SISTEMA DE MONITOREO EN TIEMPO REAL PARA PACIENTES MERIS

ANGELA MARÍA VARGAS ARCILA
CARLOS ALBERTO ORJUELA ZÚÑIGA

Documento final de trabajo de grado presentado como requisito para optar al título de Ingeniero en Electrónica y Telecomunicaciones

Director
Mag. HÉCTOR FABIO JARAMILLO ORDÓÑEZ

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Grupo de Ingeniería Telemática
Servicios Avanzados de Telecomunicaciones
Popayán
2009

TABLA DE CONTENIDO

FIGURAS

TABLAS

INTRODUCCIÓN

El proyecto MERIS (Monitorización de Emergencia de víctimas de catástrofes con Redes Inalámbricas de Sensores), a cargo de la Universidad del Cauca, está orientado a la optimización de la asignación de recursos humanos y técnicos durante la atención prehospitalaria en situaciones de emergencia por medio de un sistema que facilite a los cuerpos de socorro el acceso en tiempo real a la información que describe el estado de las víctimas recuperables. Dicho sistema está compuesto principalmente de una o varias redes inalámbricas de sensores encargadas de monitorear automática y continuamente el estado de los pacientes, clasificando cada uno de ellos según el método *triage* utilizado por el personal de emergencia y enviando, a un dispositivo denominado nodo principal, dicha información junto con la posición de cada sensor conectado a un paciente en referencia a este dispositivo.

Este trabajo de grado, reconoce la utilidad de las redes Ad-Hoc en situaciones de emergencia por el hecho de requerir muy poca configuración y permitir un despliegue rápido, por tal razón describe una solución de monitoreo de pacientes MERIS que sea capaz de desplegar la información concentrada en el o los nodos principales antes descritos en una PDA ó computador portátil por medio del establecimiento de una red WiFi Ad-Hoc entre el dispositivo de despliegue y los nodos, utilizando estrictamente restricciones de tiempo real, de esta manera brinda comodidad al socorrista teniendo en cuenta los escenarios donde pueden presentarse emergencias, mejorando el flujo de trabajo y la productividad de todo el personal de salud, y al mismo tiempo ofreciendo una mejor atención y mayor seguridad del paciente sin afectar su situación debido a manipulaciones físicas.

El presente documento está compuesto de cinco capítulos y 3 anexos de la siguiente manera:

CAPÍTULO I: Describe las generalidades para el desarrollo de aplicaciones en dispositivos móviles con restricciones de tiempo real y estudia sus diferentes sistemas operativos de tiempo real, con el fin de elegir la mejor opción para el proyecto MERIS.

CAPÍTULO II: Describe paso a paso el proceso de adaptación del protocolo de comunicación a utilizar entre el dispositivo móvil y los nodos principales de la red de sensores MERIS. Además, establece las herramientas para la verificación del protocolo.

CAPÍTULO III: Contiene la descripción de la solución desarrollada explicando de manera detallada la arquitectura, análisis, diseño e implementación.

CAPÍTULO IV: Describe las pruebas efectuadas sobre el sistema, los resultados obtenidos y los cambios efectuados al mismo.

CAPÍTULO V: Presenta las conclusiones, recomendaciones y trabajos futuros con relación al desarrollo del proyecto.

ANEXO A: Es la guía de instalación de RTAI, la herramienta seleccionada para la implementación del sistema.

ANEXO B: Describe aspectos generales acerca de RTAI.

ANEXO C: Contiene las funcionalidades de Mobilinux, el sistema operativo recomendado para realizar la aplicación de monitoreo final en una PDA.

1. GENERALIDADES PARA EL DESARROLLO DE APLICACIONES EN DISPOSITIVOS MÓVILES CON RESTRICCIONES DE TIEMPO REAL

1.1 SISTEMA DE TIEMPO REAL

Un sistema de tiempo real es aquel que realiza una correcta planificación del tiempo, de sus procesos y de sus acciones para poder cumplir con los plazos[1] que requieren las aplicaciones; además debe funcionar por largos periodos de tiempo permitiendo obtener un alto grado de confiabilidad. Por consiguiente, este debe operar bajo rigurosas restricciones de tiempo que al ser violadas implica una falla en el sistema. El objetivo principal de dicho sistema, es proporcionar respuestas autónomas[2] y a tiempo, además de reaccionar a los eventos de manera oportuna. [1]

Así, entonces, la clasificación de los sistemas de tiempo real es hecha según la resolución de tiempo[3] y el efecto del incumplimiento de los plazos establecidos. En primera instancia está el *sistema de tiempo real estricto* que cuenta con plazos que deben ser cumplidos a cabalidad y con una resolución de tiempo fina. En este tipo de sistemas, una respuesta correcta significa una respuesta a tiempo, mientras que una respuesta retardada significa una falla del sistema. [2] [3]

En segunda instancia, se establece que el *sistema de tiempo real flexible* es aquel que no tiene resolución de tiempo fina y permite plazos no estrictos. Es decir, si un proceso se retarda no es significativamente importante y las consecuencias de no cumplir un plazo no son graves pero implican un riesgo para la funcionalidad del sistema. [1] [3]

En tercera y última instancia, el *sistema de tiempo real firme* permite fallar algunos plazos y requiere de una resolución de tiempo fina. En estos sistemas, si el plazo termina sin obtener respuesta se descarta el resultado.

Por otro parte, hay otra clasificación que se da según su reacción a sucesos internos ó externos y se divide en dos tipos de sistemas, el primero de ellos es reactivo ó manejado por eventos (*event driven*), cuyo comportamiento es principalmente causado por reacciones a eventos externos más que por las causadas por él mismo, y el segundo sistema es manejado por el tiempo (*time driven*), el cual es impulsado por tareas periódicas asociadas a hechos internos en lugar de eventos aperiódicos. [4]

---

[1] Un plazo hace referencia al hecho de terminar una tarea a un tiempo predeterminado. Es un punto en el tiempo (time-driven) o un intervalo de tiempo en el que una acción del sistema debe ocurrir (event-driven)

[2] Internamente se toman decisiones sobre cómo y cuándo el sistema debe actuar, sin que el usuario lo sepa

[3] Se refiere a la unidad mínima de tiempo que el sistema es capaz de registrar para medir un plazo

1.2 CARACTERÍSTICAS DE UN SISTEMA DE TIEMPO REAL

1.2.1 PUNTUALIDAD

La puntualidad de las acciones es una característica clave para los sistemas de tiempo real y tiene que ver con las limitaciones de tiempo en la duración de las acciones, tal como lo es un plazo. Por esto, para asegurarla lo importante es planificar la temporización modelando el tiempo de ejecución, los plazos, los patrones de llegada, los patrones de sincronización y todo lo necesario para asignar los tiempos de respuesta de un evento y las tareas que implican dicho suceso para cumplir con las condiciones de tiempo real. [4]

Por ejemplo, en algunas situaciones es deseable que el tiempo de respuesta a un evento sea de un tiempo *t* como máximo, es necesario tomar como plazo dicho tiempo y asegurarse de que las tareas que implican la respuesta a dicho suceso sean ejecutadas en tiempos adecuados y precisos para que la acción resultante a todas ellas sea lanzada en o durante el tiempo estimado inicialmente. Un ejemplo claro es ilustrado en la figura 1.

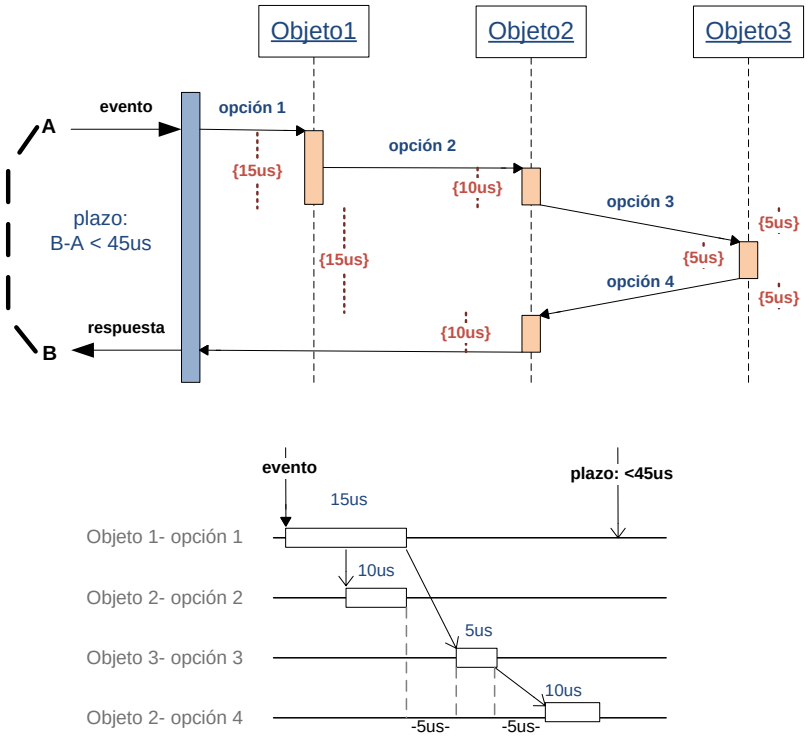


Figura 1. División de tareas en un plazo

1.2.2 CALIDAD DE RESPUESTA

El tiempo transcurrido entre un evento interno o externo y el momento de la respuesta a dicho suceso es denominado tiempo de respuesta (ver figura 2). En los sistemas de

tiempo real la calidad de una respuesta refleja que tan oportuno es el sistema para empezar a dar atención al evento una vez es activado (punto de arranque, ver figura 2) para que el tiempo de respuesta, restringido por un plazo de ejecución, permita asegurar que lo cumplirá. Para lograr que estos sistemas respondan adecuadamente es necesario priorizar las interacciones con los sensores y/o actuadores del dispositivo por encima de las peticiones del usuario. [4]

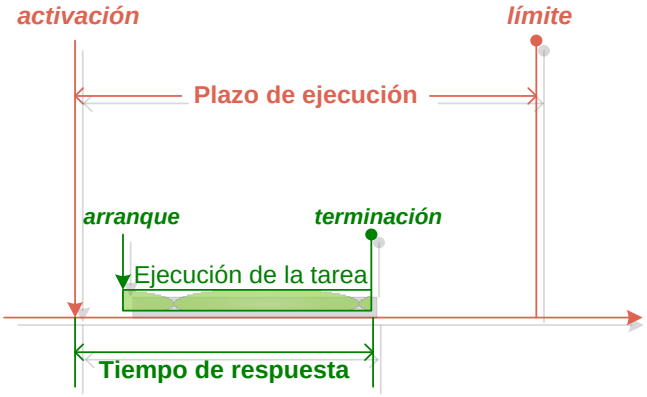


Figura 2. Tiempo de respuesta

1.2.3 CONCURRENCIA

La concurrencia de un sistema hace referencia a la capacidad de ejecución simultánea de múltiples acciones secuenciales. Un sistema de tiempo real realiza una o un pequeño grupo de tareas de alto nivel, para ello requiere de la ejecución de muchas actividades simultáneas de bajo nivel. [1] [4]

Esta propiedad de los sistemas de tiempo real tiene que ver con algunos puntos importantes en las consideraciones de rendimiento y requerimientos funcionales del mismo, tales como, las características de programación de hilos concurrentes, patrones de llegada para los eventos, sincronización de hilos y métodos de control de acceso a recursos compartidos.

1.2.3.1 PLANIFICACIÓN DE TAREAS CONCURRENTES

Existen muchas estrategias que pueden ser utilizadas como control de concurrencia. Las más importantes son las siguientes: [4]

- *Gestión FIFO[4] con ejecución hasta completar la tarea:* puede ser implementada sin acceder necesariamente a servicios provistos por el sistema operativo de tiempo real, solo es útil para aplicaciones muy simples.

- *Cambio sin prioridad de tareas:* esta estrategia está sustentada en que los hilos de ejecución liberen voluntariamente el control al sistema operativo. Una vez este obtiene el control, decide cual tarea ejecutará a continuación. Puede incluir

[4] First-in, first-out. Política de primero en entrar, primero en salir o en este caso, de ser atendido

mecanismos triviales para asegurar que todas las tareas tengan igual tiempo ejecutándose.

- *Planificación round-robin:* establece una fracción de tiempo o *quantum* en el cual da el control sobre los recursos y cambia la tarea en ejecución una vez exceda el tiempo propio asignado para tomar control sobre ellos. Una vez suspendida, la siguiente tarea en la pila de ejecución es ejecutada. Si esta no ha terminado sus procesos, almacena la información para continuar en el siguiente "turno". [5]

- *Ejecutivos cíclicos:* la idea es ejecutar las tareas de manera secuencial en un bucle cerrado "sin fin", cada tarea es mantenida hasta que es completada (figura 3). Son ampliamente utilizados por su simplicidad en la codificación y verificación. [6]

- *Ejecución basada en prioridad:* existen algunos mecanismos en el sistema operativo para asignar diferentes valores de prioridad a los procesos involucrados, con el fin de que si mientras está en atención un proceso determinado y ocurre una interrupción de parte de un proceso con una prioridad mayor, el control del procesador sea dado a este último.

