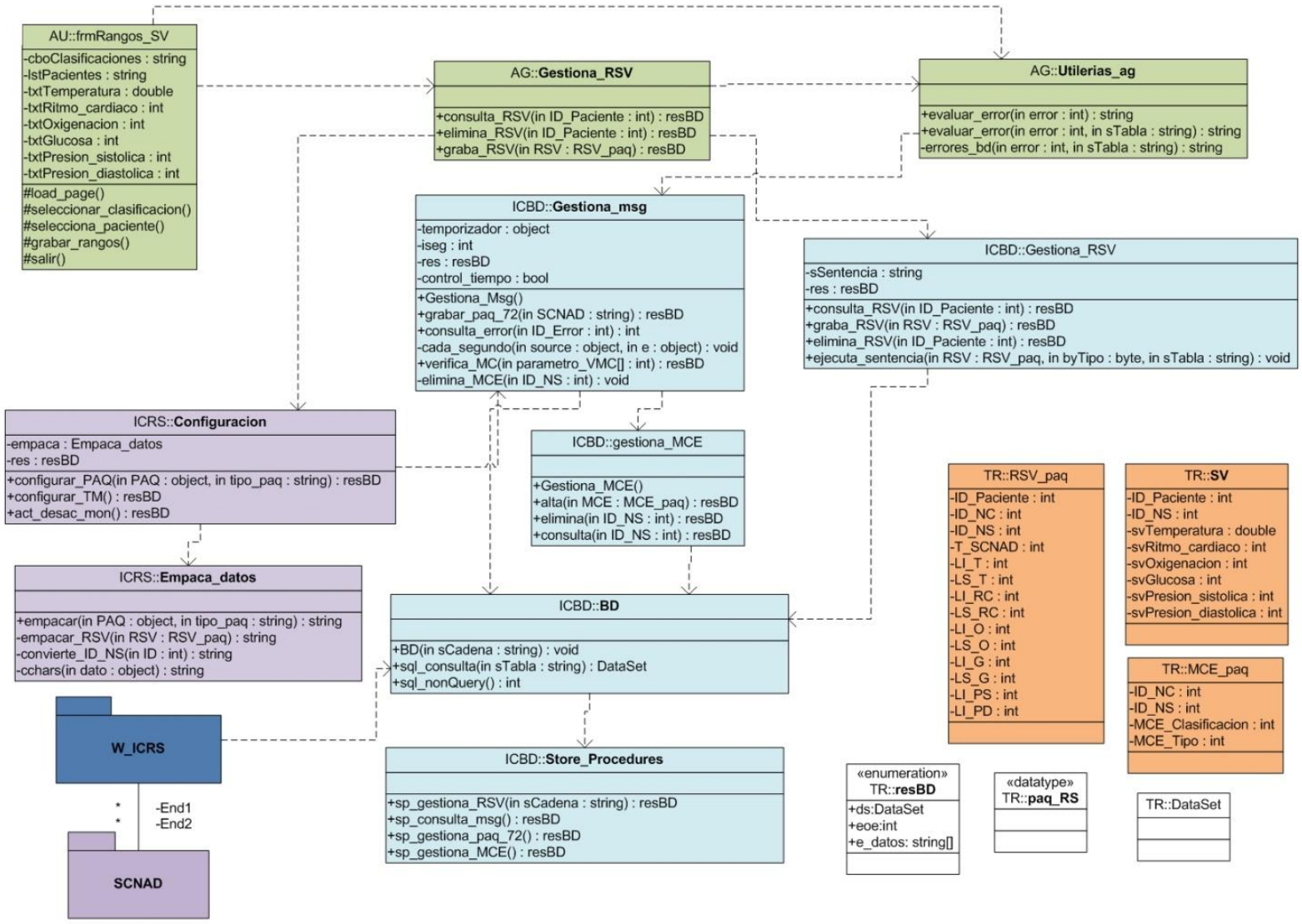
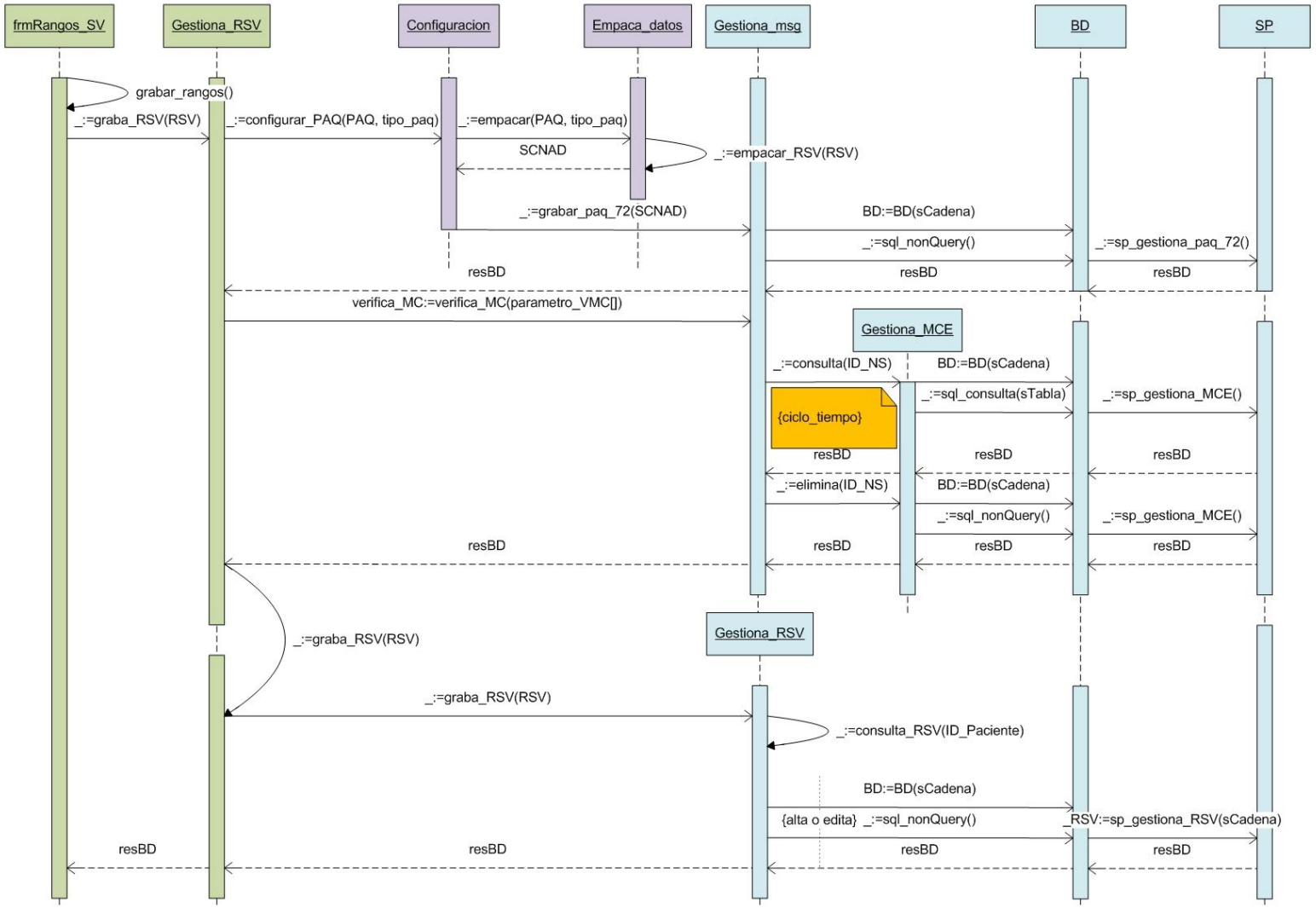


Diagrama de secuencia: **Establecer rangos a signos vitales**



Diseño detallado

Nombre de la clase	**AU:frmRangos_SV.aspx.cs**
Nombre del método	grabar_rangos
Tipo de método	Sub
Tipo de dato que regresa	Nada
Argumentos de entrada	Nada
Mensajes de error	El nodo no responde, no se pudieron establecer los rangos, la base de datos no puede ser accedida o error de BD.
Clases que utiliza	Gestiona_RSV.cs
Visibilidad	Protected
Métodos llamados	AG: Gestiona_RSV:graba_RSV() Argumentos: RSV:RSV_paq AG: Utilerias_ag:evaluar_error() Argumentos: error:int
Descripción	1. Verifica que los datos han sido capturados de manera correcta. 2. Ejecuta un ciclo y con cada ítem marcado de la lista de pacientes: a. Carga los datos capturados al paquete RSV b. Manda el mensaje al método **graba_RSV** de la clase **Gestiona_RSV**, para proseguir con la edición de datos en caso de que ya existan o el alta de los mimos si es la primera vez que se ejecuta el proceso para el paciente. 3. Verifica si el proceso se ejecutó correctamente tomando como referencia el valor de **eoe** que debe ser **111** en caso de éxito: a. Éxito: Manda el mensaje de operación realizada. b. Error: invoca el método **evaluar_error** de la clase **Utilerias_ag** (detallada en el documento **DD_Utilerias_ag**) para determinar el tipo de erro que se sucitó.

Nombre de la clase	**AG:Gestiona_RSV.cs**
Nombre del método	graba_RSV
Tipo de método	Función
Tipo de dato que regresa	resBD
Argumentos de entrada	RSV
Mensajes de error	Nada
Clases que utiliza	ICBD:Gestiona_Nodos:consulta_NS, ICRS:Configuracion
Visibilidad	Public
Métodos llamados	ICBD:Gestiona_Nodos:consulta_NS Argumentos ID_Paciente:int ICRS: Configuracion:configurar_PAQ Argumentos: RSV:RSV_pac Tipo_paq:string ICBD: Gestiona_Msg:verifica_MC Argumentos: Msg_esperado:int

	ID_NS:int Tiempo_espera:int ICBD: Gestiona_RSV:graba_RSV Argumentos: RSV:RSV_pac
Descripción	1. Asigna **MC_RSVE** a **msg_esperado** 2. Hace una consulta a la base de datos para tomar el **ID_NS** y el **ID_NC** al que está asociado el paciente. 3. Asigna a los atributos correspondientes el **ID_NS** y el **ID_NC.** 4. Manda llamar el método **configurar_PAQ** de la clase **ICRS:Configuracion** (detallado en el documento **DDPA_configurar_PAQ,** por formar un flujo completo y general a varios casos de uso) enviando el paquete **RSV** para hacer el proceso de establecer los rangos de signos vitales en el nodo indicado y la palabra "**RSV**", para identificación del tipo de paquete que se trata. 5. Verifica que el paquete se pudo generar y pudo ser enviado a la cola de paquetes **paq_72**: a. **Si**: obtiene el **ID** del mensaje que fue agregado a la cola de mensajes en la base de datos, este dato fue guardado en la posición 0 del arreglo **e_datos** al momento de enviar el mensaje a la base de datos. b. Genera el arreglo **parámetro_VMC**, donde almacenará el mensaje esperado (**msg_esperado**), el Identificador del Nodo Sensor (**ID_NS**), el tiempo de espera para el mensaje esperado (**15 segundos**) y el Identificador del paquete que se enviará (**ID_paq_72**) que está en la cola de espera **paq_72**. c. Verifica si llego el mensaje de confirmación invocando el método **verifica_MC()** de la clase **Gestiona_Msg()** detallada en el documento **DD_verifica_MC**. i. éxito: graba el **RSV** a la base de datos mediante el método **graba_RSV()** de la clase **ICBD:Gestiona_RSV.** ii. Error: nada, pasa a la siguiente instrucción. 6. Regresa a la aplicación de usuario.

Nombre de la clase	**ICBD:Gestiona_RSV.cs**
Nombre del método	graba()
Tipo de método	Función
Tipo de dato que regresa	resBD
Argumentos de entrada	RSV:RSV_paq
Mensajes de error	Nada
Clases que utiliza	ICBD: Gestiona_RSV, ICBD:BD
Visibilidad	Public
Métodos llamados	ICBD: Gestiona_RSV:consulta Argumentos: ID_Paciente:int ICBD: Gestiona_RSV:ejecutaSentencia Argumentos: PAC:Pac_paq byTipo:byte sTabla:string
Descripción	1. Declara la variable **sSentencia** que determina el tipo de sentencia que se va a

	ejecutar, se supone de inicio que será una actualización (3). 2. Declara la variable que contendrá la sentencia poniendo el nombre del **store procedure** (cabe aclarar que en los otros métodos este paso no es necesario porque ya en el constructor se puso el nombre del store procedure, sin embargo aquí se hace uso del método consulta que a su vez utiliza el método ejecuta sentencia en el cual se pone vacía la variable **sSentencia**) 3. Invoca el método **consulta** de la misma clase, para saber si existe un registro de **RSV**, para el paciente. a. **NO:** asigna 2 a la variable tipo_sentencia para hacer una inserción de datos. 4. invoca la función **ejecutaSentencia,** esta función se utiliza en las cuatro transacciones básicas (SELECT, INSERT, UPDATE y DELETE) y está detallada en el documento **DD_ejecutaSentencia_RSV**. Los parámetros que se pasan son: a. **RSV** que el paquete que lleva los datos de los rangos de signos vitales para el paciente. b. **byTipo** que especifica el tipo de sentencia que se va a ejecutar (en este caso 2 o 3). c. **sTabla** que es el nombre que tendrá la tabla en el **DataSet** que se tiene como respuesta en el caso de ser necesario. 5. Regresa **res** al método de donde fue invocado.

Nombre de la clase	**ICBD:Gestiona_RSV.cs**
Nombre del método	consulta()
Tipo de método	Función
Tipo de dato que regresa	resBD
Argumentos de entrada	ID_Paciente
Mensajes de error	Nada
Clases que utiliza	ICBD: Gestiona_RSV
Visibilidad	Public
Métodos llamados	ICBD: Gestiona_RSV:ejecutaSentencia Argumentos: PAC:Pac_paq byTipo:byte sTabla:string
Descripción	1. Crea la instancia del paquete **RSV** para utilizarlo como parámetro en el método **ejecutaSentencia**. 2. Asigna el ID del paciente al atributo correspondiente del paquete **RSV**. 3. invoca la función **ejecutaSentencia,** esta función se utiliza en las cuatro transacciones básicas (SELECT, INSERT, UPDATE y DELETE) y está detallada en el documento **DD_ejecutaSentencia_RSV**. Los parámetros que se pasan son: a. **RSV** que el paquete que lleva el atributo ID del paciente. b. **byTipo** que especifica el tipo de sentencia que se va a ejecutar (en este caso 1). c. **sTabla** que es el nombre que tendrá la tabla en el **DataSet** que se tiene como respuesta para la lectura de los datos. 4. Regresa **res** al método de donde fue invocado.

Activar desactivar monitorización

Casos de uso

Componente:	**Sistema de monitorización y control web**	
Fecha:	22 de Febrero de 2010	
Descripción:	Se presentará una pantalla donde el usuario podrá ver todos los parámetros establecidos para el paciente. El usuario puede: • Activar la monitorización, si no está iniciada y está de acuerdo con los datos mostrados. • Desactivar la monitorización, si ya no desea seguir recibiendo datos del paciente.	
Actores:	Usuarios finales (UF), SCNS	
Precondiciones:		
Flujo normal:	**Acción del Actor** 1.- (UF) Abre la pantalla para la activación-desactivación de monitorización. 3. (UF) Selecciona el paciente al que desea activar-desactivar monitorización. Hace clic en el botón **ACEPTAR**. 5. (SCNS) Envía Mensaje de confirmación de activación-desactivación realizada. 9.- Hace clic en el botón **SALIR.**	**Respuesta del Sistema** 2. Despliega los pacientes por clasificación, despliega los rangos en los que los signos vitales se consideran normales y los intervalos de muestreo para cada signo vital. 4.- Manda la información al **SCNS.** 6.- verifica la confirmación desde el **SCNS.** 7.- Graba la información a la base de datos. 8.- Muestra la lista de pacientes modificados. 10.- Cierra la ventana de **ACTIVACION-DESACTIVACION**.
Flujo alternativo:	Línea 5: Si no se puede enviar información al **SCNS**. 5.1 Regresa el mensaje correspondiente, ir al paso 3 o ir al paso 9. Línea 5: Si no se recibió la confirmación desde el **SCNS**. 5.1 Regresa el mensaje correspondiente, ir al paso 3 o ir al paso 9. Línea 7: Si la información no se puede registrar en la BD: 7.1 Regresa el mensaje que indica que la base de datos no puede ser accedida, ir al paso 4.	
Pos condiciones:	Los pacientes quedan en estatus de activado-desactivado si el estatus es activado, los dispositivos toma las muestras en los intervalos establecidos.	

Componente:	**Sistema coordinador de nodos sensores**	
Fecha:	23 de febrero de 2010	
Descripción:	Atender un mensaje **MSC** de tipo **ADM** (Activar-Desactivar Monitorización), Enviar el mensaje al **NS** implicado. Atender **MC** (Mensaje de Confirmación) **ADMR** (Activar-Desactivar Monitorización Realizada) y enviar la respuesta al **SCPD** (Sistema de Comunicación y Persistencia a Datos)**.**	
Actores:	SCPD (Sistema de Comunicación y Persistencia a Datos), NS.	
Precondiciones:	Tener la asociación del nodo realizada.	
Flujo normal:	**Acción del Actor** 1. El **SCPD** envía mensaje **MSC_ ADM**. 4.- El **NS** envía un mensaje **MC_ ADMR.**	**Respuesta del Sistema** 2. Verifica si el nodo está asociado a él. 3.-Envía al **NS**, el **MSC_ ADM**. 5.- Recibe **MC_ ADMR**, del **NS** 6.- Envía al **SCPD** el **MC_ ADMR.**

Flujo alternativo:	Línea 2: El nodo no está asociado a él: 2.1 Envía **E_ADMNR** al **SCPD**. Línea 5: Recibe **E_ADMNR** o no recibe nada en x segundos**:** 5.1 Envía **E_ADMNR** al **SCPD**.
Pos condiciones:	

ID:	**Sistema de adquisición y transmisión de datos**	
Fecha:	14 de enero de 2010	
Descripción:	Atender un mensaje **MSC_ADM**, realizar los cambios correspondientes y regresar el mensaje **MC_ADMR**, al **SCNS**.	
Actores:	SCNS (Sistema de Coordinación de Nodos Sensores)	
Precondiciones:		
Flujo normal:	**Acción del Actor** 1.- Envía mensaje **MSC_ADM**.	**Respuesta del Sistema** 2.- Lee el dato de activación-desactivación de la monitorización, y lo asigna a la bandera que controla la toma de la muestra y el envío de los datos adquiridos al SCNS**.** 3.- Envía **MC_ADMR**, al **SCNC**.
Flujo alternativo:	Línea 3: No se llevó a cabo correctamente la actualización de los rangos de los signos vitales: 3.1.- Envía **E_ADMNR**.	
Pos condiciones:	Si el dato fue 0 el NS dejará de adquirir la información y por consiguiente no la enviará al SCNS. Si el dato fue 1 el NS empezará a adquirir la información de los sensores y a enviarla al SCNS cuando se cumpla el intervalo de tiempo establecido en las variables destinadas a este propósito.	

Diagrama de clases: **Gestión de la monitorización**

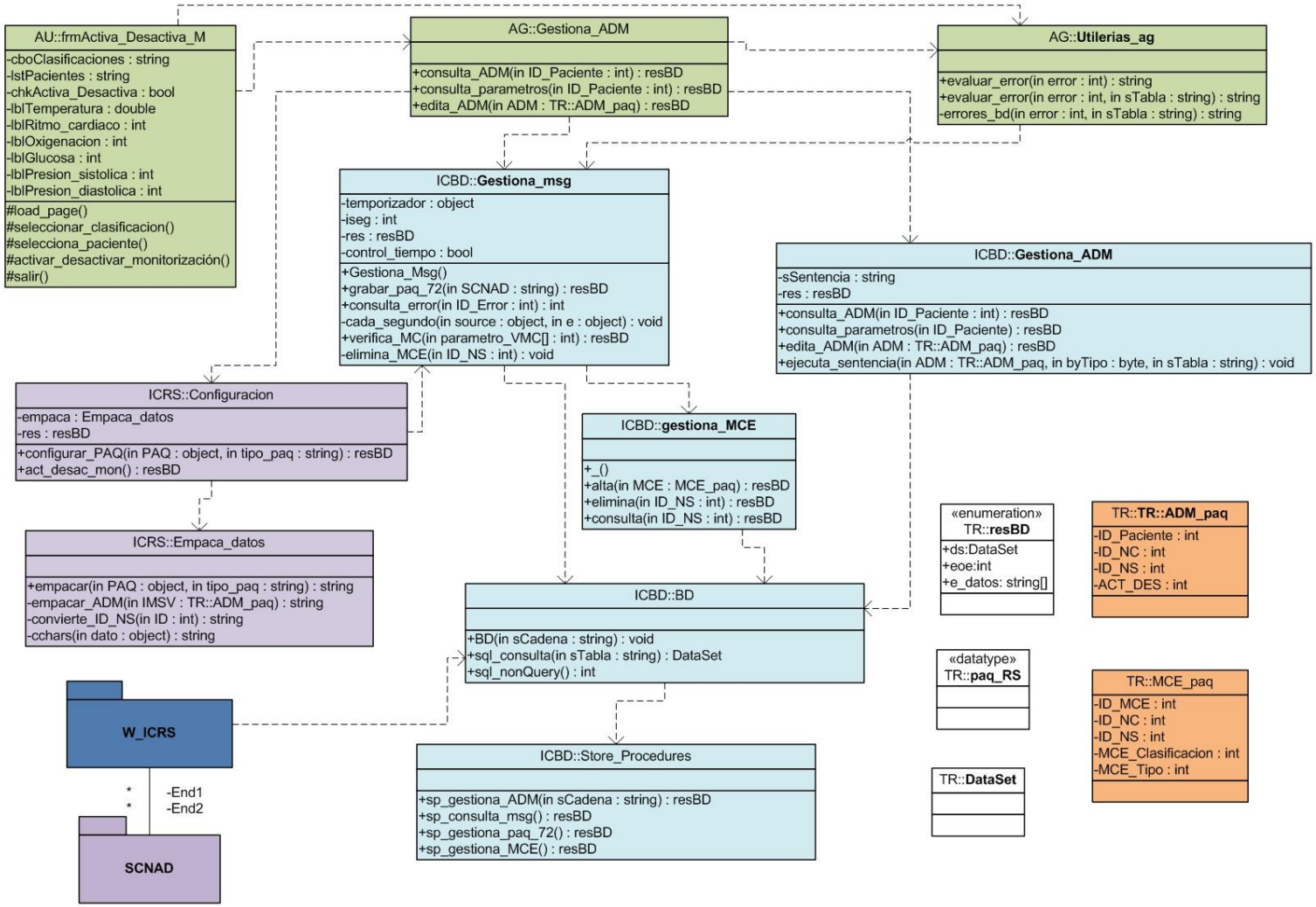
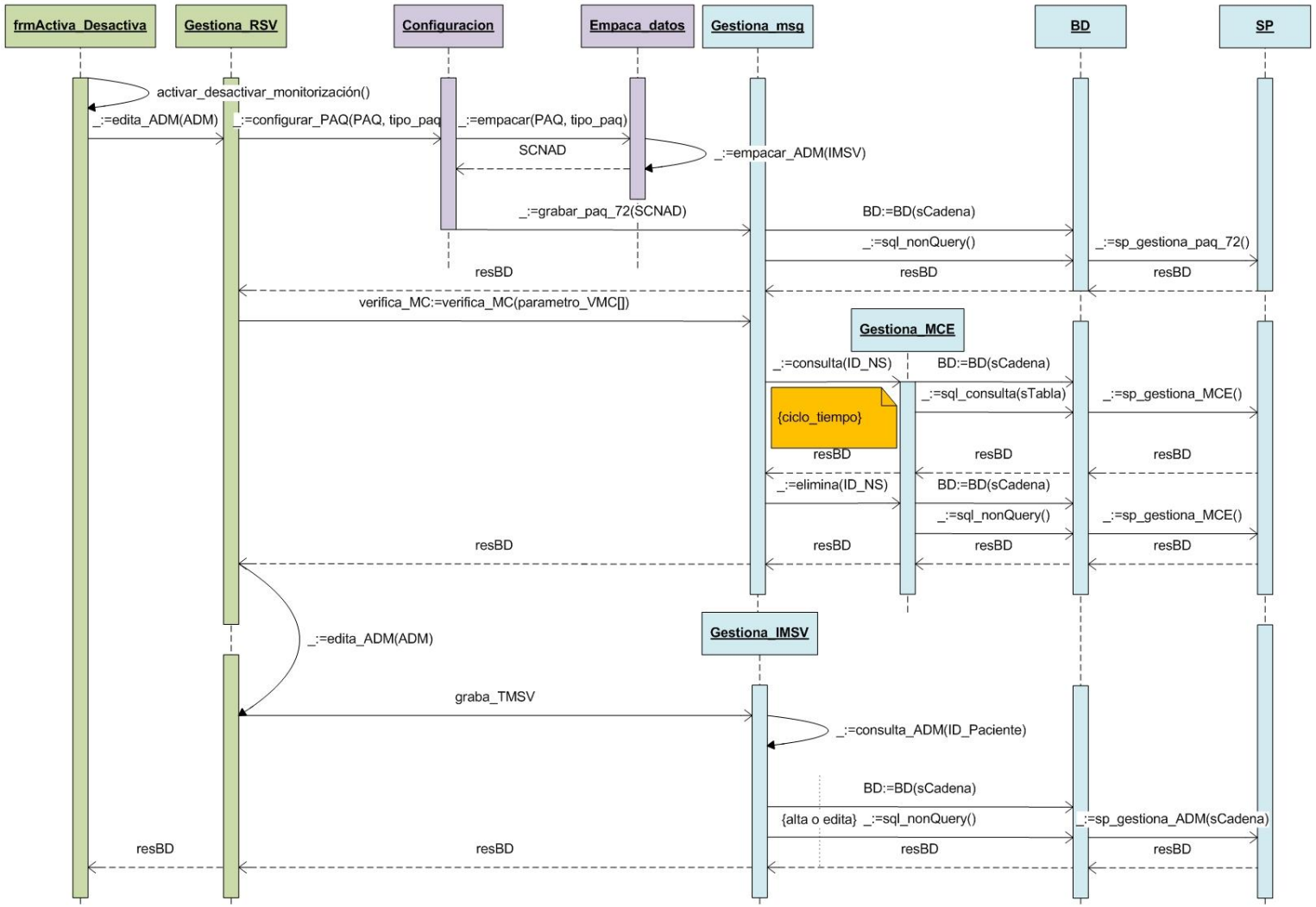


Diagrama de secuencia: **Gestión de la monitorización**



Diseño detallado

Nombre de la clase	**AU:frmActiva_Desactiva_M.aspx.cs**
Nombre del método	Selecciona_paciente
Tipo de método	Sub
Tipo de dato que regresa	Nada
Argumentos de entrada	Nada
Mensajes de error	
Clases que utiliza	Gestiona_ADM.cs
Visibilidad	Protected
Métodos llamados	AG: Gestiona_ADM:consulta_ADM() Argumentos: ID_Paciente:int AG: Gestiona_ADM:consulta_parametros() Argumentos: ID_Paciente:int
Descripción	4. Consulta el estatus actual del nodo que corresponde al paciente y lo mantiene en una variable de sesión, a su vez almacena en variables de sesión el identificador del **NC** y el identificador del **NS**. 5. Consulta los parámetros de paciente que se ha elegido y los muestra en una tabla.

Nombre de la clase	**AU:frmActiva_Desactiva_M.aspx.cs**
Nombre del método	activar_desactivar_monitorizacion
Tipo de método	Sub
Tipo de dato que regresa	Nada
Argumentos de entrada	Nada
Mensajes de error	El nodo no responde, no se pudo cambiar el estatus del nodo, la base de datos no puede ser accedida o error de BD.
Clases que utiliza	Gestiona_ADM.cs
Visibilidad	Protected
Métodos llamados	AG: Gestiona_ADM:edita_ADM() Argumentos: ADM:ADM_paq AG: Utilerias_ag:evaluar_error() Argumentos: error:int
Descripción	1. Verifica que los parámetros: intervalos de muestreo y Rangos de signos vitales hayan sido capturados y que el estatus del nodo no sea: No asociado, fuera de rango o el mismo que se quiere establecer (es decir si se quiere poner activo, debe estar inactivo y viceversa) a. **Si**: i. Carga los datos capturados al paquete **ADM** ii. Manda el mensaje al método **edita_ ADM** de la clase **Gestiona_ ADM**, para proseguir con la edición del estatus del nodo sensor correspondiente al paciente seleccionado. iii. Verifica si el proceso se ejecutó correctamente tomando como referencia el valor de **eoe** que debe ser **111** en caso de éxito:

	1. Éxito: Manda el mensaje de operación realizada. 2. Error: invoca el método **evaluar_error** de la clase **Utilerias_ag** (detallada en el documento **DD_Utilerias_ag**) para determinar el tipo de erro que se suscitó. b. **No**: Manda el mensaje correspondiente al usuario.

Nombre de la clase	**AG:Gestiona_ ADM.cs**
Nombre del método	edita_ ADM
Tipo de método	Función
Tipo de dato que regresa	resBD
Argumentos de entrada	ADM
Mensajes de error	Nada
Clases que utiliza	ICBD:Gestiona_ADM:
Visibilidad	Public
Métodos llamados	ICRS: Configuracion:configurar_PAQ Argumentos: ADM:ADM_pac Tipo_paq:string ICBD: Gestiona_Msg:verifica_MC Argumentos: Msg_esperado:int ID_NS:int Tiempo_espera:int ICBD: Gestiona_ADM:edita_ADM Argumentos: ADM:ADM_pac
Descripción	1. Asigna **MC_ADMR** a **msg_esperado** 2. Manda llamar el método **configurar_PAQ** de la clase **ICRS:Configuracion** (detallado en el documento **DDPA_configurar_PAQ**) enviando el paquete **ADM** para hacer el proceso de activación-desactivación en el nodo indicado y la palabra "**ADM**", para identificación del tipo de paquete que se trata. 3. Verifica que el paquete se pudo generar y pudo ser enviado a la cola de paquetes **paq_72**: a. **Si**: obtiene el **ID** del mensaje que fue agregado a la cola de mensajes en la base de datos, este dato fue guardado en la posición 0 del arreglo **e_datos** al momento de enviar el mensaje a la base de datos. b. Genera el arreglo **parámetro_VMC**, donde almacenará el mensaje esperado (**msg_esperado**), el Identificador del Nodo Sensor (**ID_NS**), el tiempo de espera para el mensaje esperado (**15 segundos**) y el Identificador del paquete que se enviará (**ID_paq_72**) que está en la cola de espera **paq_72**. c. Verifica si llego el mensaje de confirmación invocando el método **verifica_MC()** de la clase **Gestiona_Msg()** detallada en el documento **DD_verifica_MC**. i. éxito: graba el **ADM** a la base de datos mediante el método **graba_ADM()** de la clase **ICBD:Gestiona_ADM.** ii. Error: nada, pasa a la siguiente instrucción. 4. Regresa a la aplicación de usuario.

Nombre de la clase	ICBD:Gestiona_ADM.cs
Nombre del método	Edita_ADM()
Tipo de método	Función
Tipo de dato que regresa	resBD
Argumentos de entrada	ADM:ADM_paq
Mensajes de error	Nada
Clases que utiliza	ICBD: Gestiona_ADM, ICBD:BD
Visibilidad	Public
Métodos llamados	ICBD: Gestiona_ADM:ejecutaSentencia Argumentos: PAC:Pac_paq byTipo:byte sTabla:string
Descripción	1. invoca la función **ejecutaSentencia,** esta función se utiliza en las cuatro transacciones básicas (SELECT, INSERT, UPDATE y DELETE) y está detallada en el documento **DD_ejecutaSentencia_ADM**. Los parámetros que se pasan son: a. **ADM** que el paquete que lleva los datos de los rangos de signos vitales para el paciente. b. **byTipo** que especifica el tipo de sentencia que se va a ejecutar (en este caso 3). c. **sTabla** que es el nombre que tendrá la tabla en el **DataSet** que se tiene como respuesta en el caso de ser necesario. 2. Regresa **res** al método de donde fue invocado.