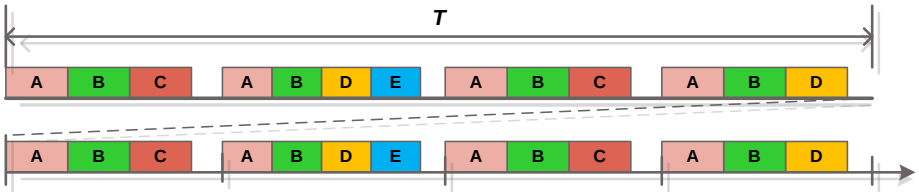


Figura 3. Ejecutivo cíclico

1.2.3.2 PATRONES DE LLEGADA DE EVENTOS

En sistemas reactivos la mayoría de hilos son programados en respuesta a eventos de entrada, por lo que una cuestión a tener en cuenta con ellos son los patrones de llegada de eventos. Pueden ser periódicos o aperiódicos, donde uno periódico es aquel en el que el hilo es reiniciado tras un periodo de tiempo fijo más o menos una pequeña variación. En cuanto al aperiódico, en lo que al sistema concierne, son aparentemente aleatorios. Para que dichos sistemas sean planificados es necesario un análisis probabilístico para determinar las condiciones de ocurrencia y así poder realizarla. La frecuencia de ocurrencia del evento que dispara el proceso debe ser promediado en el tiempo de acuerdo a algunos mecanismos ya definidos. Mayor información puede ser hallada en [1] [4].

1.2.3.3 PATRONES DE ENCUENTRO DE HILOS

Dado que los hilos no son absolutamente independientes, en la mayoría de los casos deben comunicarse para sincronizar el control y compartir recursos. La comunicación entre objetos es realizada por el paso mensajes. Este mecanismo es una abstracción lógica que incluye una variedad de patrones de "encuentro" aunque el más utilizado es la llamada síncrona a funciones, donde los objetos llaman a los métodos definidos en otro objeto operando dentro del mismo hilo de control. Un mecanismo asincrónico es el basado en poner en cola los mensajes. Por otro lado también es utilizado la *espera por el "encuentro"* donde la tarea es detenida aguardando por la otra y su variante temporizada donde espera un tiempo fijo antes de abortar el intento de sincronización.

1.2.3.4 RECURSOS COMPARTIDOS

Un problema común en los sistemas concurrentes es la robustez al usar recursos comunes, dado que es posible acceder a información parcialmente actualizada o errónea si el control de dicho recurso es liberado en el momento menos indicado. La única solución real es asegurar que solo una tarea a la vez pueda acceder al recurso y los mecanismos más correctos incluyen el serializar el acceso a través de semáforos de exclusión mutua[5] o colas.

1.2.4 TOLERANCIA A FALLAS Y SEGURIDAD

Muchos sistemas embebidos tienen altos requerimientos de disponibilidad. Algunas aplicaciones típicas incluyen control de fuego, aviación, plantas nucleares y sistemas médicos. Muchos de estos sistemas no solo deben ser confiables sino que también deben ser seguros, ello significa que si fallan deben hacerlo sin causar lesiones o pérdidas de vidas.

Hay muchas aproximaciones para el desarrollo de aplicaciones confiables y seguras, pero todas incluyen redundancia arquitectónica de alguna forma. La aproximación común para capturar y representar esta redundancia es a través de patrones de diseño arquitectónicos.

1.2.5 CORRECTO Y ROBUSTO

Un sistema es correcto cuando hace las cosas bien todo el tiempo. Dicho sistema es *robusto* cuando hace las cosas bien bajo circunstancias no planeadas incluyendo la presencia de fallas de secciones del sistema. Naturalmente, los sistemas correctos y robustos son considerados ideales en cuanto a su funcionamiento, pero conseguirlos en un diseño complejo no es nada trivial. Es necesario tener presente algunas situaciones como el estancamiento, condiciones de carrera (o estado de carrera, como también es conocido) y otras condiciones excepcionales.

[5] Este mecanismo es conocido por algunos autores como MutEx, por su traducción al ingles

1.2.5.1 ESTANCAMIENTO[6]

Es una condición en la cual una tarea espera indefinidamente por una condición que nunca puede ser satisfecha. Es equivalente a un estado sin transiciones de salida o un estado con transiciones de salida basada en eventos que nunca van a ocurrir. [7]

Para que este caso ocurra deben cumplirse las cuatro condiciones siguientes:

1. La tarea debe reclamar control exclusivo sobre recursos compartidos
2. La tarea mantiene recursos mientras espera que otros recursos sean liberados
3. La tarea no puede ser forzada a renunciar a los recursos
4. Exista una condición de espera circular

Por lo que al evitar la ocurrencia de al menos una de ellas es suficiente para evitar el inconveniente.

1.2.5.2 CONDICIONES DE CARRERA

Una condición o estado de carrera ocurre cuando el estado de un recurso depende de factores temporizados que no son universalmente predecibles. Dichas condiciones típicamente ocurren cuando dos o más hilos acceden al mismo recurso compartido con al menos una modificación a dicho recurso, pero la secuencia relativa de sus acciones al recurso no son predecibles. [8]

1.2.6 AMBIENTES CON RECURSOS LIMITADOS

Los desarrolladores de tiempo real a menudo utilizan herramientas residentes en computadores personales y estaciones de trabajo pero sus aplicaciones tienen como objetivo a plataformas computacionales más pequeñas y menos potentes. Esto significa que deben usar herramientas de compilación cruzada que a menudo son más restrictivas que las herramientas de escritorio más utilizadas. Además, las facilidades de HW disponibles en las plataformas objetivo –tales como temporizadores, conversores A/D, y sensores – no pueden ser fácilmente simuladas en una estación de trabajo. Esta discrepancia entre el desarrollo y los ambientes objetivos añade tiempo y esfuerzo para el desarrollador que quiere ejecutar y probar su código. La falta de herramientas sofisticadas de depuración en la mayoría de objetivos pequeños complican las pruebas también. Usualmente los sistemas ni siquiera tienen una pantalla sobre la cual ver mensajes de error o diagnostico.

Frecuentemente, el desarrollador de sistemas de tiempo real -SRT- debe diseñar y escribir SW para HW que aun no existe. Esto crea grandes desafíos porque no es posible verificar lo que el diseñador entiende de cómo debe funcionar el HW. Las pruebas de integración y verificación son más difíciles y largas.

[6] Deadlock, como es conocido en ingles

1.3 SISTEMAS OPERATIVOS DE TIEMPO REAL

La mayoría de los sistemas de tiempo real[7] moderados y complejos usan un RTOS. Las funciones de un RTOS son muy parecidas a un sistema operativo –SO- normal:

- Manejo de la interfaz del HW subyacente
- Planificación y preferencia de tareas
- Manejo de memoria
- Proveer servicios comunes, incluyendo E/S[8] a dispositivos estándar, tales como teclados, pantallas LCD y de video, dispositivos apuntadores e impresoras

Ellos difieren del SO normal en una variedad de maneras, las más importantes son:

- Escalabilidad[9]
- Políticas de planificación
- Soporte para ambientes empotrados, sin discos

La escalabilidad hace a los RTOS ampliamente aplicables tanto a aplicaciones pequeñas, de un solo procesador como a aplicaciones grandes, distribuidas. Los proveedores de RTOS llaman a esto *arquitectura microkernel*, enfatizando el pequeño tamaño del mínimo kernel o núcleo[10].

Los planificadores de tareas comunes no pueden planificar otras tareas hasta que el hilo actual en ejecución explícitamente libere el control, ello puede acabar en la posible terminación de otros hilos en espera. Los RTOS comúnmente proveen preferencias basadas en prioridad[11] para el control de planificación. En esta clase de planificación, las tareas de más alta prioridad siempre "adelantan" a las de más baja prioridad tomando el control cuando la primera esta lista para ejecutarse. La principal preocupación es que el sistema cumpla los plazos computacionales, incluso en el peor de los casos.

1.3.1 SERVICIOS BÁSICOS DE UN KERNEL RTOS

El kernel de un sistema operativo de tiempo real proporciona una capa de abstracción entre el nivel de aplicación y el hardware que oculta a las aplicaciones los detalles hardware del procesador sobre el cual serán ejecutadas (figura 4). De esta manera, el kernel RTOS suministra cinco categorías principales de servicios básicos a las aplicaciones (figura 5). [9]

---

[7] En adelante denominado RTOS por sus siglas en ingles (Real Time Operative System)
[8] Sigla para Entrada/Salida
[9] Los RTOS son estructurados para que solo los componentes necesarios son incluidos en la imagen que se ejecuta
[10] El kernel o núcleo es la parte del sistema operativo que proporciona los servicios básicos a aplicaciones que son ejecutadas por el procesador.
[11] La *prioridad* de una tarea determina cual tarea correrá preferencialmente cuando más de una esta lista para correr. Prioridad es diferente que urgencia (que cerca está un plazo) e importancia (valor del total de la tarea completada sobre la funcionalidad promedio del sistema)

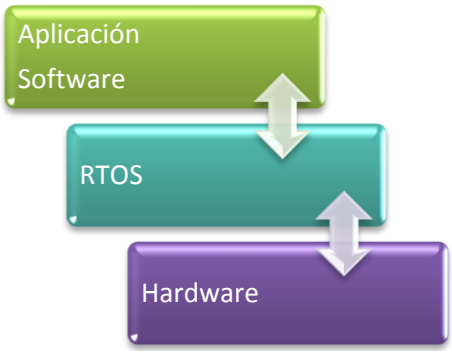


Figura 4. Capa de abstracción entre la aplicación y el hardware.

La categoría de servicios más básica del kernel es “Gestión de Tareas”. Este grupo permite a los desarrolladores diseñar las aplicaciones de forma modular, de manera que cada módulo represente una tarea. Ellos tienen la capacidad de lanzar, crear, planificar, ejecutar, terminar y destruir varias tareas, además autónomamente determinan las tareas para ejecución de acuerdo a sus políticas de planificación.

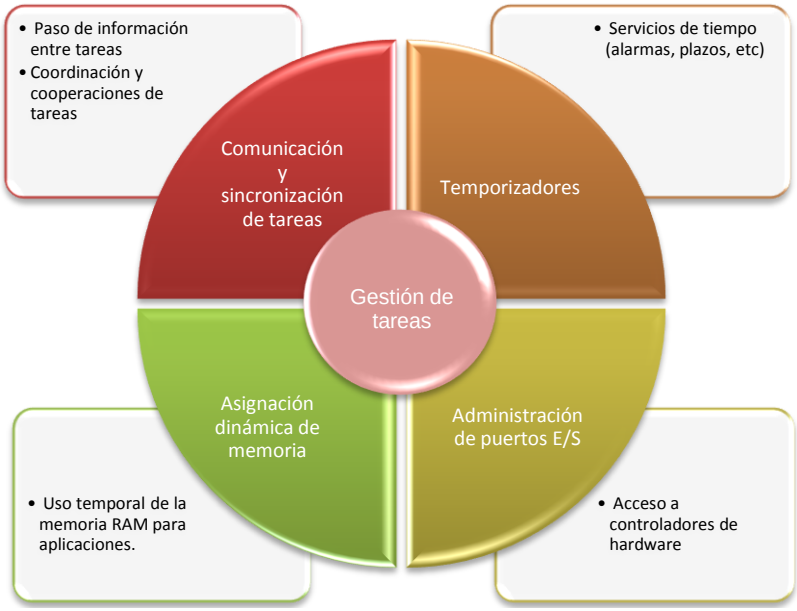


Figura 5. Servicios básicos proporcionados por un kernel RTOS

La segunda categoría de los servicios del kernel es “Comunicación y Sincronización entre Tareas”. Hacen posible el paso de información de una tarea a otra, sin que haya daños en la información. También hacen posible la coordinación y cooperación de tareas. En sistemas RT, los hilos ejecutados asincrónicamente deben encontrarse y sincronizarse en puntos específicos para compartir datos y coordinar el control. Los RTOS típicamente proveen mecanismos de bajo nivel para soportar compartición confiable de datos a través

de distintos mecanismos “pesados” y “livianos”[12]. Sin la ayuda de estos servicios RTOS, las tareas tendrían una comunicación con información dañada o interferirían entre ellas.

Por el hecho de que los sistemas embebidos tienen requerimientos de temporización estrictos, la mayoría de núcleos RTOS proporcionan servicios básicos de tiempo, tales como retrasos ó alarmas. Estos servicios hacen parte de la tercera categoría denominada “Temporizadores” y permiten el uso eficiente del CPU porque tareas en espera de eventos temporizados no necesitan hacer sondeos o ejecutar bucles de espera.

Los servicios de la categoría “Asignación Dinámica de Memoria” toman una parte de la memoria RAM para su uso temporal en aplicaciones. A menudo estas partes de memoria son pasadas de una tarea a otra como una comunicación rápida de grandes cantidades de datos. La mayoría de RTOS proveen los servicios de asignación y retiro de memoria que subyacen a los mecanismos dinámicos de memoria de los lenguajes de alto nivel. Los mecanismos usados en los sistemas de escritorio a menudo no son aplicables a los ambientes de tiempo real.

La categoría “Administración de puertos E/S” refiere a los servicios que proporcionan la organización y accesibilidad a los controladores de los dispositivos hardware del sistema embebido.

El kernel también puede proporcionar otras categorías de servicio relacionadas con la organización del sistema, comunicación en red, gestión de redes, gestión de bases de datos, interfaces gráficas de usuario, etc., pero las anteriores son el conjunto de servicios básicos que debería tener un kernel para sistemas de tiempo real.