Lectura de signos vitales en tiempo real

Casos de uso

Componente:	**Sistema de monitorización y control web**	
Fecha:	02 de Marzo de 2010	
Descripción:	El usuario tendrá la posibilidad de solicitar a la red la medición de una o más veces de cualquier signo vital de un paciente.	
Actores:	Usuarios finales.	
Precondiciones:	Que el paciente esté dado de alta y esté asociado con un nodo.	
Flujo normal:	**Acción del Actor**	**Respuesta del Sistema**
	2. Selecciona el paciente del que desea tener la lectura de sus signos vitales en ese momento. 4.- Marca los signos vitales que desea ver, hace clic en el botón **CONSULTAR**. 7.- Hace clic en el botón **SALIR.**	1. Despliega los pacientes por clasificación. 3.- Despliega la lista de signos vitales. 5.- Solicita la información a la red. 6.- Despliega el paciente con la información de los signos vitales. 8.- Cierra la ventana de **CONSULTA DE SIGNOS VITALES EN TIEMPO REAL**.
Flujo alternativo:	Línea 6: Si el dispositivo implicado no responde. 6.1 Regresa el mensaje de no obtención de datos de la red, ir al paso 4.	
Postcondiciones:		

ID:	**Sistema coordinador de nodos sensores**	
Fecha:	03 de Marzo de 2010	
Descripción:	Atender mensaje **MSI** de tipo **MSI_LSV**, si el nodo de destino está en su lista, enviar el **MSI** al **NS**, Atender **MC** de tipo **MS_LSVR** (Lectura de signos vitales recibida). Atender el mensaje **MD_SV** que lleguen por el aire y enviarlos al **SCPD**. Atender el **MC_SVR** que llega del **SCPD** y Enviar el **MC** de **MC_SVR** (Signos Vitales Recibidos) al **NS**, correspondiente.	
Actores:	SCPD (Sistema de Comunicación y Persistencia a Datos), NS.	
Precondiciones:	Haber dado de alta los nodos al sistema y a la red.	
Flujo normal:	**Acción del Actor**	**Respuesta del Sistema**
	1. El **SCPD** envía mensaje **MSI_LSV**. 4.- El **NS** envía un mensaje **MD_SV.** 7.- Envía un mensaje **MC_SVR**	2. Verifica si el nodo es un nodo asociado a él. 3.- Envía al **NS**, el **MSI_LSV**. 5.- Recibe **MD_SV** del **NS** 6.- Envía al **SCPD** el **MD_SV.** 8.- Recibe el **MC_SVR**, lo reenvía al **NS**
Flujo alternativo:	Línea 2: Ningún nodo asociado corresponde al identificador del nodo: 2.1 Envía **E_NSNE** al **SCPD**. Línea 5: No recibe el **MD_SV** del **NS:** 5.1 Envía **E_SVNR** al **SCPD**. Línea 7.- No recibe el mensaje **MC_SVR**: 7.1 Envía Mensaje **E_SVNR** al NS	
Pos condiciones:		

ID:	CU_M1	
Fecha:	03 de Marzo de 2010	
Descripción:	Atender un mensaje **MSI_LSV**, realizar los cambios correspondientes y regresar el mensaje **MC_LSVR**, al **NC**.	
Actores:	SCNS (Sistema de Coordinación de Nodos Sensores), SENSOR	
Precondiciones:		
Flujo normal:	**Acción del Actor** 1.- Envía mensaje **MSI_LSV**. 6.- Envía **MC_SVR**	**Respuesta del Sistema** 2.- Lee el **MSI_LSV**, verifica que signos vitales son los de interés en ese momento. 3.- Hace la petición de lectura al sensor 4.- Empaqueta la información 5.- Envía el **MD_SV** al **NC**, queda en espera de **MC_SVR** 8.- Atiende **MC_SVR**.
Flujo alternativo:	Línea 4: No se llevó a cabo correctamente la toma de lectura, envía E_LSV Línea 8: No recibe **MC_SVR**¸ (en tiempo). 8.1.- Pasa a estado normal de transmisión	
Pos condiciones:		

Diagrama de clases: **Gestión de Signos Vitales**

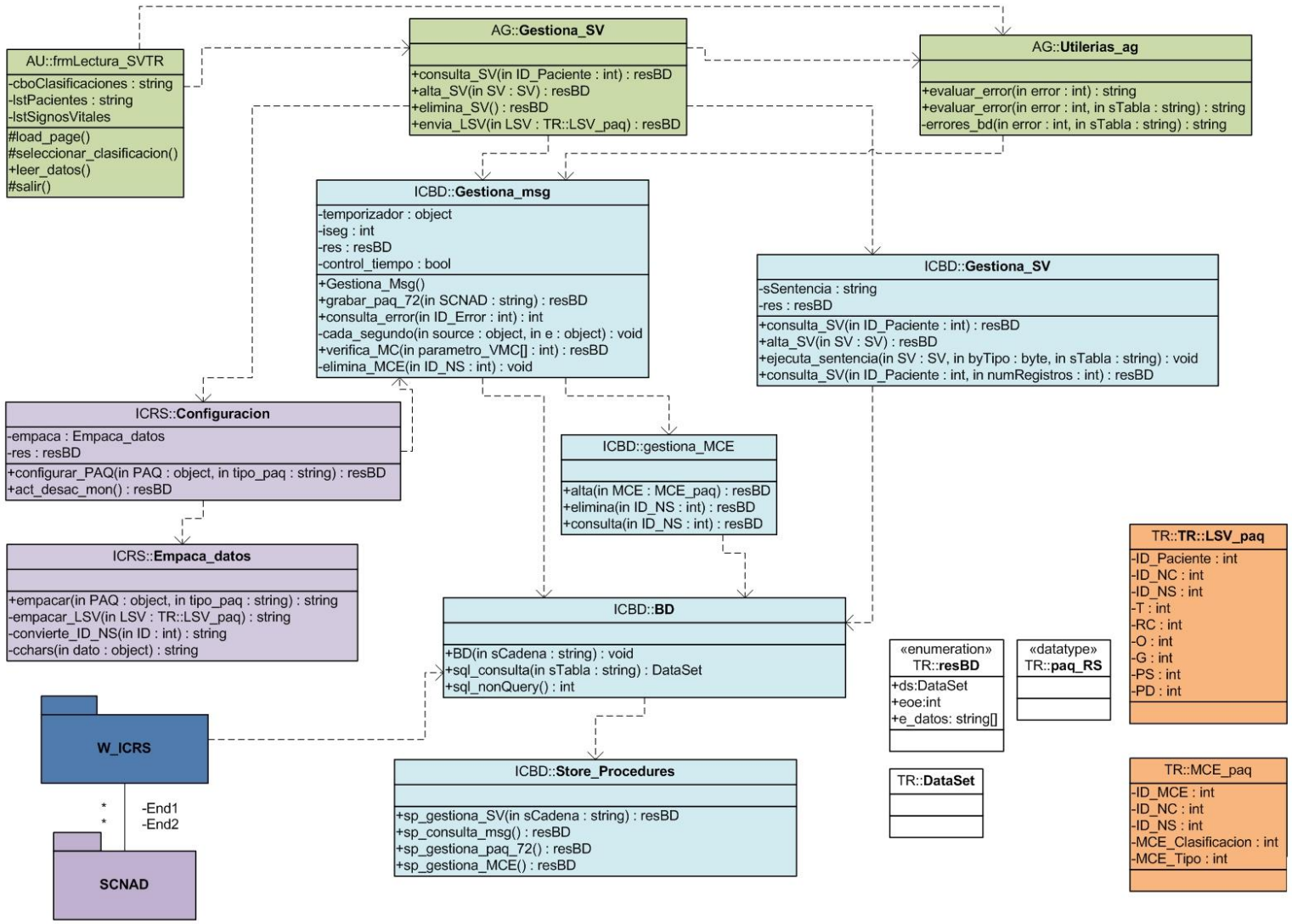
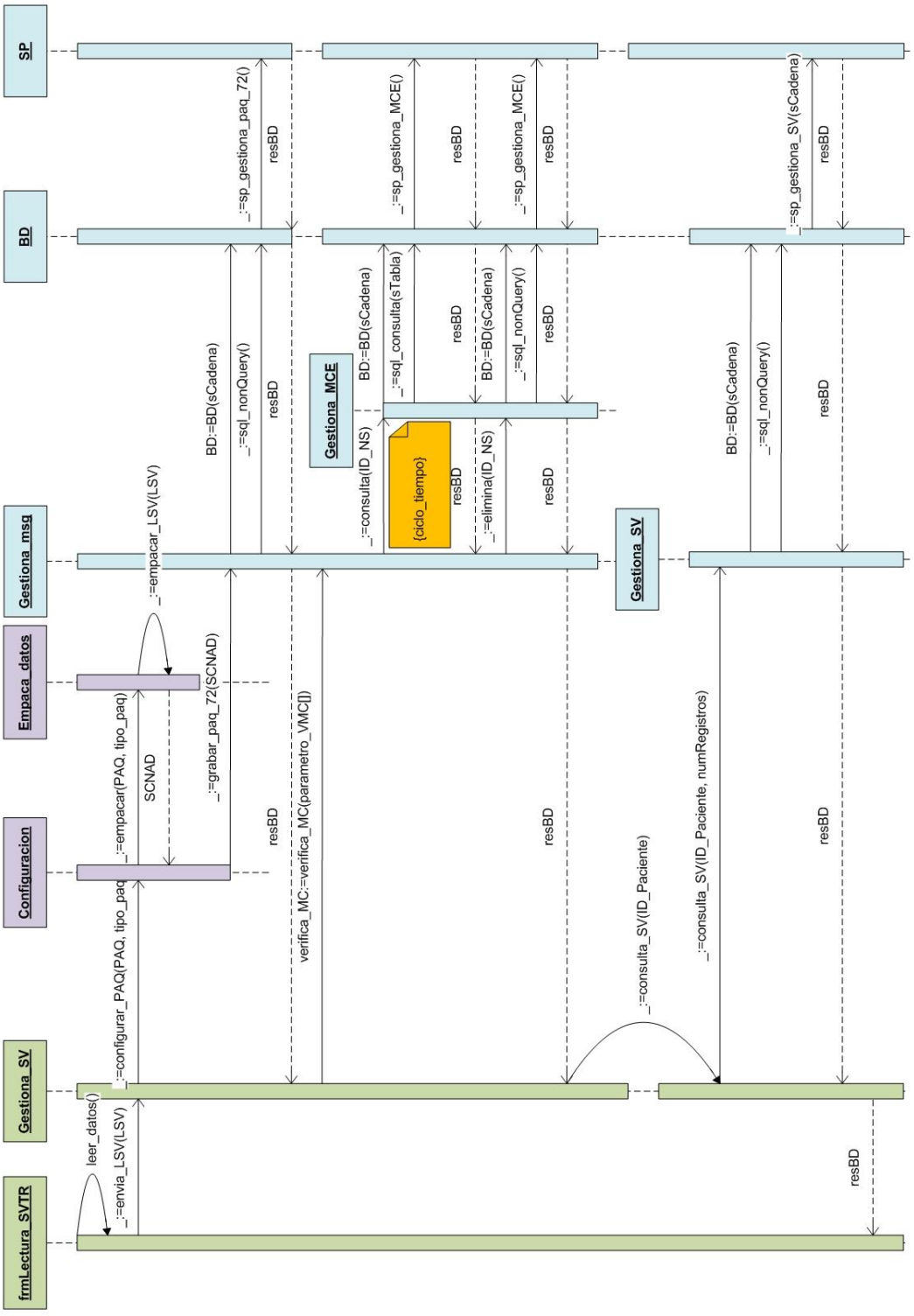


Diagrama de secuencia: **Gestión de signos vitales**



Diseño detallado

Nombre de la clase	**AU:frmLectura_SVTR.aspx.cs**
Nombre del método	leer_datos
Tipo de método	Sub
Tipo de dato que regresa	Nada
Argumentos de entrada	Nada
Mensajes de error	El nodo no responde, la base de datos no puede ser accedida o error de BD.
Clases que utiliza	Gestiona_SV.cs
Visibilidad	Protected
Métodos llamados	AG: Gestiona_SV:envia_LSV() Argumentos: ID_Paciente:int AG: Utilerias_ag:evaluar_error() Argumentos: error:int
Descripción	1. Verifica que los datos han sido capturados de manera correcta. 2. Carga el **ID** seleccionado al paquete **LSV** 3. Pone un **1** en el signo vital que se ha seleccionado para su lectura y un **0** en el que no se desea leer en ese momento. 4. Manda el mensaje al método **envia_LSV** de la clase **Gestiona_SV**, para proseguir con la consulta de los datos en el **SCNAD**. 5. Verifica si el proceso se ejecutó correctamente tomando como referencia el valor de **eoe** que debe ser **111** en caso de éxito: a. **Éxito**: Consulta los últimos **n** registros de signos vitales de que se tiene lectura. b. Presenta el resultado en pantalla. c. **Error**: invoca el método **evaluar_error** de la clase **Utilerias_ag** (detallada en el documento **DD_Utilerias_ag**) para determinar el tipo de error que se suscitó.

Nombre de la clase	**AG:Gestiona_SV.cs**
Nombre del método	envia_LSV
Tipo de método	Función
Tipo de dato que regresa	resBD
Argumentos de entrada	IMSV
Mensajes de error	Nada
Clases que utiliza	ICBD:Gestiona_Nodos:consulta_NS, ICRS:Configuracion
Visibilidad	Public
Métodos llamados	ICBD:Gestiona_Nodos:consulta_NS Argumentos ID_Paciente:int ICRS: Configuracion:configurar_PAQ Argumentos: LSV:LSV_pac Tipo_paq:string ICBD: Gestiona_Msg:verifica_MC Argumentos: parametros_VMC[]:int

	ICBD: Gestiona_SV:consulta_SV Argumentos: SV: SV_pac
Descripción	1. Asigna **MC_LSVR** a **msg_esperado** 2. Hace una consulta a la base de datos para tomar el **ID_NS** y el **ID_NC** al que está asociado el paciente. 3. Asigna a los atributos correspondientes el **ID_NS** y el **ID_NC.** 4. Manda llamar el método **configurar_PAQ** de la clase **ICRS:Configuracion** (detallado en el documento **DDPA_configurar_PAQ**) enviando el paquete **LSV** para hacer el proceso de establecer los intervalos de muestreo en el nodo indicado y la palabra "**LSV**", para identificación del tipo de paquete que se trata. 5. Verifica que el paquete se pudo generar y pudo ser enviado a la cola de paquetes **paq_72**: a. **Si**: obtiene el **ID** del mensaje que fue agregado a la cola de mensajes en la base de datos, este dato fue guardado en la posición 0 del arreglo **e_datos** al momento de enviar el mensaje a la base de datos. b. Genera el arreglo **parámetro_VMC**, donde almacenará el mensaje esperado (**msg_esperado**), el Identificador del Nodo Sensor (**ID_NS**), el tiempo de espera para el mensaje esperado (**15 segundos**) y el Identificador del paquete que se enviará (**ID_paq_72**) que está en la cola de espera **paq_72**. c. Verifica si llego el mensaje de confirmación invocando el método **verifica_MC()** de la clase **Gestiona_Msg()** detallada en el documento **DD_verifica_MC**. i. **éxito**: hace la lectura de los **SV** a la base de datos mediante el método **consulta_SV()** de la clase **ICBD:Gestiona_SV.** ii. **Error**: nada, pasa a la siguiente instrucción. 6. Regresa a la aplicación de usuario.

Nombre de la clase	**ICBD:Gestiona_SV.cs**
Nombre del método	consulta_SV()
Tipo de método	Función
Tipo de dato que regresa	resBD
Argumentos de entrada	ID_Paciente:int, numRegistros:int
Mensajes de error	Nada
Clases que utiliza	ICBD:BD
Visibilidad	Public
Métodos llamados	ICBD: BD:BD() Argumentos: sSentencia:string ICBD: BD:consulta_sql() Argumentos: sTabla:string
Descripción	1. Construye la sentencia **SQL** que ejecutará con los parámetros de entrada 2. invoca la función **BD**, pasando la sentencia sql para hacer la consulta de los últimos registros de signos vitales. 3. Invoca el método **consulta_sql** pasando como argumento el nombre de la tabla que se usará en el contenedor de los registros que regresen de la consulta realizada.

	4. Regresa **res** al método de donde fue invocado. 5. **NOTA**: A diferencia de la mayoría de transacciones con la base de datos, en este caso no se utiliza un store procedure debido a que la consulta se sale del estándar en el que están trabajando los store produre creados. Modificar el strore procedure lo haría muy diferente al estándar lo que podría causar confusión y hacer un store procedure para este único fin resulta impráctico.

Diseño detallado de procesos generales

Maestría en Ingeniería

Universidad Autónoma de Baja California

Interface de comunicación con la red de sensores

Nombre de la clase	**ICRS:frmW_ICRS.cs**
Nombre del método	tmrControl_tiempo_Tick
Tipo de método	Sub
Tipo de dato que regresa	Nada
Argumentos de entrada	Sender:object, e:EventArgs
Mensajes de error	Nada
Clases que utiliza	Gestiona_Msg, RS232
Visibilidad	Private
Métodos llamados	RS232:lee_datos() Argumentos: Gestiona_Msg:consulta_paquete_72() Argumentos: RS232:cerrar_puerto() Argumentos: RS232:escribir_puerto() Argumentos: Msg:sbyte
Descripción	1. Usa la instancia **"puerto"** de RS232 para invocar el método **lee_datos** que verificará si existen datos en el buffer del puerto para hacer la clasificación y enviarlos a la base de datos. 2. Verifica si se ha cumplido el tiempo en el que se debe consultar la base de datos para enviar paquetes que existan en las colas de envío de paquetes. a. **Si**: i. Declara la instancia **gmsg** para utilizar la clase **Gestiona_Msg**. ii. Invoca el método **consulta_paquete_72()** de la clase **Gestiona_Msg** que regresará los paquetes al **ds** para enviar al **SCNAD**. iii. Verifica si existen paquetes para enviar: 1. **Si**: a. Declara la variable **msg** para almacenar el paquete. b. Declara la variable que regresará el mensaje de error o de éxito y supone que existe un error 110 (error genérico). c. Hace un ciclo con cada registro en el paquete **ds**, por cada registro: i. Asigna el mensaje a msg. ii. Invoca cerrar puerto para no leer en ese instante y poder enviar el flujo. iii. Invoca **escribir_puerto**, pasando como argumento el mensaje **msg**. iv. Verifica si hubo éxito en la transferencia. 1. **Si**: invoca el método **elimina_paquete_72**,

	para borrar de la cola el mensaje que ya fue enviado al **SCNAD**. 2. Regresa a 0 el contador de segundos que ayuda a controlar el tiempo en el que se deben estar haciendo las consultas a la base de datos para saber si existen paquetes en la cola de envío (paq_72). 3. Incremente en uno el contador de segundos.

Nombre de la clase	**ICRS:RS232.cs**
Nombre del método	lee_datos()
Tipo de método	Sub
Tipo de dato que regresa	Nada
Argumentos de entrada	Nada
Mensajes de error	Nada
Clases que utiliza	RS232, Gestiona_Msg
Visibilidad	Public
Métodos llamados	RS232:desempacar Argumentos: Gestiona_Msg:clasificar_paquete Argumentos: lsPaquetes:<string>
Descripción	1. Declara la lista (string) **lsPaquetes** que servirá para almacenar los paquetes que estén en el puerto de la computadora. 2. Invoca el método desempacar de la misma clase para obtener la lista de paquetes. 3. Verifica si existen paquetes en la lista lsPaquetes. a. **Si**: i. Crea la instancia **gmsg**, de la clase **Gestiona_Msg** para dar el seguimiento correspondiente a cada paquete de la lista. ii. Invoca el método **clasificar_paquete** para continuar con proceso de enviar los paquetes a la tabla que corresponda en la base de datos. 4. Limpia el buffer para eliminar los paquetes que ya han sido atendidos.

Nombre de la clase	**ICRS:RS232.cs**
Nombre del método	desempacar()
Tipo de método	Función
Tipo de dato que regresa	List <string>
Argumentos de entrada	Nada
Mensajes de error	Nada
Clases que utiliza	Convert, String
Visibilidad	Public
Métodos llamados	String:Split Argumentos: divisor:char Convert:ToChar

	Argumentos: elemento:int32
Descripción	1. Declara la variable **sCadena** de tipo **string** que servirá de buffer temporal para los datos que están almacenados en el **buffer** del puerto en formato **byte**. 2. Declara el arreglo **sPaquetes** donde almacenará los paquetes que se encuentran en el buffer. 3. Declara la lista **lsPaquetes** que será devuelta con los paquetes como contenido. 4. Declara la variable **divisor** que servirá para identificar donde termina y empieza un paquete. 5. Ejecuta un ciclo mientras haya elementos en la variable **datos** que es la tiene el contenido del buffer descargado y con cada elemento: a. Compara si el elemento es el código asccii 58 (:) i. **Si**: Convierte el elemento a Int32, lo pone en formato **string** y luego a **Char** para posteriormente Agregarlo a la variable SCadena. ii. **No**: Convierte el elemento a Int32, lo pone en formato **string** para posteriormente Agregarlo a la variable SCadena. 6. Utiliza el método **Split** de la clase **string** para dividir los paquetes identificando cada división por el carácter char (:) que se encuentra en la variable **"divisor"** y asignarlos al arreglo **sPaquetes**. 7. Ejecuta un ciclo mientras haya elementos en el arreglo string y con cada elemento: a. Verifica si el elemento es diferente a vació y su longitud mayor a 8 que es la longitud menor que puede tener un paquete correcto. i. **Si**: agrega el elemento (paquete) a la lista de paquetes **lsPaquetes**. 8. Regresa la lista de paquetes **lsPaquetes**.

Nombre de la clase	**ICRS:RS232.cs**
Nombre del método	Clasificar_paquete ()
Tipo de método	Sub
Tipo de dato que regresa	Bool
Argumentos de entrada	lsPaquetes <string>
Mensajes de error	Nada
Clases que utiliza	String, Convert, Gestiona_Msg
Visibilidad	Public
Métodos llamados	String:substring() Argumentos: Inicio:int Longitud:int Convert:int32() Argumentos: campos:string Gestiona_Msg: Grabar_paquete_bd() Argumentos: MCE:MCE_paq MSG:int
Descripción	1. Declara las variables que servirán para la gestión de la información:

	a. **cDivisor**: para identificar el lugar donde existe un carácter especial (|) que separa cada campo en el paquete. b. **sCampos**: arreglo que servirá para almacenar los campos con que cuente el paquete. c. **sPaq_completo**: variable para guardar el paquete completo. d. **b**: variable booleana que servirá para regresar un valor verdadero en el caso de que todo el proceso haya resultado exitoso. e. **res**: objeto que servirá para almacenar el valor devuelto por el método que graba el paquete en la base de datos. 2. Hace un ciclo mientras existan paquetes en la **lsPaquetes** que entra como argumento, y con cada paquete: a. Asigna el paquete a la variable **sPaq_completo** para poder hacerle modificaciones. b. Elimina el carácter (|) del inicio y del fin del paquete utilizando la función **substring** de la clase **string** en el caso de que existan. c. Divide el **sPaq_completo** en sub partes identificadas por el carácter (|) y agrega las partes al arreglo **sCampos** d. Verifica la integridad del paquete al comparar su longitud real con la que dice dentro del mismo paquete que debe tener (esta fue puesta en el momento del envío dentro del dispositivo sensor). i. **Si**: 1. Convierte el campo 5 a **int32** y lo asigna a la variable **MSG**, que será la variable que sirva de selector para identificar el tipo de mensaje del que se trata. 2. Inicia el **switch** con la variable **MSG** como selector y en cada caso: a. Crea la instancia del la clase que define el paquete al que corresponde el que se está leyendo. b. Asigna a cada atributo del objeto el campo correspondiente. c. Invoca el método **grabar_paquete_bd** enviando como argumento el paquete y el tipo de paquete de que se trata. d. Si la respuesta regresada el **111** asigna VERDADERO a **b.** 3. Limpia la variable **sCampos** para atender otro paquete. 3. Regresa **b**.

Nombre de la clase	**ICRS:Gestiona_Msg.cs**
Nombre del método	grabar_paquete_bd()
Tipo de método	Función
Tipo de dato que regresa	resBD
Argumentos de entrada	TR_paq:object, iTipo_paq:int
Mensajes de error	Nada
Clases que utiliza	BD
Visibilidad	Public
Métodos llamados	BD:BD

	Argumentos: sCadena:string BD:sql_NonQuery Argumentos: sTabla:string
Descripción	1. Crea la instancia **res** para el objeto que contendrá la respuesta. 2. Declara la variable **sCadena** para poner la sentencia **SQL** que ejecutará el store procedure. 3. Ejecuta un switch teniendo como selector el argumento iTipo_paq de entrada para identificar qué tipo de paquete llegó. En cada caso: a. Crea la instancia del objeto dependiendo del tipo de paquete del que se trate. b. Asigna a **sCadena**, el nombre del "store procedure" que corresponda. c. Termina de construir la cadena con los atributos del objeto y el parámetro **2** que significa una inserción para la base de datos. 4. Crea el objeto **bd** con la instancia de la clase **BD** (descrita en el documento **DD_BD**), pasando como argumento la sentencia **SQL**, que ejecutará el "store procedure" en la base de datos. 5. Invoca el método **sql_NonQuery()** de la clase **bd**. 6. Regresa **res**

Nombre de la clase	**ICRS:Gestiona_Msg.cs**
Nombre del método	Consulta_paquete_72
Tipo de método	Función
Tipo de dato que regresa	resBD
Argumentos de entrada	Nada
Mensajes de error	Nada
Clases que utiliza	BD
Visibilidad	Public
Métodos llamados	BD:BD Argumentos: sCadena:string BD:sql_consulta Argumentos: sTabla:string
Descripción	1. Crea la instancia **res** para el objeto que contendrá la respuesta. 2. Crea la instancia para el objeto **BD** (la clase se BD se encuentra descrita en el documento DD_BD), pasando como argumento la sentencia que ejecutará el **sp_gestiona_paq_72**. 3. Ejecuta el método **sql_consulta** de la clase **BD** pasando como argumento el nombre de la tabla que contendrá los registros del resultado de la consulta. 4. Regresa **res**

Nombre de la clase	**ICRS:RS232.cs**
Nombre del método	cerrar_puerto
Tipo de método	Sub
Tipo de dato que regresa	Nada
Argumentos de entrada	Nada

Mensajes de error	Nada
Clases que utiliza	SerialPort
Visibilidad	Public
Métodos llamados	Close()
Descripción	1. Ejecuta el método **Close** de la clase **SerialPort** para cerrar el puerto mediante el cual se transiten los datos.

Nombre de la clase	**ICRS:RS232.cs**
Nombre del método	escribir_puerto
Tipo de método	Sub
Tipo de dato que regresa	Int
Argumentos de entrada	paq_72
Mensajes de error	No se puede abrir el puerto
Clases que utiliza	SerialPort, Convert, Exception
Visibilidad	Public
Métodos llamados	SerialPort:Open() Argumentos: Convert:ToByte Argumentos: cLetra:char SerialPort:Write Argumentos: mensaje:byte offset:int longitud:int
Descripción	1. Verifica si el puerto está cerrado: a. **Si**: Abre el puerto. 2. Declara el arreglo **mensaje** de tipo byte y longitud 32 donde se almacenará el paquete que se enviará al **SCNAD**. 3. Le asigna 0 a cada elemento del arreglo. 4. Declara la variable **i** de tipo **int** que servirá de índice para el manejo del arreglo, inicializa la variable i en 0. 5. Declara la variable **respuesta**, esta variable servirá para regresar el valor **int** al método que invocó este procedimiento. 6. Inicia un ciclo que se repite mientras hayan letras en el paquete **paq_72**, que fue el argumento de entrada. En cada iteración: a. Convierte la letra a byte con el método ToByte de la clase Convert y asigna el resultado al arreglo mensaje en el índice i. b. Incrementa el índice i en uno. 7. Invoca el método **Write** de la clase **SerialPort** pasando como argumentos a. Mensaje: El paquete convertido a bytes. b. 0: valor que indica que no existe desfase en la escritura del paquete. c. i: La longitud del paquete. 8. Verifica si hubo errores en la escritura al puerto: a. **Si**: asigna **114** (error al escribir en el puerto) a la variable respuesta. 9. Regresa la respuesta al método de donde fue invocado.

Verifica mensajes de confirmación

Nombre de la clase	**ICBD:Gestiona_Msg.cs**
Nombre del método	Verifica_MC
Tipo de método	Función
Tipo de dato que regresa	resBD
Argumentos de entrada	Parametro_VMC[] (MC_:int, ID_NS:int, tiempo_espera:int, int ID_paq_72)
Mensajes de error	Nada
Clases que utiliza	Gestiona_Msg,Gestiona_MCE
Visibilidad	Public
Métodos llamados	Gestiona_MCE:consulta Argumentos: ID_NS:int Gestiona_Msg:cada_segundo Argumentos: ID_NS:int Gestiona_Msg:elimina_MCE Argumentos: ID_NS:int
Descripción	1. Pone a funcionar el temporizador que permitirá hacer los ciclos en el tiempo de espera solicitado. 2. Ejecuta un ciclo tardando 1 segundo entre cada iteración. El ciclo se rompe cuando la variable llega a ser igual en valor al parámetro tiempo espera ó cuando se encuentra el mensaje esperado, lo que suceda primero: a. Valida si el valor de **control_tiempo** es verdadero, esta variable tiene verdadero cada dos segundos con el fin de no saturar a la base de datos con consultas que pudieran tomar fracciones de segundo entre una y otra sin rendir resultados. i. **Si**: 1. Asigna **115** a **eoe** (nodo no encontrado) 2. Trae los mensajes que el **SCNAD** ha enviado al nodo. 3. Verifica si el mensaje es de confirmación y si coincide con el mensaje esperado: a. **SI:** se asigna **111** a **eoe** y elimina el mensaje de MCE. b. **NO**: se asigna lo que traiga el registro en el campo correspondiente al tipo y elimina el mensaje de MCE. 4. Pone **control_tiempo** a falso. ii. **NO**: hace otra iteración. 3. Verifica si el tiempo de espera fue agotado a. **Si**: elimina el paquete **paq_72** de la cola de paquetes en la base de datos para que no se pierda la integridad de los datos entre la base de datos y el **SATD** 4. Inhabilita el temporizador. 5. Pone el contador de segundos a 0. 6. Regresa la variable **res** al método que invocó a **verifica_MC()**.

Nombre de la clase	ICBD:Gestiona_MCE.cs
Nombre del método	Consulta
Tipo de método	Función
Tipo de dato que regresa	resBD
Argumentos de entrada	ID_NS:int
Mensajes de error	Nada
Clases que utiliza	Gestiona_MCE
Visibilidad	Public
Métodos llamados	Gestiona_MCE:ejecuta_sentencia Argumentos: MCE:MCE_paq byTipo:byte sTabla:string
Descripción	1. Crea la instancia de **MCE_paq**. 2. Asigna el **ID_NS** al atributo correspondiente de **MCE_paq** 3. Invoca el método **ejecuta_sentencia**, con los parámetros: a. **MCE_paq**: Paquete con el que se formará la sentencia **sql**. b. **Tipo**: La transacción, en este caso 1, para consultar. a. **Tabla**: En este caso **MCE**. 4. Regresa **resBD** al método de donde fue invocado.

Nombre de la clase	ICBD:Gestiona_Msg.cs
Nombre del método	Cada_segundo
Tipo de método	event
Tipo de dato que regresa	Nada
Argumentos de entrada	source:object, e:ElapsedEventArgs
Mensajes de error	Nada
Clases que utiliza	Nada
Visibilidad	Private static
Métodos llamados	
Descripción	1. Aumenta en uno el contador de segundos **iseg** que se utiliza en la condición del ciclo **while**. 2. Si **iseg** es impar pone a verdadero la variable **control_tiempo** para que las consultas a la base de datos se hagan cada dos segundos en el ciclo **while**.

Nombre de la clase	ICBD:Gestiona_Msg.cs
Nombre del método	Elimina_MCE
Tipo de método	Sub
Tipo de dato que regresa	Nada
Argumentos de entrada	ID_NS:int
Mensajes de error	Nada
Clases que utiliza	Gestiona_MCE
Visibilidad	Private
Métodos llamados	Gestiona_MCE:elimina

	Argumentos ID_NS
Descripción	1. Crea la instancia de **Gestiona_MCE**. 2. Invoca el método elimina de la clase **Gestiona_MCE**.

Nombre de la clase	**ICBD:Gestiona_MCE.cs**
Nombre del método	Elimina
Tipo de método	Función
Tipo de dato que regresa	resBD
Argumentos de entrada	ID_NS:int
Mensajes de error	Nada
Clases que utiliza	Gestiona_MCE
Visibilidad	Public
Métodos llamados	Gestiona_MCE:ejecuta_sentencia Argumentos: MCE:MCE_paq byTipo:byte sTabla:string
Descripción	1. Crea la instancia de **MCE_paq**. 2. Asigna el **ID_NS** al atributo correspondiente de **MCE_paq** 3. Invoca el método **ejecuta_sentencia**, con los parámetros: a. **MCE_paq**: Paquete con el que se formará la sentencia **sql**. b. **Tipo**: La transacción, en este caso 4, para eliminar. c. **Tabla**: En este caso **MCE**. 4. Regresa **resBD** al método de donde fue invocado.