1.4 LOS DISPOSITIVOS MÓVILES Y EL TIEMPO REAL

El desarrollo de una aplicación de tiempo real, ya sea en un computador, PDA, celular o en cualquier tipo de dispositivo debe implementar los conceptos anteriores y asegurarse de que sus tiempos sean exactos; por tanto no es práctico diferenciar tiempo real en dispositivos móviles con otros dispositivos, pues el tipo de restricciones de tiempo permitidas, depende de las características de su sistema operativo y su hardware.

El presente trabajo de grado planea utilizar un computador de mano para la implementación de la aplicación, por tanto es de mucha importancia tener en cuenta los siguientes aspectos en su desarrollo:

a. Si utiliza un sistema operativo de tiempo real para dispositivos móviles debe tener las herramientas necesarias para realizar una aplicación de alta calidad en cuanto a las restricciones de tiempo.
b. Si desea trabajar con un sistema operativo común, basado en Linux, las características de tiempo real deben ser aseguradas en el kernel y por tanto modificar su código fuente si es necesario.

[12] Los mecanismos “pesados” proveen encuentros asincrónicos a través de colas de mensajes y los mecanismos livianos están usualmente basados en semáforos que bloquean el recurso y no permiten el acceso hasta que el recurso sea liberado

c. Si por el contrario, no es necesario tener un control total del hardware es posible controlar el tiempo desde un nivel superior, es el caso de RTjava que aunque su rendimiento dependa del sistema operativo, éste no obliga a utilizar un sistema operativo de tiempo real ni necesariamente estar corriendo sobre Linux y la especificación de tiempo real para java (JSR1[13], *Java Specification Request*) ya ha sido incluida en la definición de Sun Microsystems para la plataforma J2ME[14] (*Java 2 Micro Edition*). Desafortunadamente dicha especificación no ha sido implementada por ninguna de las plataformas de desarrollo para móviles más reconocidas.

Para asegurar que el sistema operativo a utilizar es el correcto, se hizo un estudio de los existentes, teniendo en cuenta los recursos que ofrecían para el óptimo desarrollo de una aplicación con características de tiempo real. Teniendo en cuenta los tres ítems anteriores, fueron analizados sistemas operativos para dispositivos móviles tanto de tiempo real como de tiempo promedio:

Acces Linux Platform (ALP): Fue la primera plataforma que incluye un sistema operativo para dispositivos móviles de carácter comercial basada en software de código abierto. Además de utilizar el kernel de Linux, ofrece un módulo para control de alarmas en tiempo real en aplicaciones activas e inactivas y asignación de prioridad a las notificaciones para hardware, servicios ó aplicaciones; estos servicios fueron diseñados para proveer a los desarrolladores un medio consistente de solicitar el disparo de una alarma en un momento determinado.

La compañía ACCESS, propietaria de la plataforma, comercializa a ALP como sistema operativo por medio de la venta de dispositivos y las herramientas de desarrollo y/o código fuente son asequibles de forma gratuita desde la página de la empresa, de ésta forma ACCES contribuye con la ideología de código abierto. [10]

Android: Es una plataforma para dispositivos móviles impulsada por la Open Handset Alliance [11] de la cual hacen parte 47 compañías del sector de la tecnología y las telecomunicaciones, es liderada por Google Inc. Es un proyecto open source que pretende desarrollar una completa pila de aplicaciones, mediador[15] y sistema operativo.

Está basado en el kernel de Linux para los servicios centrales del sistema [12] y permite desarrollar aplicaciones en el lenguaje de programación Java. Posee una máquina virtual propia optimizada llamada Dalvik con un mínimo de memoria necesaria, que es encargada de la gestión de los procesos.

La portabilidad del sistema operativo entorno a diferentes dispositivos ofrece una llamativa alternativa. Es distribuido bajo la licencia Apache Software Foundation versión 2 [13], por ello las compañías comerciales interesadas pueden mantener el carácter libre de la plataforma o restringir la distribución de cualquier modificación realizada, genera un modelo de negocio atractivo para ellas.

[13] Para más información sobre esta especificación, consultar http://community.java.net/jsr/
[14] Para más información dirigirse a la página del fabricante http://java.sun.com/javame/index.jsp
[15] Más conocido como middleware por su traducción en inglés.

ARM Linux Mobile Platform: Empresas como Mozilla, Samsung y Texas Instruments en conjunto con ARM están en el proceso de creación de esta plataforma para dispositivos móviles basada en un entorno Linux con el objetivo de liberarla e implementarla en cualquier dispositivo [14]

Familiar Linux: Es un proyecto totalmente libre, integrado por un grupo de desarrolladores que contribuyen con la creación de lo que ellos llaman "sistema operativo de próxima generación para dispositivos móviles". Actualmente, Familiar está centrado en la producción de una completa y estable distribución de Linux para dispositivos de la serie HP iPAQ y asistentes personales digitales ó PDA, además de acoplarse perfectamente con los entornos gráficos para sistemas operativos Opie[16] y GPE[17]. [15] [16]

LiMo: LiMo Foundation es un consorcio creado con la intención de desarrollar un sistema operativo para dispositivos móviles basado en Linux que sea libre e independiente del hardware. Fundado por Motorola, NEC, NTT DoCoMo, Orange, Panasonic, Samsung y Vodafone en enero de 2007, actualmente cuenta con más de 40 miembros asociados adicionales [17]. Las intenciones de este consorcio están orientadas a generar un sistema operativo estandarizado que elimine la fragmentación del mercado de los dispositivos móviles y así permitir que los esfuerzos de desarrollo sean fácilmente orientados a aplicaciones empresariales, de entretenimiento, etcétera [18]. Esta iniciativa aun está en desarrollo y se espera que pronto libere una versión definitiva del sistema y las herramientas asociadas.

Mobilinux: Es un sistema operativo Linux de tiempo real para dispositivos móviles, desarrollado y comercializado por Montavista. Su arquitectura permite y garantiza el control riguroso y exacto del tiempo, ha sido testeado y mejorado ofreciendo los mejores recursos para el desarrollo de este trabajo de grado pero la empresa creadora de Mobilinux obliga la adquisición de una licencia no gratuita. [19]

Motomagx: Esta plataforma fue desarrollada por Motorola orientada a la línea de dispositivos móviles propios de esta compañía que poseen el sistema operativo Linux. Da un énfasis importante a la interacción del usuario con aplicaciones web. El núcleo del sistema es basado en la distribución kernel versión 2.6.10 de Linux desarrollado por Montavista [20], ofrece algunas mejoras en cuanto a rendimiento, gestión de potencia, entre otras cosas.

Para los desarrolladores ofrece una plataforma llamada MOTODEV Studio que es bastante madura para el lenguaje de programación Java, existen algunas versiones beta orientadas al desarrollo de aplicaciones de Linux nativas [21] y para desarrollo web, pero aún está en fases tempranas por lo que presenta algunas inconsistencias.

[16] Open Palmtop Integrated Environment, es un entorno gráfico para sistemas operativos Linux para dispositivos móviles, de código abierto, creado por la empresa Trolltech.
[17] Es un entorno gráfico para sistemas operativos Linux o basados en UNIX.

Desafortunadamente a pesar de que MOTOMAGX está basado en el kernel de Montavista, no brinda acceso a los módulos definidos en su arquitectura lo que facilitaría manejar interrupciones a nivel de hardware ofreciendo un control confiable de tiempo real sobre las tareas de ejecución.

Openmoko: Proyecto propuesto por las compañías FIC y Openmoko que pretende desarrollar un sistema operativo totalmente libre para teléfonos móviles con aplicaciones no propietarias. Actualmente, todos los avances son de código abierto, permitiendo a los desarrolladores realizarle cambios al sistema, además los esquemas internos de los productos Openmoko ya han sido publicados [22]. Está diseñado para implementarse en la mayoría de dispositivos móviles que soporten Linux o soportados sobre Linux, pero en este momento solo lo soportan algunos dispositivos Motorola, HTC, HP iPAQ, Palm TX, y propios de la empresa que encabeza el proyecto. [23]

En el futuro, planea integrar a la arquitectura del sistema un módulo de tiempo real, pero aún es solo una propuesta tentativa. [24]

Symbian OS: Es un SO completo que inició como propiedad de un numeroso grupo de empresas que tenían como objetivo crear un sistema operativo capaz de superar los alcances de los sistemas de ese entonces. Actualmente pertenece a Nokia, quien pretende liberarlo completamente. [25]

Symbian es suficiente maduro en todos los aspectos, muestra de ello es su capacidad de realizar la comunicaciones con características de tiempo real, de proporcionar mecanismos para la información de eventos al usuario y la gran variedad de protocolos que maneja. [26]

Ubuntu Mobile and Embedded: Esta versión del ya muy conocido sistema operativo es un esfuerzo de la comunidad Ubuntu para desarrollar una plataforma basada en Linux que sea abierta. Este sistema está orientado a un nuevo tipo de dispositivos denominados MID [27] (Mobile Internet Devices, o dispositivos para internet móvil por su traducción al español). Como lo expresa su nombre está enfocada en ofrecer una experiencia al usuario en la navegación bastante usable, además de ofrecer las características comunes de cualquier sistema operativo para dispositivos móviles. Existen pruebas en algunos teléfonos celulares de gama alta (smartphones) con éxito pero este sistema no ofrece grandes facilidades en lo que al desarrollo se refiere.

Windows Mobile: Esta plataforma del gigante del software Microsoft está enfocada en el mercado de los dispositivos móviles, con una amplia gama de aplicaciones básicas para dispositivos móviles basadas en las versiones de escritorio de Windows. El *core* del sistema está basado en Windows CE (para dispositivos embebidos). Este último posee características de tiempo real estricto, pero las herramientas de desarrollo no son gratuitas y tienen un costo bastante alto [28].

La siguiente es una tabla comparativa que reúne las características básicas necesarias para la selección de un sistema operativo para dispositivos móviles que cumpla con los requisitos del trabajo de grado y las necesidades de MERIS.

SISTEMA OPERATIVO	RTOS	MODULO RT[18]	NÚCLEO LINUX	CÓDIGO ABIERTO	LIBRE	PROPIETARIO	COSTO[19]
ALP		✓	✓	✓		✓	✓
Android			✓	✓	✓		
ARM Linux Mobile Platform			✓	✓			
Familiar			✓	✓	✓		
LiMo			✓	✓			
Mobilinux	✓	✓	✓	✓		✓	✓
Motomagx			✓			✓	✓
Open Moko			✓	✓	✓		
Symbian OS		✓				✓	✓
Ubuntu Mobile and Embedded			✓	✓	✓		
Windows Mobile						✓	✓
Windows CE	✓	✓				✓	✓

En desarrollo

Tabla 1. Comparación entre sistemas operativos para dispositivos móviles

Esquematizando lo expuesto en el capítulo, un sistema operativo de tiempo real debe tener un núcleo que ofrezca a las aplicaciones, como mínimo cinco servicios básicos: Gestión de tareas, Temporizadores, Comunicación y sincronización de tareas, Asignación dinámica de memoria y Administración de puertos E/S. De esta manera es posible desarrollar sobre el sistema operativo aplicaciones de tiempo real que satisfagan las condiciones de puntualidad, calidad de respuesta, concurrencia, tolerancia a fallas y seguridad, para obtener una aplicación correcta y robusta.

Por otro lado existen sistemas operativos comunes que no toman en cuenta la priorización de tareas y cumplimiento de plazos, en este caso, las aplicaciones que sí pretenden cumplir dichos requisitos sobre este tipo de sistemas, deben recurrir a herramientas como RTjava que establecen el manejo de temporizadores que garantizan el cumplimiento de plazos en tiempo real suave pero no permiten una comunicación más directa con el hardware en caso de requerirlo. Por el contrario existen también sistemas basados en el

[18] En la arquitectura del sistema operativo existe un módulo para el manejo de restricciones en tiempo real tales como alarmas y notificaciones. Independientemente de si el sistema operativo es o no de tiempo real.
[19] El sistema operativo y/o las herramientas de desarrollo del mismo tienen costo, ya sea por medio de adquisición de licencias, compra de dispositivos u otros.

kernel de Linux, que por el hecho de ser un sistema operativo libre, pueden modificarse sus módulos para crear un mejor manejo de relojes y poder desarrollar aplicaciones de tiempo real .

2. DEFINICIÓN DEL PROTOCOLO DE COMUNICACIÓN BASADO EN EL ESTÁNDAR WIFI CON RESTRICCIONES DE TIEMPO REAL

La naturaleza de los escenarios hacia los que el sistema MERIS va dirigido hace inviable la utilización de medios de transmisión cableados, por lo que es necesario emplear tecnologías inalámbricas de transmisión de datos. En particular, debido a las características de la red de sensores del proyecto MERIS, el estándar WiFi es el utilizado para la comunicación entre el dispositivo móvil y los nodos principales de dicha red.

Partiendo del punto anterior y, teniendo en cuenta que la información que manipula el sistema MERIS es en extremo delicada obligando a que información no relevante, como la de elementos ajenos al sistema intentando registrarse en la red, no afecte con el aumento de la latencia en la transmisión de la información manejada por la red de sensores MERIS hacia la etapa de despliegue en la PDA, o disminuyendo la confiabilidad de dicha información, es necesario implementar un proceso de autorización para el ingreso a la red y un mecanismo de control de acceso, que garantice características de tiempo real para las estaciones pertenecientes a dicha red.