Nombre de la clase	**ICBD:Gestiona_MCE.cs**
Nombre del método	ejecuta_sentencia
Tipo de método	sub
Tipo de dato que regresa	Nada
Argumentos de entrada	MCE:MCE_paq, byTipo:byte sTabla:string
Mensajes de error	Nada
Clases que utiliza	BD
Visibilidad	Private
Métodos llamados	BD:sql_consulta Argumentos: sTabla:string BD:sql_NonQuery Argumentos:
Descripción	1. Crea la sentencia **SQL** para ejecutar el store procedure **sp_gestiona_MCE**. 2. Crea la instancia de **BD**. (Clase detallada en **DD_BD**) 3. Verifica el tipo de transacción a realizar a. 1: Ejecuta el método **sql_consulta** de **bd**. b. Otro: Ejecuta el método **sql_NonQuery** de **bd**.

Ejecuta Sentencia

Nombre de la clase	**ICBD:Gestiona_RSV.cs**
Nombre del método	ejecutaSentencia
Tipo de método	Función
Tipo de dato que regresa	Void
Argumentos de entrada	RSV:RSV_paq, byTipo:byte, sTabla:string
Mensajes de error	Nada
Clases que utiliza	ICBD:BD
Identificador de visibilidad	Private
Métodos llamados	ICBD:BD:sql_consulta Argumentos sTabla:string ICBD:BD:sql_NonQuery Argumentos
Descripción	1. Con los argumentos **PAC** y **byTipo** construye la sentencia que ejecutará el store procedure: sp_gestiona_pacientes. De esta manera realizará una transacción SQL dependiendo del valor que tenga **byTipo**: a. 1=SELECT b. 2=INSERT c. 3=UPDATE d. 4=DELETE 2. Verifica el valor de byTipo para saber que atributo del objeto **res** debe ser utilizado: a. 1: Llena el DataSet del objeto **res**. b. 2, 3, 4: Llena el atributo **eoe** del objeto **res**.

NOTA: La clase **BD** que contiene los métodos **BD**, **sql_NonQuery** y **sql_consulta** está definida en el documento **DD_BD**.

Nombre de la clase	**ICBD:Gestiona_IMSV.cs**
Nombre del método	ejecutaSentencia
Tipo de método	Función
Tipo de dato que regresa	Void
Argumentos de entrada	IMSV:IMSV_paq, byTipo:byte, sTabla:string
Mensajes de error	Nada
Clases que utiliza	ICBD:BD
Identificador de visibilidad	Private
Métodos llamados	ICBD:BD:sql_consulta Argumentos sTabla:string ICBD:BD:sql_NonQuery Argumentos
Descripción	1. Con los argumentos **PAC** y **byTipo** construye la sentencia que

	ejecutará el store procedure: sp_gestiona_pacientes. De esta manera realizará una transacción SQL dependiendo del valor que tenga **byTipo**: a. 1=SELECT b. 2=INSERT c. 3=UPDATE d. 4=DELETE 2. Verifica el valor de byTipo para saber que atributo del objeto **res** debe ser utilizado: a. 1: Llena el DataSet del objeto **res**. 2, 3, 4: Llena el atributo **eoe** del objeto **res**.

NOTA: La clase **BD** que contiene los métodos **BD**, **sql_NonQuery** y **sql_consulta** está definida en el documento **DD_BD**.

Nombre de la clase	**ICBD:Signos_Vitales.cs**
Nombre del método	ejecutaSentencia
Tipo de método	Función
Tipo de dato que regresa	Void
Argumentos de entrada	SV:SV_paq, byTipo:byte, sTabla:string
Mensajes de error	Nada
Clases que utiliza	ICBD:BD
Identificador de visibilidad	Private
Métodos llamados	ICBD:BD:sql_consulta Argumentos sTabla:string ICBD:BD:sql_NonQuery Argumentos
Descripción	1. Con los argumentos **PAC** y **byTipo** construye la sentencia que ejecutará el store procedure: sp_gestiona_pacientes. De esta manera realizará una transacción SQL dependiendo del valor que tenga **byTipo**: a. 1=SELECT b. 2=INSERT c. 3=UPDATE d. 4=DELETE 2. Verifica el valor de byTipo para saber que atributo del objeto **res** debe ser utilizado: a. 1: Llena el DataSet del objeto **res**. b. 2, 3, 4: Llena el atributo **eoe** del objeto **res**.

NOTA: La clase **BD** que contiene los métodos **BD**, **sql_NonQuery** y **sql_consulta** está definida en el documento **DD_BD**.

Clase BD

Nombre de la clase	ICBD:BD.cs
Nombre del método	BD
Tipo de método	Constructor
Tipo de dato que regresa	Nada
Argumentos de entrada	sCadena:string
Mensajes de error	No se encontró el archivo base.mot No se pudo establecer comunicación con el servidor No se pudo establecer comunicación con la base de datos
Clases que utiliza	IO SQLClient
Identificador de visibilidad	Public
Métodos llamados	IO:GetCurrentDirectory Argumentos IO:ReadLine Argumentos sPath:string SQLCliente:sqlConnection Argumentos sCadenaConexion:string SQLCliente:sqlCommand Argumentos sCadena:string cn:sqlConnection
Descripción	1. Lee el archivo donde están los datos de acceso al servidor y la base de datos. 2. Asigna los valores de la cadena encontrada en el archivo a las variables: sUsuario, sClave, sServidor, sCatalogo. 3. Construye la cadena de conexión con las variables. 4. Abre la conexión a la base de datos dejándola activa para ser utilizada por el objeto “command” de sql.

Nombre de la clase	ICBD:BD.cs
Nombre del método	sql_consulta
Tipo de método	Función
Tipo de dato que regresa	DataSet
Argumentos de entrada	sTabla:String
Mensajes de error	La base de datos no tiene una conexión abierta. Error en la ejecución del “strore procedure”.
Clases que utiliza	DataAdapter
Identificador de visibilidad	Public
Métodos llamados	DataAdapter:SelectCommand Argumentos: Cmd:Command DataAdapter:Fill Argumentos: Ds:DataSet

	sTabla:string
Descripción	1. Asigna la sentencia de consulta al método SelectCommand de la clase DataAdapter. 2. Llena el DataSet con los datos que resulten de la consulta en el objeto virtual llamado como indique el nombre de la variable sTabla. 3. Verifique que se hayan encontrado registros: a. No: Asigna null al DataSet. 4. Regresa el DataSet

Nombre de la clase	**ICBD:BD.cs**
Nombre del método	Sql_NonQuery
Tipo de método	Función
Tipo de dato que regresa	int
Argumentos de entrada	Nada
Mensajes de error	La base de datos no tiene una conexión abierta. Error en la ejecución del “strore procedure”.
Clases que utiliza	sqlCommand
Identificador de visibilidad	Public
Métodos llamados	executeNonQuery
Descripción	1. Verifica si la conexión está cerrada: a. Si: Abre la conexión. 2. Ejecuta la operación sobre la base de datos mediante la conexión abierta y la sentencia establecida en el constructor al objeto command. 3. Verifica si la conexión está abierta: a. Si: Cierra la conexión.

Sistema coordinador de nodos sensores

Nombre de la clase	**SCNS.cs**
Nombre del método	limpia_paquete
Tipo de método	Sub
Tipo de dato que regresa	Nada
Argumentos de entrada	Nada
Mensajes de error	Nada
Clases que utiliza	Nada
Función que lo invoca	envía_confirmacion_NCL, App_HandleMcpsInput
Métodos llamados	
Descripción	1. Declara la variable i que servirá de contador 2. Ejecuta un ciclo mientras i sea menor que **TAM_SCNAD_paq**. En cada iteración: a. Se asigna '\0' a cada elemento del arreglo **paq_72**

Nombre de la clase	**SCNS.cs**
Nombre del método	Inicializa_ID_NS
Tipo de método	Sub
Tipo de dato que regresa	Nada
Argumentos de entrada	Nada
Mensajes de error	Nada
Clases que utiliza	Nada
Función que lo invoca	envia_confirmacion_NCL
Métodos llamados	
Descripción	1. Declara las variables **i**, **j** de tipo **int** que servirán de contador en los ciclos. 2. Ejecuta un ciclo mientras **i** sea menor que **MAXNS**, en cada iteración: a. Ejecuta un ciclo mientras j es menor que 2, en cada iteración: i. Se asigna 0 a la variable **maDeviceShortAddress**, utilizando los índices i, j.

Nombre de la clase	**SCNS.cs**
Nombre del método	empaqueta_paq_27
Tipo de método	Función
Tipo de dato que regresa	uint8_t
Argumentos de entrada	*p:mcpsToNwkMessage
Mensajes de error	Nada
Clases que utiliza	Nada
Función que lo invoca	App_HandleMcpsInput
Métodos llamados	SCNS:asigna_pMsdu_a_paq_27 Argumentos: pMsdu:uint8_t[] msduLength:int l_paq:int
Descripción	1. Declara la variable **l_paq** que servirá para obtener la longitud del paquete **paq_27**.

	2. Se hace la asignación de datos al paquete **paq_27**, por cada paso la variable l_paq va incrementando, dependiendo de la cantidad de datos (bytes) que se asignaron: a. Asigna el carácter de inicio de paquete en ASCII (58) b. Asigna la dirección corta en notación little_endian c. Asigna la dirección del nodo que envió el paquete en notación Little_endian. d. Asigna la calidad del enlace que hubo cuando se hizo la transmisión. e. Invoca la función **asigna_pMsdu_a_paq_27** para asignar el **payload**, al paquete. f. Asigna la longitud del paquete g. Asigna el carácter de fin de paquete en ASCII (58) 3. Regresa la longitud del paquete.

Nombre de la clase	**SCNS.cs**
Nombre del método	asigna_pMsdu_a_paq_27
Tipo de método	Sub
Tipo de dato que regresa	Nada
Argumentos de entrada	p:uint8_t[] longitud_pMsdu:int pos_inicio:int
Mensajes de error	Nada
Clases que utiliza	Nada
Función que lo invoca	empaqueta_paq_27
Métodos llamados	
Descripción	1. Declara la variable **i** de tipo **uint8_t** que servirá como contador. 2. Inicializa i en 0. 3. Ejecuta un ciclo mientras **i** es menor que **longitud_pMsdu**, por cada iteración: a. Asigna a **paq_27**[pos_inicio], lo que contenga **p[i]**. b. Incrementa **pos-inicio**. c. Incrementa **i**.

Nombre de la clase	**SCNS.cs**
Nombre del método	envia_confirmacion_NCL
Tipo de método	Sub
Tipo de dato que regresa	Nada
Argumentos de entrada	Nada
Mensajes de error	Nada
Clases que utiliza	Nada
Función que lo invoca	AppTask en el caso stateStartCoordinatorWaitConfirm
Métodos llamados	
Descripción	1. Se hace la asignación de datos al paquete **paq_27**: a. Asigna el carácter de inicio de paquete en ASCII (58) b. Asigna la dirección corta en notación little_endian c. Asigna 0 a los campos donde se debe asignar la dirección del nodo que envió el paquete por ser un paquete que se origina en el coordinador.

	d. Asigna 0 al campo donde se debe estar la calidad del enlace porque el paquete se origina en el nodo coordinador. e. Asigna MC al siguiente campo con el que indica que es un mensaje de configuración. f. Asigna NCL al siguiente campo para indicar que el mensaje confirma que el nodo coordinador está listo. g. Asigna la dirección de la red. h. Asigna el canal por el que va a transmitir el nodo coordinador. i. Asigna la longitud del paquete. j. Asigna el carácter de fin de paquete en ASCII (58)

Nombre de la clase	**SCNS.cs**
Nombre del método	App_TransmitUartData
Tipo de método	Sub
Tipo de dato que regresa	Void
Argumentos de entrada	Nada
Mensajes de error	Nada
Clases que utiliza	Nada
Función que lo invoca	App_Init:listen
Métodos llamados	SCNS:UartX_GetByteFromRxBuffer() Argumentos: &paq_72:uint8_t[] SCNS:Msg_AllocType() Argumentos: nwkToMcpsMessage_t: struct nwkToMcpsMessage_tag SCNS:MSG_Send() Argumentos: NWK_MCPS:conts int mpPacket:nwkToMcpsMessage
Descripción	Este método envía al aire el mensaje que le llega a través del puerto serial, las condiciones para que lo envíe es que los datos que llegan estén entre los caracteres : (58) que han sido establecidos como inicio y fin de paquete. 1. Verifica que la cantidad de bytes recibidos sea menor a la máxima permitida mediante la comparación de **bytes_recibidos** con **TAM_SCNAD_paq**: a. **Si**: i. Toma los datos que hay en el buffer del puerto serie. ii. Verifica si aun tiene datos: 1. **Si**: incrementa la variable **bytes_recibidos** en uno para llevar el control de la cantidad de bytes que han llegado. 2. Verifica si el último byte recibido es el código 58 (:): a. **Si**: 1. Verifica que haya espacio para introducir un nuevo paquete a la cola de paquetes y que **mpPacket** no tenga asignada una dirección de memoria: a. **Si**: solicita una localidad de memoria para alojar el nuevo paquete. 2. Verifica que haya sido posible la asignación de memoria para el nuevo paquete: a. **Si**:

	i. Asigna el paquete que se ha recibido del puerto al buffer **MSDU** empezando por el byte 5. (0 y 1 dirección del nodo coordinador, 2 y 3 dirección del nodo sensor, 4 es el espacio reservado para la calidad de enlace, siempre será 0 puesto que en el paquete que va de smgi a satd no se mide la calidad del enlace). ii. Asigna a **mpPacket**->**msgType** el tipo de mensaje que se va a enviar, en este caso un **gMcpsDataReq_c**. iii. Se asigna la dirección del nodo destino. iv. Se asigna la dirección del nodo fuente v. Se asigna la dirección de la red tanto de destino como fuente. vi. Se asigna el tipo de direccionamiento que se está utilizando en ambos nodos, en este caso **gAddrModeShort_c** para ambos. vii. Se especifica la longitud del paquete en **mpPacket**->**msgData**.**dataReq**.**msduLength**. viii. Se indica que se requiere **ack**. ix. Se envía el mensaje mediante el método **MSG_Send()**. x. Se pone en 0 la variable **bytes_recibidos** 3. Verifica si **bytes_recibidos** es 0 a. **Si**: i. Agrega el paquete a la cola de paquetes pendientes mediante **TS_SendEvent,** para ser enviado mas tarde.

Sistema de adquisición y transmisión de datos

Nombre de la clase	**SATD.cs**
Nombre del método	clasificar_paq_72
Tipo de método	Sub
Tipo de dato que regresa	Nada
Argumentos de entrada	*paq_72:uint8_t
Mensajes de error	Nada
Clases que utiliza	Nada
Función que lo invoca	App_HandleMcpsInput
Métodos llamados	configurar_RSV(&paq_72[2]) Argumentos Paq_72[]:uint8_t configurar_IMSV(&paq_72[2]) Argumentos Paq_72[]:uint8_t
Descripción	1. Declara la variable **clasificacion** para que sirva de selector en el **switch** que determina la clasificación del paquete. 2. Declara la variable **tipo** para que sirva de selector en el **switch** que determina el tipo del paquete, una vez que se ha seleccionado una clasificación. 3. Declara la variable **respuesta** que servirá para recibir la respuesta de éxito o fracaso en la ejecución de la función que se ejecute dependiendo del caso que se eligió y supone **FRACASO** como respuesta del método que se vaya a invocar. 4. Asigna a la variable **clasificación** el dato que se encuentra en la posición 0 del arreglo. 5. Asigna a la variable **tipo** el dato que se encuentra en la posición 1 del arreglo. 6. Se ejecuta el **switch** con el selector **clasificación,** en cada caso: a. Ejecuta un el switch con el selector **tipo**, en cada caso: i. Ejecuta la función que corresponda a la clasificación y el tipo de paquete que haya llegado, recibiendo la respuesta de éxito o fracaso en la variable **respuesta:** 1. **ÉXITO**: a. Asigna a paq_27, MC en la posición 0. b. Asigna a paq_27, el MC que corresponda. c. Ejecuta el método envía_paq_27, indicando el tamaño del mensaje. 2. **FRACASO**: a. Asigna a paq_27, E en la posición 0. b. Asigna a paq_27, el ME que corresponda. c. Ejecuta el método envía_paq_27, indicando el tamaño del mensaje.