Bajo este panorama, fue realizada una depuración de los diferentes sistemas operativos para dispositivos móviles estudiados en el capítulo anterior con el objetivo de optar por el más adecuado para el proyecto y sujeto a las condiciones del mismo. De esta manera se optó por la utilización de Mobilinux porque es el único sistema operativo, de todos los estudiados, que cumple completamente con las características de conectividad y de tiempo real necesarias para un adecuado desarrollo del actual trabajo de grado. Mobilinux es una plataforma de desarrollo y sistema operativo *open source* 100% Linux nativo para dispositivos móviles. Tiene características de rendimiento en tiempo real porque incluye el manejo de temporizadores de alta resolución y características adicionales como MutEx rápidos, controladores de interrupciones a nivel de hardware estrictos (duros) y suaves, herencia de prioridad a nivel de aplicación y encolado, proveyendo una latencia tan baja como los RTOS de escritorio [29]. Esta plataforma ha sido portada a los procesadores con arquitecturas basadas en ARM9 y ARM11 líderes en el mundo, implementados por Intel, Freescale, Texas Instruments y Samsung, lo que está traducido en la gran experiencia de Montavista, la empresa creadora de Mobilinux, y que asegura la portabilidad del mismo [30]. Para conocer más acerca de este sistema operativo consulte el ANEXO C.

Los demás sistemas operativos descritos en el capítulo 1, no ofrecen las herramientas necesarias, adecuadas o totalmente desarrolladas para realizar aplicaciones de tiempo real. Se analizaron sistemas operativos que a pesar de ser código abierto, aún no ha sido liberado y por esta razón fueron clasificados como proyectos (ARM Linux Mobile Platform, Familiar y Limo). Se estudiaron dos sistemas operativos propiedad de Microsoft, Windows CE y Windows Mobile, el primero presentado como de tiempo real, pero Windows no nos garantiza al igual que Linux una conexión directa con el núcleo y por lo tanto los tiempos y

manejos de relojes no son exactos, el segundo no ofrece restricciones de tiempo real y por lo tanto los dos fueron descartados.

Los tres sistemas operativos totalmente libres Open Moko, Ubuntu Mobile y Android, aparentemente con las mismas características, a pesar de no ser RTOS, por el hecho de ser código abierto y basados en el núcleo de Linux, pueden ser adaptados a las condiciones del proyecto por medio de la creación de módulos que permitan un manejo exacto de relojes; pero Open Moko tiene una implementación limitada en cuanto a los dispositivos, es decir, no es portable, por otro lado Ubuntu Mobile está orientado a la navegación y no hay suficiente soporte para el adecuado desarrollo de aplicaciones; de esta manera queda como opción Android, una plataforma con un sistema operativo ya liberada y testeada en diferentes dispositivos, actualmente existen dispositivos que lo utilizan inyectando seguridad al desarrollo de este proyecto, además cualquier desarrollador puede aplicarle cambios al sistema adecuándolo a sus necesidades y contribuyendo con su mejora. Android, además no tiene ningún costo, es portable y permite una comunicación directa con su kernel facilitando el control de relojes lo cual es un recurso importante para realizar ajustes que tengan como objetivo el óptimo funcionamiento de los requerimientos que establece este trabajo de grado, sin embargo, todo el proceso de adaptación a pesar de ser posible, implica mucho tiempo y el resultado final no sería mejor que un RTOS como lo es Mobilinux[20].

De esta manera fueron analizados cuatro sistemas operativos propietarios que implicaban costos al proyecto. Motomagx, basado en el núcleo de Linux pero no ofrece ningún módulo de tiempo real ni la forma de adecuarlo a las necesidades del proyecto. Symbian OS, que ofrece un módulo de tiempo real pero para obtenerlo debe adquirirse un dispositivo con el sistema operativo ya incluido, lo que significa la poca portabilidad que ofrece. ALP ofrece lo necesario para hacer una buena aplicación en tiempo real, pero no todos sus módulos de tiempo real, son código abierto y esto podría ser un obstáculo en el desarrollo si fuera necesario manipular uno de ellos.

Por todo lo anterior Mobilinux es el sistema operativo recomendado para utilizar en el despliegue de la información de la red de sensores MERIS.

El presente capítulo, pretende describir paso a paso el proceso de adaptación del protocolo de comunicación a utilizar entre el dispositivo móvil y los nodos principales de la red de sensores MERIS. En esta parte, protocolo de comunicación es entendido como el conjunto de elementos necesarios para garantizar el envío y recepción de mensajes en tiempos acotados, es decir, el mecanismo de control de acceso al medio, proceso de autorización y formatos de mensajes.

En primer lugar será definido el escenario de trabajo (número de dispositivos, topología, tipo de conexión, etc.), en segundo lugar y según los recursos ofrecidos por el sistema operativo escogido, será planteado el proceso de autorización. Luego los tipos de protocolos de control de acceso al medio son analizados y estudiados para definir el más

[20] Es preciso aclarar que Android estuvo a prueba y sometido a adaptación aproximadamente 6 meses, pero no se obtuvieron resultados que garantizaran una pronta y total transformación para satisfacer las necesidades de MERIS.

conveniente y las clases existentes en entornos inalámbricos. Finalmente, y con base en la investigación anterior, es planteado el protocolo de comunicación a manejar.

2.1 ESCENARIO

A continuación están definidas las condiciones de la red formada entre los nodos principales de la red de sensores MERIS y el dispositivo encargado del monitoreo en tiempo real de los pacientes.

2.1.1 MODO DE CONEXIÓN

La red inalámbrica de sensores de MERIS está conformada por un conjunto de dispositivos conectados entre sí vía ZigBee encargados de obtener los signos vitales de los pacientes, y que a su vez hacen conexión con un nodo principal (NP ó FFD[21]) que es un dispositivo encargado de recolectar toda la información de dichos sensores como puede apreciarse en la figura 6. El NP también tiene la capacidad de establecer conexiones a través del protocolo WiFi.

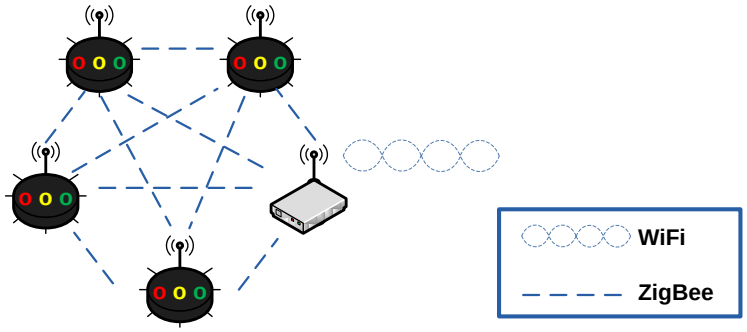


Figura 6. Red inalámbrica de sensores MERIS

Partiendo del punto de que el proyecto MERIS es escalable y por tanto podría ser conformada por varias redes como la anterior, para realizar un módulo de despliegue de la información concentrada por uno o varios nodos principales, solución que pretende desarrollar este trabajo de grado, es necesario analizar los modos de conexión que brinda WiFi, la tecnología de comunicación que comparte el o los NP y el dispositivo de despliegue.

2.1.1.1 MODO INFRAESTRUCTURA

Antes que nada es necesario precisar ciertas bases para establecer una definición clara de los modos de operación de las redes WiFi. Un conjunto de componentes inalámbricos en un grupo de gestión común es denominado como un Conjunto de Servicios Básicos, o BSS[22] por sus siglas en ingles. El área de cubrimiento que tiene un BSS es denominado

[21] Full Function Device. En redes de sensores cumple funcionalidades como coordinador de red y/o nodo enrutador.
[22] Basic Service Set

Área de Servicios Básicos o BSA[23] por sus siglas en ingles, que aunque no es un término muy preciso es lo suficientemente claro para ilustrar su funcionamiento, delimita la zona sobre la cual las estaciones pertenecientes a un mismo BSS pueden comunicarse. [31]

El hecho de que una estación haga parte de un BSS no implica que pueda comunicarse con otra estación del mismo BSS, para esto deben haberse cumplido unos mecanismos específicos. Primero que todo debe hacerse un escaneo para verificar existencia de redes inalámbricas sea conocido o no el SSID[24] o identificador de la red, una vez esto sea hecho debe implementarse la autenticación para establecer la identidad de la estación. Dos tipos de autenticación están definidos en el estándar, sistema abierto o clave compartida. El primer caso consiste en una simple solicitud de autenticación que contiene la identificación de la estación y una respuesta de éxito o fracaso. Este admite que cualquier estación haga parte del sistema. El segundo caso hace uso de una palabra clave que determinará el éxito del mecanismo de autenticación o la denegación del servicio. [32]

La fase de asociación es utilizada por la estación cliente para seleccionar el punto de acceso que le ofrece la mejor calidad en el enlace y la mayor velocidad de transmisión, a la vez que el punto de acceso responde con información de sincronización, carga del mismo, entre otras relevancias. [33]

Esta configuración es la más común y utiliza al punto de acceso como corazón del área de servicio (ver figura 7). Las estaciones miembros pueden comunicarse entre sí directamente pero el control básico y la gestión del área de servicio es desempeñada por el punto de acceso (AP).

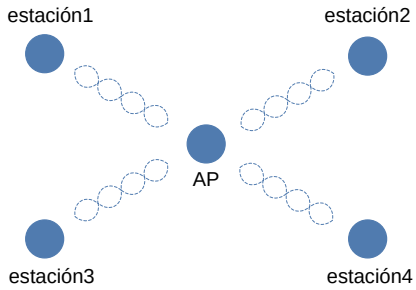


Figura 7. Modo de conexión WiFi en infraestructura

Si la solución final planteara un modo de conexión de este tipo sería necesario el uso de un punto de acceso para permitir la comunicación entre los componentes de la red (figura 8 a) implicando un mayor costo debido a la utilización de otro dispositivo, mayores retardos en la comunicación, disminuyendo la usabilidad teniendo en cuenta su manejo y el tipo de usuario[25] a quien va dirigido el proyecto MERIS. Por otro lado si es deseable mantener este modo de conexión, podría pensarse en la configuración de la PDA como punto de acceso, pero esta característica no es común en los dispositivos móviles y por tanto es descartada (figura 8 b).

[23] Basic Service Area
[24] Service Set Identifier, por sus siglas en ingles
[25] Personal de socorro.

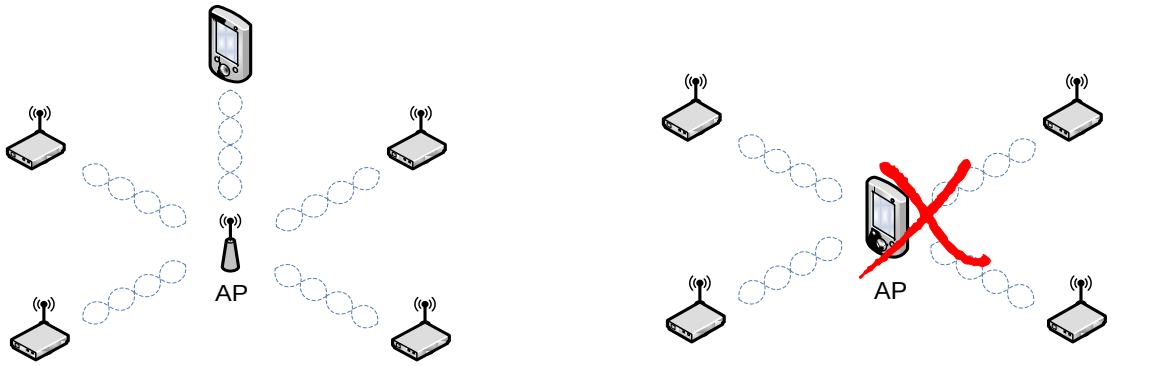


Figura 8. Opciones de conexión en modo infraestructura

2.1.1.2 MODO AD HOC

El modo de operación Ad-Hoc también es conocido como BSS independiente o IBSS[26] por sus siglas en ingles, o par a par[27]. Es el tipo más básico de red 802.11, y no contiene puntos de acceso, únicamente posee estaciones.

Este tipo de instalación es fácil, rápida de configurar y generalmente no se hace una planeación previa para la misma, simplemente bajo una necesidad ocasional de compartir datos entre las estaciones, la conexión es establecida bajo demanda. También puede ser utilizada en entornos de conexiones pequeñas privadas con recursos limitados donde uno de los nodos o estación actúa como enrutador para enviar el tráfico fuera de la red inalámbrica. [32] [33]

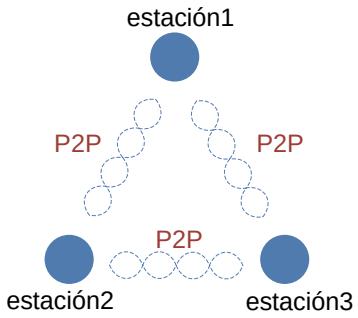


Figura 9. Modo de conexión WiFi en Ad Hoc

La implementación de una red en modo Ad Hoc en la solución que plantea este trabajo de grado no implica más gastos en hardware o retardos causados por dispositivos de enrutamiento, además requiere muy poca configuración y permite un despliegue de red rápido, de esta manera, representa la mejor opción en cuanto al modo de conexión.

[26] Independent Basic Service Set
[27] P2P, peer-to-peer

2.1.2 CONDICIONES DE ESPACIO

Partiendo del establecimiento de la red Ad Hoc entre los NP y la PDA (denominada desde ahora como Red de Etapa de Monitoreo MERIS ó REMM), conviene tener en cuenta características de distancia entre las estaciones. Por tal razón, es necesario recordar que el sistema MERIS está dirigido a apoyar el procedimiento médico "triage prehospitalario"[28] y es por esta razón que los NP estarán en un ambiente abierto, con un máximo de distancia de 50m entre la PDA y cada uno de ellos y con poca movilidad. Además de lo anterior, según el personal encargado del proyecto MERIS, cada NP soporta 60 sensores lo que permite establecer un máximo de 5 NP para la etapa de monitoreo, es decir, 300 pacientes.

Ahora, es necesario tener en cuenta que toda información transmitida en la REMM principalmente fluye entre los NP y la PDA, no entre NP y que el despliegue de la información debe tener características de tiempo real, obligando a que la conexión de los nodos principales con la PDA sea lo más dedicada posible de manera tal que no afecte la información que la aplicación maneje en un momento determinado para que sea lo más actualizada posible, de esta manera es importante ignorar el tráfico de cualquier otro dispositivo conectado a la red Ad Hoc ajeno al proyecto MERIS. Por lo anterior la creación de dicha red es definida desde la PDA, debe existir un mecanismo de autenticación y acceso al medio que asegure la adecuada transmisión de datos proveniente de los nodos principales e ignore la de otros dispositivos.

2.2 MECANISMO DE AUTENTICACIÓN

Teniendo en cuenta que WiFi es un medio compartido, razón por la cual pueden vincularse a la red usuarios involuntarios o que no pertenecen a MERIS, existe la necesidad de restringir el acceso a la REMM para evitar conexiones no deseadas. En este punto, Mobilinux maneja dos tipos de seguridad para sus conexiones inalámbricas, WEP y WPA.

WEP (*Wired Equivalent Privacy*) es el método de cifrado más común para las redes inalámbricas que maneja un nivel de seguridad mínimo y tiene alta vulnerabilidad. WPA (*Wireless Protected Access*), en cambio, proporciona un mejor esquema de cifrado y autenticación [34] [35]. Aunque la seguridad de REMM no es objeto de investigación de este trabajo de grado, estará implementada WPA porque tiene mejores características y es necesario que REMM sea dedicada solamente a los dispositivos que hacen parte del proyecto MERIS y tienen que ver en la etapa de monitoreo del mismo.

[28] El Triage consiste en la evaluación rápida de víctimas para determinar prioridades de atención, en particular, el triage prehospitalario es aplicado en el lugar del evento y permite determinar prioridades de traslado a los centros de atención de salud. [56] [57]

2.3 MECANISMO DE CONTROL DE ACCESO AL MEDIO

Un mecanismo de control de acceso al medio es la forma de gestionar el medio de transmisión, en este caso particular, determina los momentos en que cada NP tiene derecho a transmitir hacia la PDA. Además es el que definirá los tiempos acotados en el envío y recepción de mensajes entre los NP y el dispositivo móvil.

Para proponer el mejor mecanismo es importante tener en cuenta los elementos descritos en los apartados anteriores y resumidos en:

La REMM (ver figura 10):

a. Es una red WiFi con modo de conexión Ad Hoc.
b. La distancia entre nodos y la PDA es de 50m como máximo, característica que no la hace muy extensa.
c. Se manejarán 5 NP máximo.
d. Los NP están prácticamente fijos (poca movilidad).
e. La información que se transmite va únicamente desde los NP hacia la PDA.
f. No es necesaria la comunicación entre NP.
g. Las estaciones que conformen la red de monitoreo, pueden ingresar a la red si y solo si, tienen la clave WPA para ingresar.

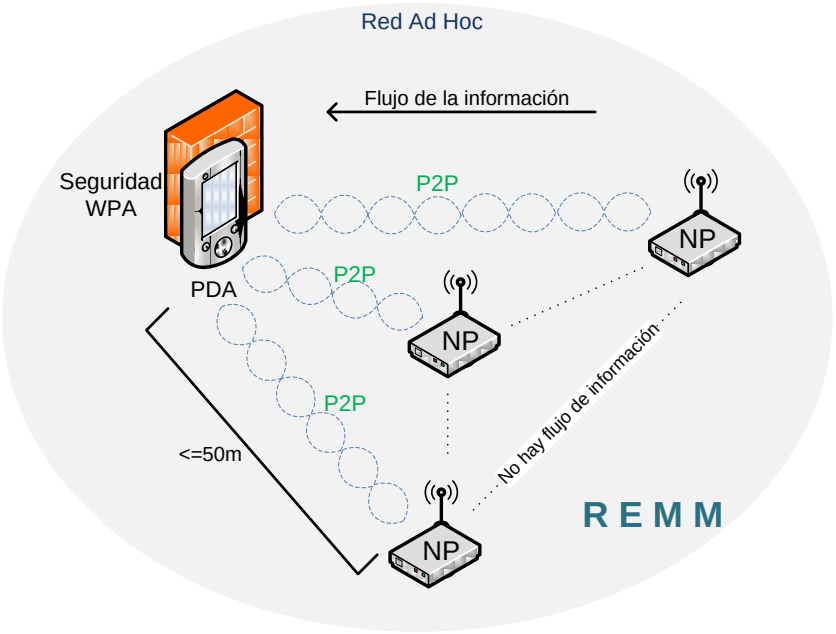


Figura 10. Condiciones de la REMM

Para el proceso de creación de dicho mecanismo fueron estudiadas las diferentes técnicas de acceso con el fin de escoger la más adecuada para los sistemas de tiempo real y posteriormente analizar los protocolos que utilizan esta técnica en redes inalámbricas para finalmente extraer las características más importantes y adecuadas para la REMM, creando un protocolo MAC a esta red.

2.3.1 TÉCNICAS DE ACCESO AL MEDIO

Los protocolos de control de acceso al medio, son basados en diferentes técnicas de acceso según las necesidades de la red y sus características. Generalmente dichas técnicas están divididas en tres clases:

Acceso por contienda: El principio de esta técnica es transmitir cuando sea necesario, es decir, las estaciones pueden transmitir en cualquier momento y no es seguro que una transmisión no va a ser colisionada. Además, el tiempo de transmisión no es regulado, por lo que algunas estaciones transmitirán mayor cantidad de tiempo que otras. [36]

Existen algoritmos de contienda que pretenden evitar las colisiones regulando el acceso al medio entre los contendientes. Estos algoritmos pueden ser de contienda simple (la estación que desee transmitir lo puede hacer y el receptor debe notificar la correcta llegada de la información por medio de un mensaje de control. En caso de transmisiones simultáneas entre dos estaciones, es producida una colisión y el mensaje de control no será enviado a la estación transmisora por lo que ésta retransmite después de un tiempo aleatorio para evitar una nueva colisión), contienda con escucha (a diferencia de la anterior, las estaciones pueden conocer el estado del canal y transmiten cuando creen que está libre ó esperan si detectan que el medio está ocupado) ó contienda con escucha y detección de colisiones (las estaciones escuchan para conocer el estado del canal y en caso de poder transmitir, mantienen la escucha para detectar colisiones, en caso de detectar una, interrumpe la transmisión y espera un tiempo). [36]

En redes inalámbricas, los protocolos que utilizan esta técnica son ALOHA, CSMA/CA, ISMA, DQRUMA, MACA, entre otras. [37]

Acceso por reserva: Está técnica utiliza el principio de multiplexación y divide el acceso al medio por tiempos de acceso, frecuencias de acceso ó códigos de acceso para que todas las estaciones tengan reservado un espacio del canal para transmitir, por ende es una técnica libre de colisiones en la transmisión. La división del acceso por frecuencias y/o códigos no es ajustable a las necesidades de REMM porque no manejan tiempos acotados de transmisión y la implementación de dichas técnicas necesita otro elemento que funcione como módem y transmita hacia la PDA. Por el contrario, la división del acceso por tiempos posibilita la implementación de tiempos acotados porque permite la asignación de tiempos fijos para el acceso al medio, al mismo tiempo es una técnica ejecutada de manera centralizada ó descentralizada y no añade elementos a la red. Una desventaja de la técnica de división por tiempos de acceso, es que si una estación no tiene que transmitir, en el tiempo asignado a la misma no hay transmisión y en este sentido, habrá desperdicio de ancho de banda.

Acceso por turnos: En ésta técnica, el derecho a transmitir está definido por turnos de tal forma que todas las estaciones tienen la misma oportunidad de transmitir, el mismo tiempo de transmisión y solamente una transmite a la vez (no hay colisiones), además permite definir prioridades para que las estaciones con mayor prioridad puedan tomar un turno más largo. [38]

El momento en que un nodo puede transmitir es definido de forma centralizada donde una estación controla las transmisiones, ó de forma distribuida donde cada estación sabe a

través de un testigo si puede o no transmitir. En el primer caso están los protocolos tipo *polling*[29], donde una estación funciona como "nodo maestro" y es la que sondea a las demás estaciones "nodos esclavos" de forma ordenada para preguntar si desean transmitir o no; cuando el nodo maestro envía un mensaje de *polling* a un nodo esclavo y éste tiene información para transmitir, el nodo esclavo envía una respuesta al nodo maestro indicando que ha recibido su mensaje y ha transmitido su información. De esta manera el nodo esclavo hace el sondeo con las demás estaciones de manera ordenada.

En el segundo caso, están los protocolos que utilizan el principio de paso de testigo. El testigo o *token* es una trama especial que circula por la red, que contiene información sobre el estado del medio (ocupado ó libre). Cuando una estación recibe el testigo y al mismo tiempo es un testigo libre, la estación lo marca como ocupado, lo envía nuevamente y empieza a transmitir. Cuando el testigo vuelve a la estación, ésta lo marca como libre y lo envía al siguiente nodo. [39]

La técnica de acceso por turnos puede ser utilizada en topologías en anillo ó bus. Este sistema es muy eficiente cuando la mayoría de estaciones quieren transmitir, de tal forma que el tiempo de transmisión es repartido equitativamente.

A continuación puede apreciarse una tabla comparativa de las técnicas de acceso al medio con las características más relevantes que permitirán analizar el caso específico de REMM y la mejor técnica para su caso.

TÉCNICA	CARACTERÍSTICAS
Contienda	✓ Todas las estaciones pueden transmitir en cualquier momento. ✓ Existen colisiones. Los algoritmos de contienda no las eliminan totalmente. ✓ **No hay tiempo límite de transmisión. No garantiza tiempos de respuesta.** ✓ **No es centralizado.** ✓ **No soporta prioridades.**
Reserva	✓ **Centralizado** ó distribuido. ✓ Libre de colisiones. ✓ Es necesario otro elemento en la REMM para la división del acceso por frecuencias y/o códigos. ✓ Posibilita la implementación de **tiempos acotados.** ✓ **La división de acceso por tiempos es equivalente a la técnica de acceso por turnos.**
Turno	✓ **Todas las estaciones tienen la misma oportunidad de transmitir.** ✓ **Centralizado** ó distribuido. ✓ No hay colisiones. ✓ Puede introducir **plazos para la transmisión de cada estación.** ✓ **Puede soportar prioridades,** en caso de ser necesario.

Tabla 2. Comparación de las técnicas de acceso al medio

Según la Tabla 2, la técnica de acceso por contienda, es muy útil para redes pequeñas donde no existe una carga alta, pero teniendo en cuenta que REMM debe implementar un protocolo de comunicación con tiempo acotados, dicha técnica no garantizaría esta

[29] *Polling* hace referencia a "consulta constante ó sondeo".

condición, además según lo descrito al principio del apartado 2.3, la información fluye desde los NPs hacia la PDA y ésta es la encargada del monitoreo de los mismos, por tal motivo, es necesario que la técnica sea centralizada desde la PDA para asegurar los tiempos de transmisión y las conexiones de los NPs hacia la misma.

Por el contrario, la técnica de acceso por turnos, sí permite una gestión centralizada, que aunque implique una mayor transmisión de mensajes de control, asegura que el dispositivo móvil, como dispositivo central, tenga total control de la REMM. Además, permite limitar en tiempo la transmisión de cada estación y asegura que todas las estaciones tengan la misma oportunidad de transmitir, aspectos importantes en sistemas de tiempo real.

Por otro lado está la técnica de acceso por reserva, descartando la división del acceso al medio por frecuencias y códigos porque implican un mayor gasto económico (otro elemento) y un control de acceso fuera de la PDA. Al contrario de la división de acceso por tiempo que puede satisfacer las necesidades de MERIS. Este último tipo de división es equivalente a la técnica de acceso por turnos.

Siendo así, el protocolo de comunicación para REMM, utilizará una técnica de acceso por turnos lo que lleva a un estudio de los diferentes protocolos de control de acceso al medio que la implementan en entornos inalámbricos.

2.3.2 PROTOCOLOS MAC PARA REDES INALÁMBRICAS AD-HOC

Muchos protocolos han sido propuestos para soportar aplicaciones sin requerimientos de tiempo en redes Ad-Hoc, pero el incremento de la demanda de aplicaciones QoS, en entornos inalámbricos Ad-Hoc, requiere servicios con tiempos acotados. Con base en esto, han surgido diversos protocolos que pretenden llenar este vacío y dar soporte a aplicaciones de tiempo real.