Nombre de la clase	**SATD.cs**
Nombre del método	envia_paq_27
Tipo de método	Sub
Tipo de dato que regresa	Void
Argumentos de entrada	l_paq:uint8_t
Mensajes de error	Nada

Clases que utiliza	Nada
Función que lo invoca	clasificar_paq_72
Métodos llamados	
Descripción	1. Verifica que haya espacio para introducir un nuevo paquete a la cola de paquetes y que **mpPacket** no tenga asignada una dirección de memoria: i. **Si**: solicita una localidad de memoria para alojar el nuevo paquete. 2. Verifica que haya sido posible la asignación de memoria para el nuevo paquete: a. **Si**: i. Copia el paquete **paq_27** al **Msdu**. ii. Se asigna a **mpPacket->msgType** el tipo de mensaje que contiene la información que se quiere enviar, en este caso **gMcpsDataReq_c**. iii. Crea la cabecera del paquete con los datos adquiridos en la asociación (dirección del nodo destino, dirección del nodo fuente, dirección de la red destino, dirección de la red fuente, tipo de direccionamiento que se está manejando, longitud del paquete). iv. Indica que se requiere ACK. v. Asigna al paquete el número de paquete del que se trata. vi. Envía el paquete mediante el método **MSG_Send** a través de la interface MCPS. vii. Asigna NULL a **mpPacket** a fin de que quede preparado para aceptar otro mensaje. viii. Aumenta la cuenta de mensajes en la cola de mensajes pendientes.

Nombre de la clase	**SATD.c**
Nombre del método	configurar_RSV
Tipo de método	Función
Tipo de dato que regresa	uint8_t
Argumentos de entrada	paq_72[]:uint8_t
Mensajes de error	Nada
Clases que utiliza	Nada
Función que lo invoca	clasificar_paq_72
Métodos llamados	

Descripción	1. Asigna a las variables de límites de parámetros los valores correspondientes del paq_72:
	a. LI_T=paq_72[0];
	b. LS_T=paq_72[1];
	c. LI_RC=paq_72[2];
	d. LS_RC=paq_72[3];
	e. LI_O=paq_72[4];
	f. LS_O=paq_72[5];
	g. LI_G=paq_72[6];
	h. LS_G=paq_72[7];
	i. LI_PS=paq_72[8];
	j. LS_PS=paq_72[9];

	k. LI_PD=paq_72[10];
	l. LS_PD=paq_72[11];
	2. Regresa ÉXITO al método del que fue invocado.

Nombre de la clase	**SATD.c**
Nombre del método	configurar_IMSV
Tipo de método	Función
Tipo de dato que regresa	uint8_t
Argumentos de entrada	paq_72[]:uint8_t
Mensajes de error	Nada
Clases que utiliza	Nada
Función que lo invoca	clasificar_paq_72
Métodos llamados	

Descripción	1. Asigna a las variables de límites de parámetros los valores correspondientes del paq_72:
	a. IM_T=paq_72[0];
	b. IM_RC=paq_72[1];
	c. IM_O=paq_72[2];
	d. IM_G=paq_72[3];
	e. IM_PS=paq_72[4];
	f. IM_PD=paq_72[5];
	2. Regresa ÉXITO al método del que fue invocado.

Nombre de la clase	**SATD.c**
Nombre del método	configurar_ADM
Tipo de método	Función
Tipo de dato que regresa	uint8_t
Argumentos de entrada	paq_72[]:uint8_t
Mensajes de error	Nada
Clases que utiliza	Nada
Función que lo invoca	clasificar_paq_72
Métodos llamados	
Descripción	1. Asigna a la bandera de control de activación (**bAct_Desac**) el valor correspondientes del paq_72: a. bAct_Desac=paq_72[0]; 2. Regresa ÉXITO al método del que fue invocado.

Nombre de la clase	**SATD.cs**
Nombre del método	enviar_muestras()
Tipo de método	Función
Tipo de dato que regresa	uint8_t
Argumentos de entrada	Nada
Mensajes de error	Nada
Clases que utiliza	SATD
Función que lo invoca	clasificar_paq_72

Métodos llamados	toma_T(), toma_RC(), toma_O(), toma_G(), tomaPS(), tomaPD(), empaqueta_envia_SV()
Descripción	1. Declara la variable **b_enviar_paq**, que es la bandera que servirá para decidir si es momento de enviar un paquete de signos vitales al **SCPD**. 2. Declara la variable **mc**, suponiendo que el mensaje de confirmación será **MC_SVR** en caso de que un paquete de signos vitales sea enviado. Este valor puede cambiar a **MC_AR** si el paquete de signos vitales ha sido enviado por una alerta. 3. Verifica si el valor de la bandera **bAct_Desac=1**: a. Verifica si **MC_esperado** != **0** y si **cont_MC_esperado** == **X1** b. **Si**: i. Asigna **1** a **b_enviar_paq,** para que el paquete se reenvíe de nuevo puesto que no se ha recibido el mensaje de reconocimiento en cuyo caso **MC_esperado** tendría 0. ii. Verifica si el contador de intentos fallidos **cont_IF** es igual a **3**, esto indicaría que se ha intentado 3 veces el envío del mismo paquete sin recibir confirmación entonces se procede a: 1. Desactivar el nodo poniendo la variable **bAct_Desac** a 0, para no seguir enviando información al **SCPD**. 2. Pone el contador de intentos fallidos **cont_IF** a 0. 3. Pone a **0** la bandera **bEnvia_MC_LSVR** que indica que se debe enviar un mensaje de confirmación **MC_LSVR**, después de recibir un mensaje de confirmación **MC_SVR**, eso se hace porque es sólo cuando se recibió una solicitud de signos vitales en tiempo real que se debe enviar este mensaje de confirmación. Cuando la bandera está en **0** indica que el envío de los signos vitales fue de manera normal. iii. Incrementa el contador **cont_IF** de intentos fallidos en uno iv. Asigna a **mc** lo que tenga **MC_esperado** c. **No**: i. Verifica si el contador del control de alertas es mayor o igual a **X2** ii. **Si:** 1. Invoca la función para evaluar si hay parámetros fuera de rango y enviar una alerta. 2. Verifica si la función regresó **ÉXITO**, en cuyo caso lo que indica es que si existe por lo menos un parámetro que está fuera de rango porque es necesario enviar el paquete en ese momento. 3. **Si**: a. Se asigna **MC_AR** a **mc** b. Se pone la variable **b_enviar_paq** a 1 4. Pone el contador de control de alertas a 0; iii. **No**: 1. Verifica si es tiempo de enviar un signo vital: a. Verifica si **IM_T** es igual **cont_T** i. **Si**: 1. Invoca **toma_T()** y

	asigna el resultado a **var_T**. 2. Asigna **1** a la bandera **b_enviar_paq** 3. Asigna **0** a **cont_T** b. Verifica si **IM_RC** es igual **cont_RC** i. **Si**: 1. Invoca **toma_RC()** y asigna el resultado a **var_RC**. 2. Asigna **1** a la bandera **b_enviar_paq** 3. Asigna **0** a **cont_RC** c. Verifica si **IM_O** es igual **cont_O** i. **Si**: 1. Invoca **toma_O()** y asigna el resultado a **var_O**. 2. Asigna **1** a la bandera **b_enviar_paq** 3. Asigna **0** a **cont_O** d. Verifica si **IM_G** es igual **cont_G** i. **Si**: 1. Invoca **toma_G()** y asigna el resultado a **var_G**. 2. Asigna **1** a la bandera **b_enviar_paq** 3. Asigna **0** a **cont_G** e. Verifica si **IM_PS** es igual **cont_PS** i. **Si**: 1. Invoca **toma_PS()** y asigna el resultado a **var_PS**. 2. Asigna **1** a la bandera **b_enviar_paq** 3. Asigna **0** a **cont_PS** f. Verifica si **IM_PD** es igual **cont_PD** i. **Si**: 1. Invoca **toma_PD()** y asigna el resultado a **var_PD**. 2. Asigna **1** a la bandera **b_enviar_paq** 3. Asigna **0** a **cont_PD** g. Invoca la función **incrementa_contadores** para agregar una unidad de tiempo a todos los contadores que llevan el control de tiempo para el envío de paquetes de signos vitales.

	iv. Verifica el valor de **b_enviar_paq**. Si es **1** 1. Invoca la función **empaqueta_envia_SV**(mc) v. Verifica si **MC_esperado** != 0: vi. **Si**: 1. incrementa en uno **cont_MC_esperado**. Esta condición se hace con el fin de que la variable solo se incremente cuando se está esperando un mensaje de confirmación por parte del **SCPD**, de otra manera este contador se incrementaría siempre y se perdería su control. vii. Lanza el temporizador **TMR_StartTimer** que volverá a contar la cantidad de tiempo para volver a invocar la función **enviar_muestras**. De esta manera tenemos un ciclo que se ejecuta cada intervalo de tiempo dado en el temporizador. (La función **enviar_muestras**, invoca el temporizador **TMR_StartTimer** y el temporizador dispara la función **enviar_muestras**, cuando se cumple el intervalo de tiempo). d. **No**: i. Invoca la función **TMR_StopTimer** pasando como argumento el identificador del **TMR_StarTimer** que dispara la función **enviar_muestras**, para detener el envío de paquetes puesto que se ha desactivado esta opción en la aplicación **SCPD** o esta no responde.

Nombre de la clase	**SATD.c**
Nombre del método	empaqueta_envia_SV
Tipo de método	Función
Tipo de dato que regresa	uint8_t
Argumentos de entrada	mc:uint8_t
Mensajes de error	Nada
Clases que utiliza	Nada
Función que lo invoca	enviar_Muestras()
Métodos llamados	envía_paq_27(l_paq) Argumentos: l_paq:int
Descripción	1. Declara la variable **l_paq** y le asigna 0, esta variable servirá para tener la longitud del paquete al final del método y pasarla como parámetro a la función **Envía_paq_27**, esa longitud es requerida para formar el paquete **MAC 802.15.4.** 2. Asigna el argumento de entrada **mc** a la variable **MC_esperado**, para indicar el tipo de mensaje que se va a enviar. 3. Asigna **0** a **cont_MC_esperado** 4. Asigna **paq_27[0]=**MD, incrementa **l_paq** en 1 5. Compara el valor de **MC_esperado** con **MC_AR**: a. **Igual**: i. Asigna **paq_27[1]=MD_SV** b. Compara el valor de **MC_esperado** con **MC_SVR** i. **igual**: 1. Asigna **paq_27[1]=MD_EA**

	6. incrementa **l_paq** en 1 7. Asigna **paq_27[2]=**var_T, incrementa **l_paq** en 1 8. Asigna **paq_27[3]=**var_RC, incrementa **l_paq** en 1 9. Asigna **paq_27[4]=**var_O, incrementa **l_paq** en 1 10. Asigna **paq_27[5]=**var_G, incrementa **l_paq** en 1 11. Asigna **paq_27[6]=**var_PS, incrementa **l_paq** en 1 12. Asigna **paq_27[7]=**var_PD, incrementa l_**paq** en 1 13. Invoca **envia_paq_27(**l_**paq)** 14. Regresa el valor **EXITO**

Nombre de la clase	**SATD.c**
Nombre del método	Incrementa_contadores
Tipo de método	Función
Tipo de dato que regresa	uint8_t
Argumentos de entrada	Nada
Mensajes de error	Nada
Clases que utiliza	Nada
Función que lo invoca	enviar_muestras()
Métodos llamados	Envía_paq_27(l_paq)
Descripción	1. Incrementa contador de temperatura **cont_T()**. 2. Incrementa contador de ritmo cardiaco **cont_RC()**. 3. Incrementa contador de oxigenación **cont_O()**. 4. Incrementa contador de glucosa **cont_G()**. 5. Incrementa contador de presión sistólica **cont_PS()**. 6. Incrementa contador de presión diastólica **cont_PD()**. 7. Incrementa contador de mensaje esperado **cont_MC_esperado**. 8. Incrementa contador de alertas **cont_Alertas**.

Nombre de la clase	**SATD.c**
Nombre del método	evalua_MC_recibido
Tipo de método	Función
Tipo de dato que regresa	Nada
Argumentos de entrada	mc
Mensajes de error	Nada
Clases que utiliza	Nada
Función que lo invoca	clasificar_paq_72
Métodos llamados	
Descripción	1. Si mc = MC_esperado: a. Asigna **0** a **MC_Esperado** b. Ejecuta un switch con MC como selector: i. Caso MC_SVR ii. Caso MC_AR 1. Asigna **0** a var_T 2. Asigna **0** a var_RC 3. Asigna **0** a var_O 4. Asigna **0** a var_G 5. Asigna **0** a var_PS 6. Asigna **0** a var_PD

	7. Asigna **0** a cont_IF
	NOTA: Con esto se limpia cada variable que contiene el valor de la última muestra tomada para usarla cuando se necesite nuevamente. Si esta función no se ejecuta, el valor de la última muestra permanece en la variable y este puede ser enviado en el momento en que el tiempo de espera de reconocimiento indicado haya transcurrido sin recibir el mensaje de confirmación.
	8. Si la variable **bEnvia_MC_LSVR** es igual a 1:
	a. paq_27[0] =MC;
	b. paq_27[1] =MC_LSVR;
	c. invoca el método envia_paq_27().
	d. Pone la variable **bEnvia_MC_LSVR** a **0**

Nombre de la clase	**SATD.c**
Nombre del método	Atiende_MSI_LSV
Tipo de método	Función
Tipo de dato que regresa	uint8_t
Argumentos de entrada	Nada
Mensajes de error	Nada
Clases que utiliza	Nada
Función que lo invoca	clasificar_paq_72()
Métodos llamados	Ninguno
Descripción	1. Se verifica el valor de cada celda del arreglo correspondiente a los parámetros que se miden (T,RC,O,G,PS,PD), la celda 0 es la temperatura, la 1 el ritmo cardiaco y así sucesivamente. 2. Si el valor dentro de la celda es 1, el dato debe ser transmitido: a. Se procede a verificar que los contadores no han alcanzado el valor del intervalo en ese ciclo de reloj con lo que se evita duplicar la transmisión: i. **No ha alcanzado el valor del intervalo**: El contador correspondiente a ese parámetro se le asigna el valor del intervalo de muestreo, para que el dato sea transmitido inmediatamente. 3. Se pone la bandera **bEnvia_MC_LSVR** a uno para que después de enviados los datos en la función **enviar_muestras**, y recibido el mensaje de confirmación **MC_SVR**, sea enviado el mensaje de confirmación **MC_LSVR**, esto se hace con el fin de que cuando el mensaje de confirmación sea recibido por el **SCPD** se tenga la seguridad de que los datos ya están en la base de datos y se pueda dar lectura inmediata. 4. Si la operación fue realizada con éxito se regresa el mensaje **ÉXITO**, de lo contrario se regresa el mensaje **FRACASO.**

Nombre de la clase	**SATD.c**
Nombre del método	verifica_parametros_fuera_de_rango
Tipo de método	Sub
Tipo de dato que regresa	uint8_t
Argumentos de entrada	Nada
Mensajes de error	Nada