Principalmente sobresale WTRP (*Wireless Token Ring Protocol*) porque a partir de éste derivan muchos otros. WTRP es un protocolo especializado para aplicaciones ejecutadas en una red Ad-Hoc, que garantiza calidad de servicio en términos de latencia limitada y reserva de ancho de banda, ambos cruciales para aplicaciones de tiempo real. WTRP mejora la eficiencia por medio de la reducción del número de retransmisiones debido a colisiones, y es más justo en el sentido de que todas las estaciones usan el canal durante una misma cantidad de tiempo. Las estaciones tienen el turno de transmitir y renuncian a su derecho de transmitir después de una determinada cantidad de tiempo. WTRP es un protocolo distribuido que soporta muchas topologías, ya que no todas las estaciones tienen que estar conectadas entre sí o a una estación central. Este protocolo está diseñado para redes con gran cantidad de nodos, puesto que permite la creación de varios anillos en una misma red según la posición de las estaciones. Utiliza algoritmos para mantener un único sentido de paso de testigo y operación de múltiples anillos cercanos. [40] [41]

A partir de WTRP surgen diversos protocolos tales como IWTRP, diseñado para redes WMAN, que proporciona un método novedoso que permite el paso de múltiples testigos y como resultado son permitidas múltiples transmisiones. EWTRP (*Enhanced Wireless*

Token Ring Protocol) añade a WTRP un mecanismo de hibernación, un mecanismo de contienda y un mecanismo de ajuste dinámico del tiempo de retención del token (THT[30]) según el número de estaciones de la red en cada ciclo. WDTP (*Wireless Dynamic Token Protocol*) para redes móviles Ad-Hoc (MANET) que mejora la adaptabilidad a la topología de la red e incrementa el rendimiento. HFTP (*High Frequency Token Protocol*) que conservan el mismo principio de WTRP pero soporta además dos nuevas operaciones, una de ellas es reasignación de token y la otra ring merging. [42]

De igual manera, existen otros protocolos enfocados al control de acceso de las redes AD-Hoc, como por ejemplo Rether, que es un mecanismo QoS que proporciona garantías de ancho de banda a flujos individuales sobre redes 802.11. Rether no realiza ningún cambio a la capa MAC. Es implementado por encima de la capa de datos y abajo del nivel de red. Su arquitectura es cliente-servidor, donde un servidor inalámbrico Rether (Wireless Rether Server - WRS) es el responsable del control de acceso y coordinación. Es un método centralizado de paso de testigo en el cual un centro es el responsable del paso de testigo y mantenimiento. Soporta movilidad y diferenciación de tráfico.

T-MAH (*Token Passing MAC Protocol for Ad-Hoc Networks*), es un protocolo distribuido, donde las estaciones están organizadas en celdas y cada celda tiene una estación líder; este esquema reduce las posibilidades de colisión y los recursos de la red son utilizados con mayor eficiencia. Pero el tiempo real no es una preocupación para este protocolo.

WICN, es otro protocolo MAC de paso de testigo para redes inalámbricas orientadas al control industrial.

Con el mismo propósito existen protocolos de tiempo real para redes cableadas y su uso en un escenario inalámbrico tiene algunas limitaciones. En consecuencia, han surgido otros que adecúan dichos protocolos a redes inalámbricas. Un ejemplo de ellos es el protocolo TPT (*Token Passing Tree*) derivado del protocolo TTP (*Timed Token Protocol*). Este protocolo soporta aplicaciones QoS en redes Ad-Hoc *indoor* en las que los terminales tienen un espacio limitado para moverse.

De la misma forma existe el protocolo WRT-Ring derivado del protocolo MAC de tiempo para redes cableadas RT-Ring, diseñado para redes con poca movilidad y espacio limitado. Este protocolo utiliza el mecanismo CDMA, el cual permite múltiples transmisiones sin causar colisiones e integra dos tipos de tráfico: best-effort y tiempo real [43]. En particular este tipo de protocolo no es una solución para REMM puesto que el tipo de tráfico manejado en la red es el mismo y no es necesario hacer una diferenciación.

Por otro lado existe también un mecanismo de control por medio de una estación base virtual (*VBS Virtual Base Station*) que utiliza el paso de testigo, denominado *Token Ring Based VBS*. En este mecanismo la VBS censa el medio y envía una pequeña trama, llamada token, por todos los miembros de su *cluster*. La posesión del token garantiza la transmisión. Si un nodo que recibe el token no tiene información para enviar, éste retorna el token a la VBS. Cada estación puede mantener el token de acuerdo al esquema de sondeo usado por la VBS. Si una estación que recibe el token sí tiene información para transmitir, esta aprovecha el token, añade la información que desea transmitir y envía la

[30] Token Holding Time

información al VBS. Mientras la trama de información está circulando en el anillo, no hay token en la red, lo que significa que otras estaciones que esperan transmitir deben esperar, por lo tanto, las colisiones minimizan. VBS, de esta forma soporta calidad servicio en términos de latencia limitada y reserva de ancho de banda. Además, es eficiente en el sentido de que reduce el número de retransmisiones debido a colisiones, y es justo en el sentido de que cada estación tiene un turno para acceder al medio y es forzado a renunciar al derecho de transmitir después de transmitir por un periodo de tiempo específico. Por lo tanto, muchas clases de servicios pueden ser soportados con un agente controlador de admisión para ancho de banda y latencia limitada. [44]

De lo anteriormente dicho, podemos ver que la mayoría de protocolos utilizan un mecanismo basado en el paso de testigo, lo que asegura un acceso al medio más junto en el sentido de que cada estación tiene la misma oportunidad de transmitir y el mismo tiempo para hacerlo. Igualmente, estos protocolos son diseñados para redes con un mayor número de estaciones que REMM y son obligados a utilizar un mecanismo de clasificación de las estaciones en celdas según su posición y/o conexión con los demás nodos. En este sentido, el protocolo MAC para REMM no necesita hacer esta clasificación.

A continuación está descrita la solución propuesta, basada en los aspectos más importantes de los protocolos anteriores y teniendo en cuenta las condiciones particulares de REMM.

2.3.3 PROTOCOLO DE CONTROL DE ACCESO AL MEDIO PARA REMM

Por la naturaleza de las redes Ad-Hoc (medio compartido, no tienen topología fija, pueden existir nodos ocultos, etc.), proponer un protocolo MAC con tiempos acotados de transmisión es un reto, ya que las redes Ad-Hoc no proporcionan ningún tipo de garantías.

En particular, REMM tiene características que la diferencian de una red Ad-Hoc común y es precisamente por la forma o sentido en que siempre va a fluir la información, y por la comunicación innecesaria entre NP. Teniendo en cuenta lo anterior y algunos aspectos importantes de los protocolos descritos anteriormente, un protocolo MAC centralizado y basado en el paso de testigo para la REMM es definido, de ahora en adelante llamado PMR (Procolo MAC para REMM).

En primer lugar, PMR mantiene las características básicas de los protocolos basados en el paso de testigo, pero debido al control centralizado en la PDA, ésta debe crear un anillo virtual según el orden de conexión de los NP, por ejemplo, existen 4 NP conectándose a la red Ad-Hoc, (NP **A**, NP **B**, NP **C** y NP **D**), y el orden de conexión es establecido de la siguiente manera:

NP D → primero
NP B → segundo
NP C → tercero
NP A → cuarto

Con base en el anterior orden, la PDA crea un anillo virtual para asignar un turno de acceso al medio a cada NP, como lo muestra la figura 11.

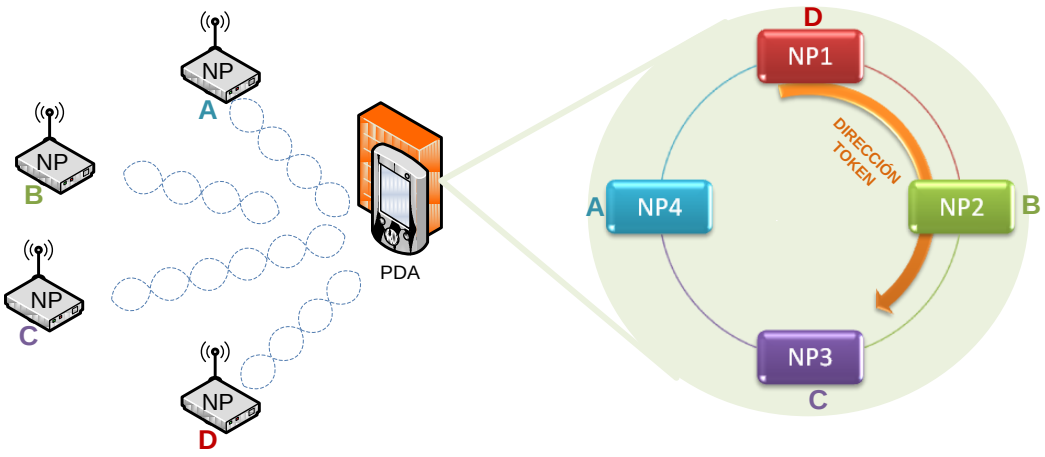


Figura 11. Anillo virtual a partir de un orden de conexión.

El anillo virtual no es estático, es decir, cuando la PDA está en movimiento y sale de cobertura un NP, ésta también sale del anillo, en consecuencia la PDA debe reordenar los turnos de acceso al medio debido a la ausencia de dicho NP. En este caso, una alarma es activada en la PDA indicando la ausencia del nodo para que el usuario final, si desea recibir información del NP ausente, cambie su posición para que el nodo reingrese al anillo virtual y vuelva a tener oportunidad de transmitir. Cuando sucede un reingreso, el NP no vuelve a su posición anterior sino a la última posición en el orden de turno de acceso igualmente sucede para los nuevos ingresos.

Todas las estaciones deben esperar por el token enviado por la PDA, para obtener el derecho de transmisión, y renuncian a este derecho una vez el plazo de transmisión acabe. Este plazo es un tiempo acotado definido según el tiempo de procesamiento de los NP y la PDA, durante el cual el NP que tiene información para transmitir y tiene el token en su poder, envía sus datos hacia la PDA. El tiempo correspondiente al plazo de transmisión es enviado en el token.

El proceso de monitoreo del medio es ejecutado por medio de tres estados (ver figura 12). En primer lugar está un estado “registro”, donde la PDA registra en orden de conexión a los NP y genera el anillo virtual para la transmisión del token. Este registro es realizado cada segundo con el fin de actualizar el anillo virtual.

Una vez los nodos están registrados, la PDA envía el testigo al primer NP (estado “envío testigo”), en éste punto, la PDA inicia un temporizador denominado TEMPO_TOKEN_RECIBIDO que a su vez es inicializado en MAX_TIEMPO_TOKEN_RECIBIDO, si el temporizador expira (es menor que cero), sin que la PDA haya recibido un mensaje que indique la recepción del token, ésta asume que la transmisión no fue exitosa y nuevamente envía el token a dicho nodo (si al reenviar el token ocurre lo mismo, la PDA asume que el NP está fuera de cobertura y por lo tanto lo elimina de su lista haciendo que el NP quede fuera del anillo virtual). Si por el contrario el

NP envía el mensaje de confirmación y tiene algo para transmitir, la PDA le envía un mensaje “ready” y pasa al estado “recepción” donde está dispuesta a recibir todos los datos enviados por el NP en un tiempo acotado definido TIEMPO_ACOTADO. Cuando TIEMPO_ACOTADO termina, la PDA envía un mensaje “ack” indicándole al nodo que ya recibió los datos durante dicho tiempo y por lo tanto debe renunciar a su derecho de transmisión; de esta manera la PDA vuelve al estado “envío testigo” para dar el turno de acceso al medio al siguiente nodo.

Se aclara que el mensaje de confirmación enviado por el NP hacia la PDA indicando la recepción del token está dividido en dos clases; primero puede ser un mensaje “ok” que indica la recepción del token y el deseo de transmitir información, y segundo, un mensaje “cancel” que indica la recepción del token y el no envío de información. En el primer caso el proceso es como el descrito anteriormente, en el segundo caso la PDA no pasará al estado “recepción” sino que enviará el token al siguiente NP y permanecerá en su estado “envío testigo”.

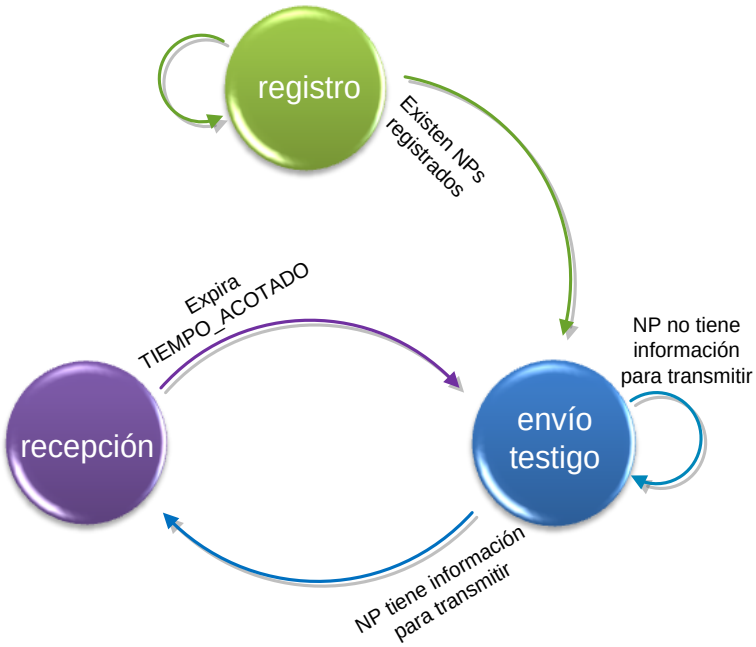


Figura 12. Estados de la PDA.

De la misma manera, los NP pasan por cinco estados (ver figura 13). El estado “desconectado”, es el estado inicial o por defecto de cada nodo cuando está asociandose a la REMM, una vez asociado, pasa automáticamente al estado “ocioso”, lo que indica que está en espera del token para poder transmitir.

Una vez la PDA le pasa al NP el token (estado “testigo”), éste inicia un temporizador denominado TEMPO_EN_ANILLO inicializado en MAX_TIEMPO_EN_ANILLO y es equivalente a 1 segundo por los requerimientos de MERIS, si TEMPO_EN_ANILLO expira, el NP asume que está fuera del anillo porque ha pasado el tiempo máximo que puede utilizar el sistema para conceder el permiso de transmisión a todos los NP de la red y debe volver a conectarse a la REMM.

Para indicarle a la PDA que ha recibido un token, el NP le envía un mensaje de confirmación según el caso, “ok” si tiene información para transmitir ó “cancel” si su caso es el contrario. En el primer caso, la PDA enviará un “ready” indicando que está lista para recibir los datos y en este instante el NP pasa al estado “enviando” donde tiene un plazo determinado por TIEMPO_ACOTADO para transmitir. Cuando dicho tiempo acaba, la PDA envía al NP un “ack” para que el NP termine la transmisión enviando un mensaje “bye” hacia la PDA y volviendo al estado “ocioso” (Es preciso aclarar que TIEMPO_ACOTADO define el máximo tiempo que puede durar la transmisión. Sin embargo, esta duración puede ser menor dependiendo de la cantidad de información a transmitir). En el segundo caso, el NP devuelve el token a la PDA para que ésta lo pase al sucesor del NP en cuestión y pasa al estado ocioso.

Cuando el NP pasa al estado “ocioso”, inicia un temporizador denominado TEMPO_OCIOSO inicializado en MAX_TIEMPO_OCIOSO, dicho tiempo es equivalente al tiempo que transcurre como si todas las estaciones conectadas a REMM tuvieran información para transmitir en el peor de los casos, es decir a una distancia de 50m, si este temporizador expira, la estación es considerada fuera de cobertura de REMM y debe esperar su reconexión.

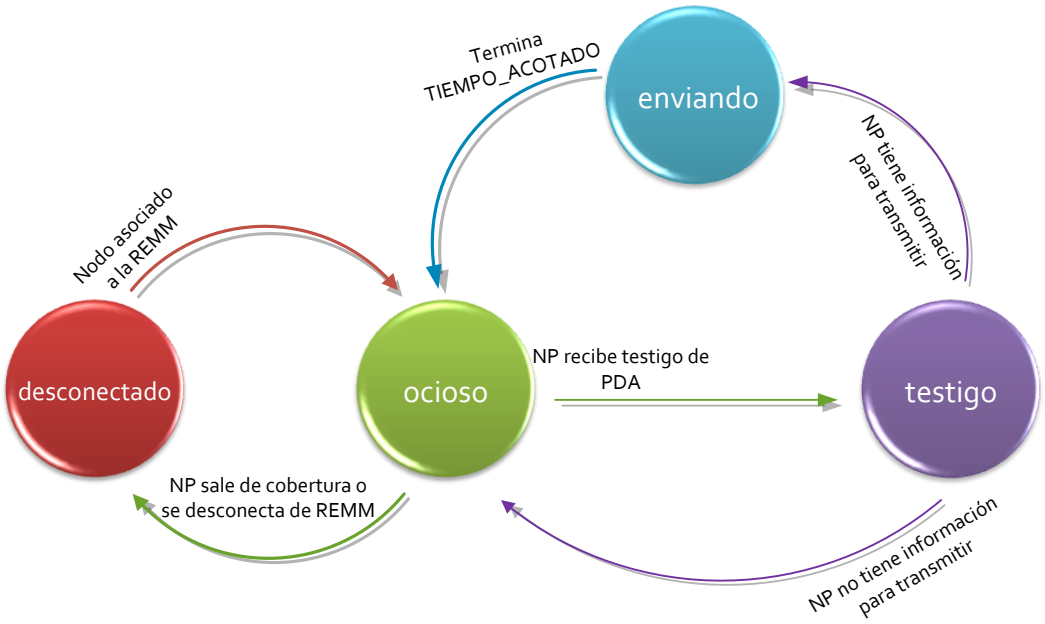


Figura 13. Estados de un NP.

Con lo expuesto anteriormente debe tenerse en cuenta que:

$$TIEMPO\_ACOTADO < MAX\_TIEMPO\_OCIOSO < MAX\_TIEMPO\_EN\_ANILLO$$

En resumen, el proceso de asignación de acceso al medio es realizado como lo describe la Tabla 3. Las flechas sólidas indican el flujo normal de asignación de testigo, las flechas punteadas son los casos que se pueden dar cuando no se cumplen los plazos establecidos o no hay información para enviar. La figura 14 ilustra el paso de mensajes.

PDA	NP
Envía testigo • Inicia el temporizador TEMPO_TOKEN_RECIBIDO. • Si expira el temporizador y no ha llegado un mensaje de confirmación del NP, reenvía el token.	**Recibe testigo** • Inicia el temporizador TEMPO_EN_ANILLO inicializado en 1s. Si expira el NP debe volver a conectarse.
Recibe mensaje de confirmación • Si es “cancel”. Envía testigo al siguiente nodo del anillo. • Si es “ok” envía un “ready” para que el NP transmita e inicia temporizador TEMPO_ACOTADO	**Envía mensaje de confirmación** • Si no tiene información para enviar, confirma con “cancel”. • Si tiene información para enviar, confirma con “ok”.
Envía ready	**Recibe ready**
Recibe información de pacientes • Cuando expira el TIEMPO_ACOTADO, envía un “ack” para que el NP termine su transmisión.	**Empieza la transmisión** • Debe transmitir durante TIEMPO_ACOTADO.
Envía ack	**Recibe ack**
Recibe mensaje “bye” • Está listo para enviar el token al siguiente nodo del anillo	**Envía mensaje “bye”** • Pasa al estado ocioso e inicia el temporizador TEMPO_OCIOSO. Si el temporizador expira, el NP se considera fuera de cobertura de la PDA.

Tabla 3. Proceso de asignación de acceso al medio de REMM

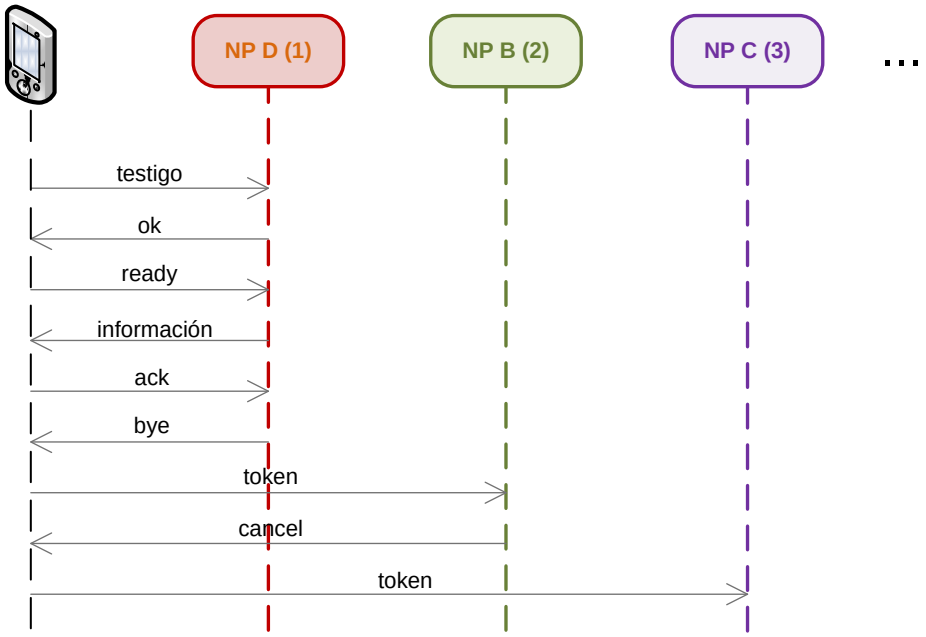


Figura 14. Paso de mensajes entre la PDA y NPs para envío y recepción de testigo.

De esta manera, los mensajes para el envío y recepción del testigo se resumen en:

TOKEN	Para dar el turno de acceso al medio.
OK	Hay información para transmitir.
READY	Listo para recibir.
INFORMACIÓN	Datos hacia la PDA.
ACK	Información recibida.
BYE	Terminación de la transmisión.
CANCEL	No hay información para transmitir.

2.4 LENGUAJE DE COMUNICACIÓN

Para una correcta comunicación entre la PDA y los NP es necesario que los mensajes descritos en el apartado anterior tengan un formato que sea reconocido en todas las estaciones y así tener un lenguaje común en la comunicación. Para esto, fueron establecidos varios tipos de tramas según el mensaje a enviar, teniendo en cuenta que dichas tramas no alterarán el entramado utilizado por el estándar 802.11b, sino que utilizarán el espacio de la trama de datos correspondiente a 2312 bytes [45] para poner las tramas de PMR.

2.4.1 ENCABEZADO DE TRAMAS

Figura 15. Encabezado de tramas

Los mensajes utilizados por el protocolo de comunicación son, el token, “ok”, “cancel”, “ack”, “bye”, e información; por cada uno de ellos será utilizada una trama. La figura 15 muestra el encabezado utilizado por todos los tipos de tramas. En primer lugar, está el campo TC ó Trama de Control, que es un campo de 1 byte que define el tipo de mensaje que está siendo enviando y su valor es de la siguiente manera:

00111001 → Token
01100100 → Mensaje de Confirmación (ok, cancel, ready, ack)
10010101 → Bye
11010100 → Datos

El campo DN indica la Dirección MAC del Nodo que tiene el token y toma importancia solamente cuando la trama es enviada desde el NP hacia la PDA.

La única trama que utiliza solamente el encabezado es la trama que indica la terminación de la transmisión de información desde el NP hacia la PDA o mensaje “bye”.

2.4.2 TIPOS DE TRAMAS

2.4.2.1 TRAMA TOKEN

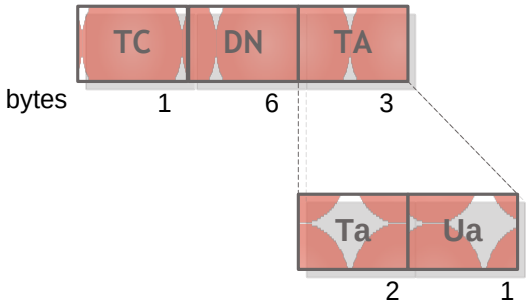


Figura 16. Trama Token

La figura 16 muestra el formato de la trama token donde TA es el Tiempo Acotado que tiene como plazo un NP para transmitir la información de su red de sensores cuando recibe el token y está compuesta de dos campos, Ta indicando el tiempo que puede durar la transmisión y Ua indicando las unidades, éste último campo tiene dos opciones 00000001 para milisegundos ó 00000010 para microsegundos.

2.4.2.2 TRAMA DE MENSAJES DE CONFIRMACIÓN

La figura 17 muestra la trama para los mensajes de confirmación intercambiados entre la PDA y el NP cuando fue transmitido un token.

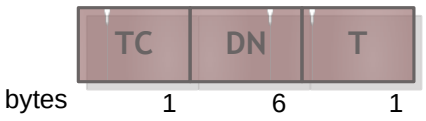


Figura 17. Trama de mensajes de confirmación

La trama de mensajes de confirmación puede ser utilizada para enviar un “ok”, un “cancel”, un “ready” o un “ack”. El campo T es utilizado para determinar qué clase de confirmación está siendo enviada y puede tener los siguientes formatos según el caso:

Confirmaciones	01010101	→	Ok
de NP	01111000	→	Cancel
Confirmaciones	10111110	→	Ack
de PDA	10010110	→	Ready

2.4.2.3 TRAMA DE INFORMACIÓN

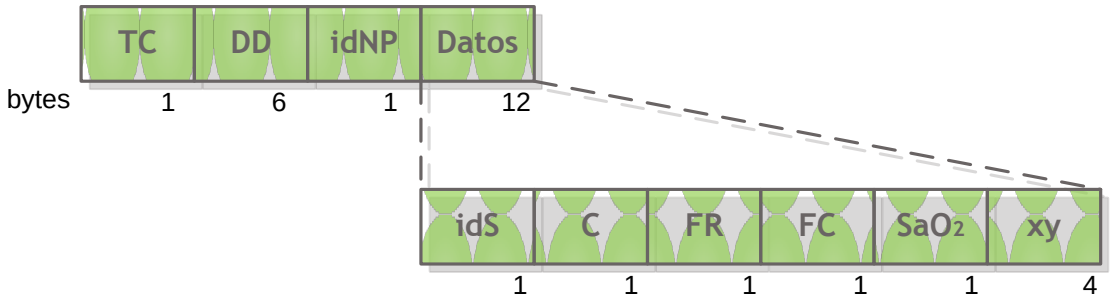
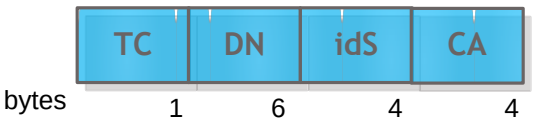


Figura 18. Trama de datos

Por último está la trama de datos con el formato indicado en la figura 18. idNP es el campo que indica el identificador del NP relacionado al sensor conectado al paciente. El campo datos está conformado por idS correspondiente al identificador del sensor al que pertenece la información, C indica la clasificación del paciente según su estado (1 - rojo, 2 - amarillo ó 3 - verde), FR es la frecuencia respiratoria del paciente, FC es la frecuencia cardiaca del paciente, SaO_2 es la saturación de oxígeno y xy indica la posición del sensor con respecto al NP al que pertenece. Estos datos una vez recibidos en la PDA son desplegados inmediatamente de manera gráfica como lo describe el siguiente capítulo.