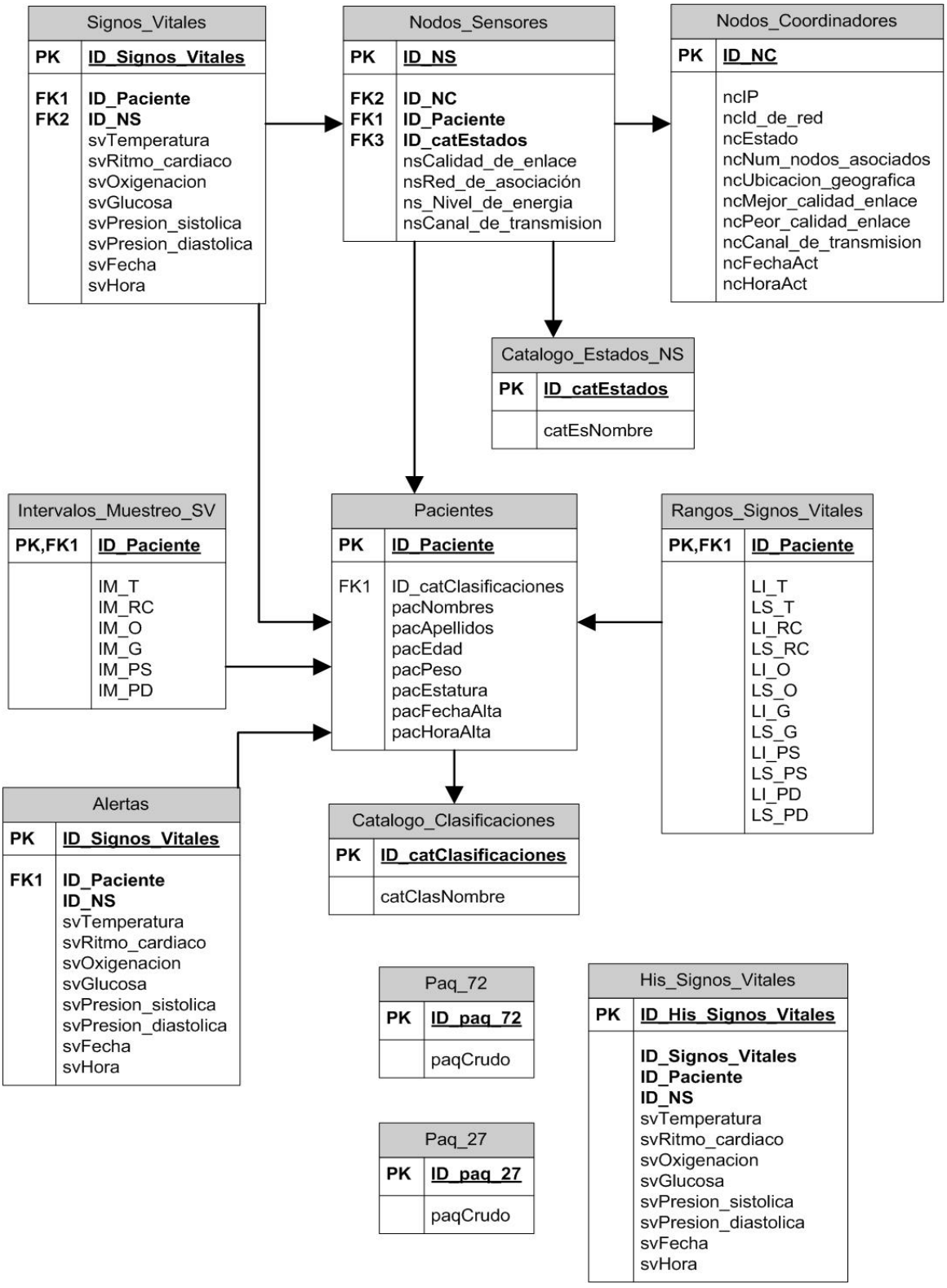
Clases que utiliza	Nada
Función que lo invoca	envia_muestras
Métodos llamados	Nada
Descripción	1. Declara la variable **respuesta** que servirá para regresar el valor **ÉXITO** o **FRACASO**. 2. Declara la bandera **b** suponiendo que no se emitirá una alerta, por lo que se le asigna **0,** este valor cambiará a 1 si un parámetro está fuera de rango. 3. Para cada parámetro: a. Invoca el método **toma_X()**, (donde X se sustituye por la sigla correspondiente al parámetro en cada caso) y Asigna el valor de resultante a la variable **a var_X** (donde X se sustituye por la sigla correspondiente al parámetro en cada caso). b. Verifica si el valor está fuera de rango i. **Si**: 1. Pone la bandera **b** a **1**. 4. Verifica el valor de **b**: a. **1**: i. Hace las siguientes asignaciones: 1. Asigna **0** a **cont_X** (donde X se sustituye por la sigla correspondiente para cada parámetro). ii. Asigna **ÉXITO**, a la variable **respuesta**. b. 0: i. Asigna **FRACASO** a la variable **respuesta**.

Base de datos

Maestría en Ingeniería

Universidad Autónoma de Baja California

Diagrama


Signos_Vitales
PK ID_Signos_Vitales
FK1 ID_Paciente
FK2 ID_NS
svTemperatura
svRitmo_cardiaco
svOxigenacion
svGlucosa
svPresion_sistolica
svPresion_diastolica
svFecha
svHora
Nodos_Sensores
PK ID_NS
FK2 ID_NC
FK1 ID_Paciente
FK3 ID_catEstados
nsCalidad_de_enlace
nsRed_de_asociación
ns_Nivel_de_energia
nsCanal_de_transmision
Nodos_Coordinadores
PK ID_NC
ncIP
ncId_de_red
ncEstado
ncNum_nodos_asociados
ncUbicacion_geografica
ncMejor_calidad_enlace
ncPeor_calidad_enlace
ncCanal_de_transmision
ncFechaAct
ncHoraAct
Catalogo_Estados_NS
PK ID_catEstados
catEsNombre
Intervalos_Muestreo_SV
PK,FK1 ID_Paciente
IM_T
IM_RC
IM_O
IM_G
IM_PS
IM_PD
Pacientes
PK ID_Paciente
FK1 ID_catClasificaciones
pacNombres
pacApellidos
pacEdad
pacPeso
pacEstatura
pacFechaAlta
pacHoraAlta
Rangos_Signos_Vitales
PK,FK1 ID_Paciente
LI_T
LS_T
LI_RC
LS_RC
LI_O
LS_O
LI_G
LS_G
LI_PS
LS_PS
LI_PD
LS_PD
Alertas
PK ID_Signos_Vitales
FK1 ID_Paciente
ID_NS
svTemperatura
svRitmo_cardiaco
svOxigenacion
svGlucosa
svPresion_sistolica
svPresion_diastolica
svFecha
svHora
Catalogo_Clasificaciones
PK ID_catClasificaciones
catClasNombre
Paq_72
PK ID_paq_72
paqCrudo
Paq_27
PK ID_paq_27
paqCrudo
His_Signos_Vitales
PK ID_His_Signos_Vitales
ID_Signos_Vitales
ID_Paciente
ID_NS
svTemperatura
svRitmo_cardiaco
svOxigenacion
svGlucosa
svPresion_sistolica
svPresion_diastolica
svFecha
svHora

Descripción de datos

NC				
Utilización	frmAlta_NC:btnGrabar_Click, AG:Gestiona_Nodos:alta_NC, ICBD:Gestiona_Nodos:gestion_NC			
Clasificación	Complejo			
Tipo	NC_paq			
Tabla	Nodos_Coordinadores			
Datos que lo componen				
Nombre	**Tipo**	**Valor P.**	**Restricciones**	**Descripción**
ID	Int	0	Debe ser un valor entero compuesto por un par de números no mayores a 255, no debe haber un separador entre ambos y la cantidad de dígitos siempre debe ser par.	Identificador del nodo coordinador:
IP	String	0.0.0.0	Se compone de cuatro valores enteros no mayores a 255, separados por un punto.	Dirección IP del nodo coordinador
IR	Int	0	Debe ser un valor entero compuesto por un par de números no mayores a 255, no debe haber un separador entre ambos y la cantidad de dígitos siempre debe ser par.	Dirección de la celda del nodo coordinador
NNA	Byte	0	Cualquier valor byte	Número de nodos asociados del nodo coordinador.
UG	String	“”	Cadena que puede contener cualquier carácter	Ubicación geográfica del nodo coordinador
MCE	Int	-100	Valor entero negativo entre -100 y -15	Representa la mejor calidad de enlace establecida entre el nodo coordinador y uno de los nodos sensores asociados a él.
PCE	Int	-15	Valor entero negativo entre -100 y -15	Representa la peor calidad de enlace establecida entre el nodo coordinador y uno de los nodos sensores asociados a él.
CT	Byte	0	Valor byte que debe estar entre 11 y 26	El canal de transmisión elegido por el nodo coordinador al momento de activarse.
FECHAACT	Datetime	Null	Valor de tipo fecha. No existe como atributo en la clase de la aplicación, debido a que su actualización en la base de datos se hace mediante un trigger que se ejecuta cuando un registro se inserta o se actualiza.	Representa la fecha en la que el registro fue editado por última vez.
HORAACT	string	null	Valor de tipo string. No existe como atributo en la clase de la aplicación, debido a que su actualización en la base de datos se hace mediante un trigger que se ejecuta cuando un registro se inserta o se actualiza.	Representa la fecha en la que el registro fue editado por última vez.

resBD		
Utilización	Todos los métodos que se utilizan en el proceso de interacción con la base de datos	
Clasificación	Complejo	
Tipo	resBD	
Datos que lo componen		
Nombre	**Tipo**	**Descripción**
ds	DataSet	Objeto que puede contener una o más tablas que hayan sido consultadas a la base de datos.
eoe	int	Dato que servirá para indicar si se tuvo éxito en la operación que se realizó, cuando la operación realizada no haya sido una consulta este dato debe tener el número 111 que indica éxito en la operación de lo contrario se sabrá que se produjo un error.
e_datos	String[]	Vector que puede contener una lista de datos que hayan sido capturados con error por el usuario. Que contenga información indica que hubo error en la captura de los datos.

sCadena	
Utilización	Gestiona_Nodos:gestión_NC, gestión_NS, BD:BD
Clasificación	Simple
Tipo	String
Descripción	
Contiene una sentencia válida para ser ejecutada por la base de datos o bien contiene la orden para ejecutar un estore procedure y sus parámetros correspondientes.	

sDatos	
Utilización	AG:Utilerias.valida_NC,
Clasificación	Simple
Tipo	String
Descripción	
Contiene la concatenación del identificador y la IP del nodo coordinador en ese orden y separados por coma ¨,¨ .	

saDatos	
Utilización	AG:Validaciones:valida_ID_NC
Clasificación	Simple
Tipo	String[]
Descripción	
Arreglo unidimensional que contiene el identificador y la IP del nodo coordinador en ese orden.	

sTabla	
Utilización	BD:BD, AG:Utilierias_frm:evalua_respuesta_bd
Clasificación	Simple
Tipo	String
Descripción	
Nombre que tendrá la tabla que se crea en memoria dentro de un DataSet, esto para acceder a ella cuando se necesiten los valores de los registros que contiene.	

NS				
Utilización	AG:Gestiona_Nodos:alta_NS, AG:Gestiona_Nodos:edita_NS, ICBD:Gestiona_Nodos:gestion_NS			
Clasificación	Complejo			
Tipo	NS_paq			
Tabla	Nodos_Sensores			
Datos que lo componen				
Nombre	**Tipo**	**Valor P.**	**Restricciones**	**Descripción**
ID	int	0	Debe ser un valor entero compuesto por un par de números no mayores a 255, no debe haber un separador entre ambos y la cantidad de dígitos siempre debe ser par.	Identificador del nodo sensor:
ID_NC	int	0	Definido en NC.	Definido en NC
ID_Paciente	Int	0	Definido en Paciente.	Definido en Paciente
ID_catEstados	int	0	Definido en cat_estados	Definido en cat_estados.
CE	Int	-100	Valor entero negativo entre -100 y -15	Representa la calidad de enlace establecida entre el nodo coordinador y el nodo sensor.
IR	Int	-15	Valor entero, está compuesto por dos números donde el valor de cada número no debe exceder 255.	Representa la dirección de red que tiene el nodo coordinador al cual se encontraba asociado la última vez que reportó información.
NE	Int	0	Valor entero entre 0 y 100.	Representa el porcentaje de energía que tienen las baterías del nodo.
CT	byte	0	Valor byte que debe estar entre 11 y 26	El canal de transmisión que le asignó el nodo coordinador al que se encuentra asociado.
FECHA	Datetime	Null	Valor de tipo fecha. No existe como atributo en la clase de la aplicación, debido a que este campo en la base de datos se actualiza mediante una función que se ejecuta cuando un el registro se inserta.	Representa la fecha en la que el registro fue insertado.
HORA	string	null	Valor de tipo string. No existe como atributo en la clase de la aplicación, debido a que este campo en la base de datos se actualiza mediante una función que se ejecuta cuando un el registro se inserta.	Representa la fecha en la que el registro fue insertado.

error	
Utilización	AG:Utilierias_frm:evalua_respuesta_bd, Utilerias_frm:errores_bd
Clasificación	Simple
Tipo	Int

Descripción
Valor regresado por la base de datos cuando se produjo un error o bien el número entero 111 que significa ausencia de error al realizar una transacción con la base de datos.

byTipo	
Utilización	AG:Gestiona_Nodos, AG:Gestiona_Nodos, AG:Gestiona_Pacientes
Clasificación	Simple
Tipo	Byte
Descripción	
Este dato se utiliza para pasar un argumento a los "store procedure" que les indicará el tipo de acción que deben realizar. Los valores que puede tomar con sus respectivos significados son: • 1 = SELECT • 2 = INSERT • 3 = UPDATE • 4 = DELETE	

Pacientes				
Utilización	AG:Gestiona_Pacientes, ICBD:Gestiona_Pacientes			
Clasificación	Complejo			
Tipo	Pac_paq			
Tabla	Pacientes			
Datos que lo componen				
Nombre	**Tipo**	**Valor P.**	**Restricciones**	**Descripción**
ID_Paciente	int	0	Valor entero de tamaño int32.	Identificador del paciente.
pacNombre	string	Null	Puede contener cualquier caracter.	Se compone del nombre o de los nombres que tenga un paciente
pacApellidos	string	Null	Puede contener cualquier caracter.	Se compone del o los apellidos que el paciente tenga
pacEdad	int	0	Debe ser un valor entero de máximo 3 dígitos, no se admiten letras u otro tipo de caracteres.	Especifica la edad en años del paciente que será monitorizado.
pacPeso	double	0.0	Debe ser un valor de tipo real, por lo que el único carácter valido que no es numérico es el punto (.).	Representa el peso del paciente en kilogramos.
pacEstatura	Int	0	Debe ser un valor numérico entero que tiene como mínimo 1 dígito y como máximo 3 dígitos.	Representa la estatura del paciente en centímetros.
ID_Clasificaciones	Int	0	Definido en Clasificaciones.	Es el identificador del catálogo de clasificaciones en las que un paciente puede estar.
pacFechaAlta	datetime	01-01-1900	Fecha válida entre 01-01-1900 y 01-01-2100	Expresa la fecha en la que el paciente fue dado de alta para empezar a monitorizar sus signos vitales.
pacHoraAlta	string	Null	Hora válida con formato hh:mm	Expresa la hora en la que el paciente fue dado de alta para

				empezar a monitorizar sus signos vitales.

cat_Clasificaciones				
Utilización	AG:Gestiona_Pacientes, ICBD:Gestiona_Pacientes			
Clasificación	Complejo			
Tipo	clasific_paq			
Tabla	Catalogo_Clasificaciones			
Datos que lo componen				
Nombre	**Tipo**	**Valor P.**	**Restricciones**	**Descripción**
ID_catClasificaciones	int	0	Valor entero de tamaño int32.	Identificador de la clasificación.
catClasNombre	string	Null	Puede contener cualquier caracter.	Representa el nombre de la clasificación en la que puede estar diagnosticado un paciente.

SV				
Utilización	AG:Gestiona_Pacientes:elimina			
Clasificación	Complejo			
Tipo	SV_paq			
Tabla	Signos_Vitales			
Datos que lo componen				
Nombre	**Tipo**	**Valor P.**	**Restricciones**	**Descripción**
ID_Paciente	Int	0	Definido en Pacientes.	Identificador del Paciente.
ID_NS	Int	0	Definido en NS.	Identificador del NS.
svTemperatura	float	0.0	Valor de tipo real. No admite más caracteres que el punto decimal. Debe tener el formato ##.#	Representa la temperatura de un paciente tomada por el NS.
svRitmo_cardiaco	Int	0	Valor de tipo entero. No admite más caracteres que dígitos.	Representa la cantidad de latidos del corazón en un minuto, tomado en determinado momento por NS.
svOxigenacion	Int	0	Valor de tipo entero. No admite más caracteres que dígitos.	Representa el índice de oxigeno que tiene el paciente, tomado en determinado momento por el NS.
svGlucosa	Int	0	Valor de tipo entero. No admite más caracteres que dígitos.	Representa el índice de glucosa en la sangre del paciente, tomado en determinado momento por el NS.
svPresion_sistolica	Int	0	Valor de tipo entero. No admite más caracteres que dígitos.	Representa la presión sistólica del paciente, tomada en determinado momento por el NS.
svPresion_diastolica	Int	0	Valor de tipo entero. No admite más	Representa la presión diastólica

			caracteres que dígitos.	del paciente, tomada en determinado momento por el NS.
svFecha	date	0	Valor de tipo datatime. Se genera en la base de datos mediante una rutina al hacerse la inserción de un registro.	Representa la fecha en la que los signos vitales fueron tomados y enviados a la base de datos.
svHora	String	0	Valor de tipo string. Se genera a partir de la fecha, mediante una rutina en la base de datos.	Representa la hora en la que los signos vitales fueron tomados y enviados a la base de datos.

RSV				
Utilización	AG: Gestiona_RSV:graba_RSV(),ICBD:Gestiona_NC:consulta_NC , ICRS:RS232:empacar(), ICBD:Gestiona_RSV:graba_RSV			
Clasificación	Complejo			
Tipo	RSV_paq			
Tabla	Rangos_Signos_Vitales			
Datos que lo componen				
Nombre	**Tipo**	**Valor P.**	**Restricciones**	**Descripción**
ID_Paciente	Int	0	Definido en Pacientes.	Identificador del Paciente.
ID_NC	int	0	Definido en NC.	Identificador del NC.
ID_NS	Int	0	Definido en NS.	Identificador del NS.
T_SCNAD	int	0	Valor numérico entero de longitud 3.	Representa el tipo de mensaje que se envía o se recibe de la red sensores. De los tres dígitos el primero representa la clasificación del mensaje y el resto (1 o 2 dígitos) representa el mensaje. Los mensajes que existen están descritos en el documento: "Taxonomía SMGI"
LI_T	int	0	Valor de tipo entero, de dos dígitos (la temperatura normal está entre 36 y 37)	Valor que se considera como el límite inferior de temperatura que puede tener un paciente para ser considerado estable.
LS_T	int	0	Valor de tipo entero, de dos dígitos (la temperatura normal está entre 36 y 37)	Valor que se considera como el límite superior de temperatura que puede tener un paciente para ser considerado estable.
LI_RC	int	0	Valor de tipo entero de dos o 3 dígitos (los latidos de una persona adulta normal están entre 60 y 100 por minuto, los de un atleta de alto rendimiento pueden estar hasta 40 por minuto)	Valor que se considera como el límite inferior de palpitaciones por minuto que puede tener un paciente para ser considerado estable.
LS_RC	int	0	Valor de tipo entero de dos o 3 dígitos (los latidos de una persona adulta	Valor que se considera como el límite superior de palpitaciones

			normal están entre 60 y 100 por minuto, los de un atleta de alto rendimiento pueden estar hasta 40 por minuto)	por minuto que puede tener un paciente para ser considerado estable.
LI_O	int	0	Valor de tipo entero uno o dos dígitos (Los niveles normales en un adulto, están entre 12 y 14 kpa).	Valor que se considera como el límite inferior de oxigeno en la sangre que debe tener un paciente para ser considerado estable.
LS_O	int	0	Valor de tipo entero uno o dos dígitos (Los niveles normales en un adulto, están entre 12 y 14 kpa).	Valor que se considera como el límite superior de oxigeno en la sangre que debe tener un paciente para ser considerado estable.
LI_G	int	0	Valor de tipo entero de dos o tres dígitos (Los niveles normales en un adulto, están entre 70 y 110 mg/dl).	Valor que se considera como el límite inferior de oxigeno en la sangre que debe tener un paciente para ser considerado estable.
LS_G	int	0	Valor de tipo entero de dos o tres dígitos (Los niveles normales en un adulto, están entre 70 y 110 mg/dl).	Valor que se considera como el límite superior de oxigeno en la sangre que debe tener un paciente para ser considerado estable.
LI_PS	int	0	Valor de tipo entero de dos o tres dígitos (Los niveles normales en un adulto, están entre 90 y 120 mmHg).	Valor que se considera como el límite inferior de presión sistólica que debe tener un paciente para ser considerado estable.
LS_PS	int	0	Valor de tipo entero de dos o tres dígitos (Los niveles normales en un adulto, están entre 90 y 120 mmHg).	Valor que se considera como el límite superior de presión sistólica que debe tener un paciente para ser considerado estable.
LI_PD	int	0	Valor de tipo entero de dos o tres dígitos (Los niveles normales en un adulto, están entre 80 y 60 mmHg).	Valor que se considera como el límite inferior de presión diastólica que debe tener un paciente para ser considerado estable.
LS_PD	int	0	Valor de tipo entero de dos o tres dígitos (Los niveles normales en un adulto, están entre 80 y 60 mmHg).	Valor que se considera como el límite superior de presión diastólica que debe tener un paciente para ser considerado estable.

SCNAD_paq (paq_72 y paq_27)	
Utilización	ICRS:Configuraracion, ICRS:RS232, SCNS:Comunicacion
Clasificación	Complejo
Tipo	String
Tabla	Paq_72, paq_27
Datos que lo componen	

Nombre	Tipo	Valor P.	Restricciones	Descripción
I_P	Sbyte	:		Es el carácter que indica el inicio y fin de paquete, se encuentra en el byte 0 y 31 respectivamente.
ID_NC	Sbyte	0	Cadena compuesta de dos caracteres ASCII.	Son dos caracteres que representan el identificador del nodo coordinador, ocupan el byte 1 y 2 del paquete.
ID_NS	Sbyte	0	Cadena compuesta de dos caracteres ASCII.	Son dos caracteres que representan el identificador del nodo sensor al que se dirige el mensaje, ocupan el byte 3 y 4 del paquete.
LQI	Sbyte	0	Valor entero negativo entre -100 y -15.	Este valor es la calidad de la señal entre el NC y NS, estará en el byte 5 del paquete.
CL_MSG	Sbyte	0	Puede tomar un valor entre 0 y 4, establecido en formato ASCII	Representa la clasificación del mensaje que se envía o recibe de la red de sensores. Ocupa el byte 6 del paquete.
T_MSG	Sbyte	0	Valor de tipo sbyte restringido a 2 dígitos, establecido en formato ASCII.	Representa el tipo de mensaje que se envía o recibe de la red de sensores. Ocupa el byte 7 del paquete.
MSG	sbyte	0	Cadena de caracteres ASCII con una longitud de 20 bytes	Es el cuerpo del mensaje que ocupará hasta 24 bytes a partir del byte 8, los bytes que no sean usados estarán ocupados con el carácter NUL (0 en decimal). La lista y descripción de mensajes así como su contenido se encuentran en el documento "Taxonomía SMGI"

control_tiempo	
Utilización	ICBD:Gestiona_Msg.
Clasificación	Simple
Tipo	Bool
Descripción	
• Variable tipo booleana utilizada dentro del método verifica_MC para asegurar que la consulta a la base de datos se realizará únicamente cuando su valor sea verdadero, esto sucede cada dos segundos.	

MCE

Utilización	ICBD:Gestiona_MCE:consulta, elimina, etc.			
Clasificación	Complejo			
Tipo	MCE_paq			
Tabla	MCE			
Descripción	Paquete que tiene como fin almacenar en la base de datos los mensajes de confirmación y error que regresen como respuesta después de haber enviado un paquete al SCNAD.			
Datos que lo componen				
Nombre	**Tipo**	**Valor P.**	**Restricciones**	**Descripción**
ID_MCE	Int	0	Valor entero que se asigna de manera automática al insertar un registro en la tabla.	Identificador del paquete MCE.
ID_NC	Int	0	Definido en NC.	Identificador del NC.
ID_NS	Int	0.0	Definido en NS.	Identificador del NS.
MCE_Clasificacion	Int	0	Valor de tipo byte.	Representa la clasificación del mensaje. Puede ser 0 (Mensaje de error) ó 1 (Mensaje de confirmación).
MCE_Tipo	int	0	Valor de tipo byte.	Dependiendo de lo que tenga **MCE_clasificación** error o confirmación tomará un valor entre 1 y 17 o entre 1 y 14 respectivamente. Para saber los diferentes tipos de mensajes de error y confirmación referirse al documento “taxonomía”.

iseg	
Utilización	ICBD:Gestiona_Msg:Cada segundo, ICBD:Gestiona_Msg:Verifica_MC.
Clasificación	Simple
Tipo	Int
Descripción	
• Variable de tipo int que sirve para contar los segundos. Cada vez que el temporizador lanza el evento que se activa cuando el intervalo llegó a su límite, la variable iseg se aumenta en uno. La variable sirve para determinar cada cuanto tiempo estará monitorizándose la base de datos para saber si hay mensajes que atender.	

TAM_SCNAD_paq	
Utilización	SCNAD:SCNS.
Clasificación	Simple
Tipo	Int
Descripción	
• Constante que define la longitud de los paquetes que se utilizan en el SCNAD, paq_72 así como paq_27 tienen esta longitud.	

MAXNS	
Utilización	SCNAD:SCNS.

Clasificación	Simple
Tipo	Int
Descripción	
• Constante que define la cantidad máxima de nodos que podrá tener asociado un nodo coordinador.	

maDeviceShortAddress	
Utilización	SCNAD:SCNS.
Clasificación	Simple, arreglo de dos dimensiones
Tipo	Int
Descripción	
• Arreglo que sirve para almacenar las direcciones cortas de los dispositivos asociados al nodo coordinador.	

pMsdu	
Utilización	SCNAD:SCNS.
Clasificación	Simple, arreglo
Tipo	uint8_t
Descripción	
• Es el arreglo que se usa para poner el cuerpo de los mensajes que se transmiten a través del SCNAD, se encuentra dentro de la estructura mcpsToNwkMessage_t.msgData.dataInd de la MAC	

msduLength	
Utilización	SCNAD:SCNS.
Clasificación	Simple
Tipo	uint8_t
Descripción	
• Variable que contiene la longitud del arreglo pMsdu a través del SCNAD, se encuentra dentro de la estructura mcpsToNwkMessage_t.msgData.dataInd de la MAC	

l_paq	
Utilización	SCNAD:SCNS.
Clasificación	Simple
Tipo	uint8_t
Descripción	
• Variable que contiene la longitud del paquete SCNAD_paq, que será enviado del SCNAD al SMGI y con la cual se verificará la integridad del paquete.	

clasificacion	
Utilización	SCNAD:SATD.
Clasificación	Simple
Tipo	uint8_t
Descripción	
• Variable que almacena una de 5 clasificaciones que existen en la taxonomía (E, MC, MSC, MD, MSI).	

tipo	
Utilización	SCNAD:SATD.
Clasificación	Simple
Tipo	uint8_t
Descripción	
• Variable que almacena una el tipo de mensaje según la clasificación a la que pertenezca.	

EXITO	
Utilización	SCNAD:SATD.
Clasificación	Simple
Tipo	#define
Descripción	
• Constante para determinar ÉXITO (111) en cualquier proceso que se ejecute dentro del código SATD.	

FRACASO	
Utilización	SCNAD:SATD.
Clasificación	Simple
Tipo	#define
Descripción	
• Constante para determinar FRACASO (111) en cualquier proceso que se ejecute dentro del código SATD.	

IMSV				
Utilización	AG: Gestiona_IMSV:graba_IMSV(),ICBD:Gestiona_NC:consulta_NC , ICRS:RS232:empacar(), ICBD:Gestiona_IMSV:graba_IMSV			
Clasificación	Complejo			
Tipo	IMSV_paq			
Tabla	Intervalos_Muestreo_SV			
Datos que lo componen				
Nombre	**Tipo**	**Valor P.**	**Restricciones**	**Descripción**
ID_Paciente	Int	0	Definido en Pacientes.	Identificador del Paciente.
ID_NC	int	0	Definido en NC.	Identificador del NC.
ID_NS	Int	0	Definido en NS.	Identificador del NS.
IM_T	int	0	Valor de tipo entero, de dos dígitos.	Intervalo de tiempo que debe transcurrir entre el envío de una muestra y otra.
IM_RC	int	0	Valor de tipo entero, de dos dígitos.	Intervalo de tiempo que debe transcurrir entre el envío de una muestra y otra.
IM_O	int	0	Valor de tipo entero, de dos dígitos.	Intervalo de tiempo que debe transcurrir entre el envío de una muestra y otra.

IM_G	int	0	Valor de tipo entero, de dos dígitos.	Intervalo de tiempo que debe transcurrir entre el envío de una muestra y otra.
IM_PS	int	0	Valor de tipo entero, de dos dígitos.	Intervalo de tiempo que debe transcurrir entre el envío de una muestra y otra.
IM_PD	int	0	Valor de tipo entero, de dos dígitos.	Intervalo de tiempo que debe transcurrir entre el envío de una muestra y otra.

ADM				
Utilización	ICBD:Gestiona_ADM			
Clasificación	Complejo			
Tipo	ADM_paq			
Tabla	Utiliza el campo **idcatEstados** en la tabla **Nodos_Sensores**			
Descripción	Paquete que sirve para cambiar el estatus de monitorización entre Activo e Inactivo.			
Datos que lo componen				
Nombre	**Tipo**	**Valor P.**	**Restricciones**	**Descripción**
ID_Paciente	Int	0	Definido en Pacientes.	Identificador del paquete MCE.
ID_NC	Int	0	Definido en NC.	Identificador del NC.
ID_NS	Int	0.0	Definido en NS.	Identificador del NS.
ACT_DES	Int	0	Valor de tipo int, sólo 1 ó 0.	Representa el valor que se enviará al **NS**, para el que un 1 significa activar monitorización y 0 significa desactivar monitorización.

LSV				
Utilización	ICBD:Gestiona_LSV			
Clasificación	Complejo			
Tipo	LSV_paq			
Tabla				
Descripción	Paquete que sirve para tener la lectura de los signos vitales de determinados pacientes en el momento que se requiera.			
Datos que lo componen				
Nombre	**Tipo**	**Valor P.**	**Restricciones**	**Descripción**
ID_Paciente	Int	0	Definido en Pacientes.	Identificador del paquete LSV.
ID_NC	Int	0	Definido en NC.	Identificador del NC.
ID_NS	Int	0.0	Definido en NS.	Identificador del NS.
T	Int	0	Valor de tipo int, sólo 1 ó 0.	Este es el valor que dirá al nodo sensor si se desea el valor de la

				temperatura. Si el valor es 1 el parámetro es requerido, en caso de ser 0 el parámetro no es requerido.
RC	Int	0	Valor de tipo int, sólo 1 ó 0.	Este es el valor que dirá al nodo sensor si se desea el valor del **ritmo cardiaco**. Si el valor es 1 el parámetro es requerido, en caso de ser 0 el parámetro no es requerido.
O	Int	0	Valor de tipo int, sólo 1 ó 0.	Este es el valor que dirá al nodo sensor si se desea el valor de la **oxigenación**. Si el valor es 1 el parámetro es requerido, en caso de ser 0 el parámetro no es requerido.
G	Int	0	Valor de tipo int, sólo 1 ó 0.	Este es el valor que dirá al nodo sensor si se desea el valor de la **glucosa**. Si el valor es 1 el parámetro es requerido, en caso de ser 0 el parámetro no es requerido.
PS	Int	0	Valor de tipo int, sólo 1 ó 0.	Este es el valor que dirá al nodo sensor si se desea el valor de la **presión sistólica**. Si el valor es 1 el parámetro es requerido, en caso de ser 0 el parámetro no es requerido.
PD	Int	0	Valor de tipo int, sólo 1 ó 0.	Este es el valor que dirá al nodo sensor si se desea el valor de la **presión sistólica**. Si el valor es 1 el parámetro es requerido, en caso de ser 0 el parámetro no es requerido.

SV				
Utilización	BD:Signos_Vitales			
Clasificación	Complejo			
Tipo	Trigger			
Tabla	Alertas			
Datos que lo componen				
Nombre	**Tipo**	**Valor P.**	**Restricciones**	**Descripción**
ID_Paciente	Int	0	Definido en Pacientes.	Identificador del Paciente.
ID_NS	Int	0	Definido en NS.	Identificador del NS.
svTemperatura	Float	0.0	Valor de tipo real. No admite más caracteres que el punto decimal. Debe tener el formato ##.#	Representa la temperatura de un paciente tomada por el NS.
svRitmo_cardiaco	Int	0	Valor de tipo entero. No admite más caracteres que dígitos.	Representa la cantidad de latidos del corazón en un minuto, tomado en determinado momento por NS.

svOxigenacion	Int	0	Valor de tipo entero. No admite más caracteres que dígitos.	Representa el índice de oxigeno que tiene el paciente, tomado en determinado momento por el NS.
svGlucosa	Int	0	Valor de tipo entero. No admite más caracteres que dígitos.	Representa el índice de glucosa en la sangre del paciente, tomado en determinado momento por el NS.
svPresion_sistolica	Int	0	Valor de tipo entero. No admite más caracteres que dígitos.	Representa la presión sistólica del paciente, tomada en determinado momento por el NS.
svPresion_diastolica	Int	0	Valor de tipo entero. No admite más caracteres que dígitos.	Representa la presión diastólica del paciente, tomada en determinado momento por el NS.
svFecha	Date	0	Valor de tipo datatime. Se genera en la base de datos mediante una rutina al hacerse la inserción de un registro.	Representa la fecha en la que los signos vitales fueron tomados y enviados a la base de datos.
svHora	String	0	Valor de tipo string. Se genera a partir de la fecha, mediante una rutina en la base de datos.	Representa la hora en la que los signos vitales fueron tomados y enviados a la base de datos.
NOTA: Esta tabla se llena de registros de alertas cuando los signos vitales están fuera de los rangos establecidos en la tabla **RSV** para cada paciente, los registros se insertan mediante una rutina que se dispara al momento de una inserción en la tabla Signos_Vitales, los registros se eliminan y se consultan mediante el procedimiento almacenado: **gestiona_alertas**				