2.4.2.4 TRAMA ADICIONAL

Considerando las recomendaciones de los cuerpos de socorro, desde el punto de vista de que es necesario el envío de una señal desde la PDA hacia un sensor específico, para razones de atención inmediata y/o ubicación rápida, fue incluida una trama denominada *trama de alarma* con los siguientes campos:



TC, en este caso toma el valor 10110110 y CA es un campo reservado para trabajos futuros que tengan como objetivo un manejo de alarmas más dedicado.

2.5 HERRAMIENTAS PARA LA VERIFICACIÓN DE PMR

Para la verificación del protocolo propuesto, debe implementarse la etapa de conexión y envío de mensajes de los NP en varias estaciones (computadores portátiles) que simulen su funcionalidad, por lo tanto, para que dicha simulación sea lo más aproximada posible a lo que serán los nodos principales en el sistema final MERIS, se decidió utilizar un sistema operativo de tiempo real para su implementación. Teniendo en cuenta que el desarrollo de los NP no es objeto del presente trabajo de grado, se utilizará un sistema operativo de tiempo real gratuito, por lo que Linux presenta la mejor opción debido a sus características de confiabilidad, flexibilidad y extensiones utilizadas para convertirlo en un completo RTOS.

Existen varias extensiones de Linux para tiempo real que modifican su kernel con el fin de cumplir su propósito. En primer lugar, está RTLinux como el primer intento de adaptación de Linux a un sistema de tiempo real llevado a cabo por Victor Yodaiken y Michael Barabanov, quienes decidieron añadir un segundo kernel situado entre el hardware y el kernel estándar de Linux, denominado HAL (*Hardware Abstraction Layer*), encargado de gestionar las interrupciones hardware para asegurar la prioridad de las tareas de tiempo real sobre las comunes, ejecutando el kernel estándar solamente cuando no hay tareas de tiempo real pendientes. Este mecanismo es denominado estrategia de micro-kernel. [46] [47]

En segundo lugar, está RTAI (*Real Time Application Interface*), creado por Paolo Mantegazza e inicialmente basado en RTLinux con su arquitectura de micro-kernel, pero gracias a su evolución hoy utiliza una arquitectura de nano-kernel similar a la estrategia anterior pero que permite la ejecución paralela de varios sistemas operativos por encima de él [46]. El desarrollo de aplicaciones con RTAI requiere una comunicación directa con el kernel por lo que las tareas o procesos de dichas aplicaciones son implementadas como módulos del kernel, en este sentido, RTAI facilita la creación de aplicaciones de tiempo real en el espacio del usuario por medio de la opción denominada LXRT (*Linux Real-Time*) sin la necesidad de crear módulos para el kernel convirtiéndose en un complemento enfocado en proporcionar baja latencia tanto en el espacio del kernel como en el del usuario [48].

En tercer y último lugar, se encuentra Xenomai, que surge a partir de RTAI en el año 2003 con el nombre RTAI Fusion, pero a causa de las diferencias entre los desarrolladores de cada proyecto fueron separados totalmente en dos extensiones de Linux distintas (ver figura 19). Xenomai a diferencia de RTAI está enfocado en la portabilidad en lugar del ofrecimiento de una latencia baja [49] [50].

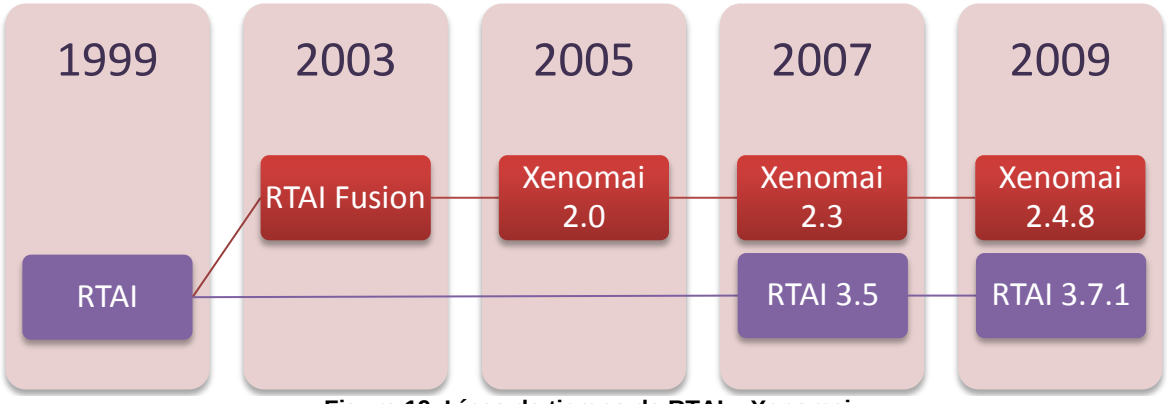


Figura 19. Línea de tiempo de RTAI y Xenomai.

Para la implementación de la comunicación con la PDA en los NP es escogido RTAI como sistema operativo porque, por una parte, RTLinux aunque es un sistema maduro, no incorpora nuevas versiones ni mayores recursos que los otros dos sistemas [51] y por otra parte, con base en estudios realizados como el que se muestra en [52] o el reporte de [49], establecen claramente que aunque RTAI y Xenomai presentan grandes similitudes en su comportamiento, RTAI tiene un mejor comportamiento en el tiempo de respuesta para aplicaciones de tiempo real estricto; además de demostrar su constante desarrollo e innovación. Para conocer más acerca de este complemento de tiempo real diríjase al ANEXO B.

En cuanto a la conexión de red en tiempo real, RTnet es un complemento *open source* para RTAI y Xenomai, creado por Ulrich Marx en la universidad de *Hannover* en Alemania, que permite el funcionamiento en tiempo real de la interfaz Ethernet por medio de un conjunto de protocolos de red de tiempo real soportando varias clases de chips según la tarjeta de red que sea deseable utilizar. Actualmente existe un solo driver para la interfaz inalámbrica, permite el manejo de las tarjetas con chip 2550, pero la disponibilidad de los recursos para la verificación del protocolo no utilizan estos chips y por lo tanto no es posible utilizar en tiempo real la interfaz inalámbrica de red para el presente trabajo de grado.

Por último, es necesario aclarar que la PDA también será implementada con RTAI[31], porque aunque desde un principio Mobilinux fue escogido como el sistema operativo más adecuado para el computador de mano, hubo razones que imposibilitaron su utilización y que están explicadas a continuación:

- Fue contactada la empresa proveedora de Mobilinux (Montavista), con el fin de conocer las licencias y precios ofrecidos. De esta manera, la licencia educativa con un precio de 1000 dólares era la más económica y conveniente para el proyecto, pero los recursos estimados para este trabajo de grado no satisficieron esta compra, por lo que fue explicado a Montavista que MERIS es un proyecto de carácter social orientado a las entidades de socorro y a cargo de una Universidad pública, con el fin de que la empresa adjudicara al proyecto MERIS una licencia

[31] La instalación de RTAI se llevará a cabo en la distribución de Linux, Ubuntu.

gratuita durante su desarrollo, sin embargo Montavista al conocer esta situación no volvió a responder a estas peticiones.

- Android, como segunda opción, es un sistema operativo *open source* totalmente gratuito que permite su personalización, así entonces, durante 6 meses se trabajó sobre este sistema con el objetivo de adaptarlo a las condiciones mínimas de tiempo real que este trabajo de grado requiere, pero este proceso implica mucho tiempo que sobrepasa los plazos establecidos para el proyecto.

- Una vez más Montavista fue contactada, explicando de nuevo las condiciones de MERIS, pero la diferencia con el primer intento de conseguir a Mobilinux es que la financiación estaba siendo gestionada a través de la Universidad del Cauca. Así entonces, fue obtenida la financiación y acuerdo con Montavista, pero aunque el proceso de compra estaba iniciado, el pago por parte de la Universidad demoró y nunca fue efectivo. Durante esta espera el plazo para la realización del trabajo de grado ya había vencido por lo que fue necesario pedir prórroga sin saber que no concluiría el proceso de compra.

De esta manera, este trabajo de grado intenta ofrecer una aplicación de escritorio con dimensiones y funcionalidades lo más cercanas posibles a lo que sería en la PDA, con el objetivo de facilitar su implementación final en Mobilinux, una vez sea posible adquirirlo.

3. PROTOTIPO DE APLICACIÓN DE MONITOREO EN TIEMPO REAL SOBRE UNA PDA PARA MERIS

Este capítulo realiza una descripción del proceso de desarrollo del sistema de monitoreo, así como de la aplicación misma que será responsable de desplegar la información de los nodos sensores de la red inalámbrica conectados a los pacientes, la cual es centralizada por el NP, cada uno con capacidad de registrar la información vital de hasta 60 pacientes.

Dicha implementación está sujeta a algunas de las restricciones y elementos de los sistemas de tiempo real ya descritas en el capitulo uno, así como características de confiabilidad y usabilidad, por lo que ilustrará de manera clara los mecanismos y servicios de tiempo real utilizados para cumplir con estas condiciones y que encuentran disponibles en la herramienta a utilizar.

Así mismo, además de la parte de rendimiento de la aplicación, mostrará el diseño de la interfaz gráfica de usuario, para la cual recurrió en calidad de consultores a algunos integrantes de los organismos de socorro, quienes dieron ciertas recomendaciones al respecto de dicho diseño con el objetivo de crear una aplicación de alta usabilidad.

Es importante recordar que debido a la imposibilidad de utilizar la herramienta Mobilinux, ideal para el desarrollo de aplicaciones con restricciones de tiempo real para dispositivos móviles, desarrollará en un computador portátil un sistema con las mismas funcionalidades que tendría la PDA.

Como proceso de ingeniería de software guiará el análisis, diseño, implementación y pruebas en el proyecto a través del Proceso Unificado de Rational (RUP, *Rational United Process*) [53]. En un principio mostrará el modelado del prototipo de la aplicación descrito a través de lenguaje UML, para posteriormente mostrar el desarrollo en concordancia con dicho modelado.

Finalmente mostrarán las pruebas respectivas que permitirán establecer el valor de los temporizadores manejados por PMR.

3.1 ANALISIS DE REQUISITOS

Inicialmente es necesario hacer una lista de los requisitos funcionales y no funcionales que deben ser satisfechos con la aplicación a desarrollar. Los requerimientos funcionales corresponden al comportamiento del sistema mientras que los requerimientos no funcionales corresponden a detalles de rendimiento, diseño gráfico, entre otros elementos.

Por consiguiente los requisitos funcionales a tener en cuenta son:

- El sistema debe permitir visualizar la información de todos los nodos sensores conectados a los pacientes (R1.1)

- La posibilidad de visualizar los nodos que están clasificados con el color verde, amarillo y rojo, además de poder diferenciar los nodos que pertenecen a cada NP (R1.2)
- Permitir el envío de una señal de alarma manual a un nodo en particular (R1.3)

Por otro lado, los requisitos no funcionales son los siguientes:

- La aplicación debe mantener un refresco con ciertas restricciones de tiempo que aseguren que la información presentada está actualizada (R2.1)
- Dado el carácter temporal e inalámbrico de la REMM debe ofrecerse un mecanismo que avise la desconexión de los NP debido al aumento de la distancia de separación entre el NP y la PDA (R2.2)
- Debe mantenerse la navegación y el cambio de pantallas al mínimo dado la información tan critica que maneja el sistema (R2.3)
- El protocolo de comunicación debe ofrecer mecanismos que aseguren la comunicación entre los NP y la PDA, evitando que agentes externos irrumpan en la REMM, o sean descartados rápidamente para evitar latencias (R2.4)
- El diseño debe mantener ciertos criterios de usabilidad, es decir, orientación al usuario más que las funcionalidades del sistema y mucho menos a los conocimientos de los analistas y desarrolladores (R2.5)

3.2 MODELO DE CASOS DE USO

En la fase de análisis fueron identificados dos actores que van a interactuar con el sistema, en primera instancia, el operador de la PDA que en general es la persona denominada director de maniobra[32], de acuerdo a las directrices de los organismos de socorro, y en segunda instancia, está el NP que reunirá la información de los sensores conectados a los pacientes del sistema MERIS.

La figura 20 muestra un diagrama del modelo de casos de uso para la aplicación de monitoreo del sistema MERIS, recogiendo los requisitos funcionales y mostrando las capacidades que este ofrece a los actores anteriormente mencionados. En referencia al operador, dicha aplicación permite visualizar la información de los nodos conectados a las víctimas de catástrofes, además de poder enviar una señal a un nodo en particular para que este emita una alarma visual o auditiva que llame la atención de alguno de los socorristas para su rápida identificación. Es necesario aclarar que el protocolo incluye el manejo de una trama de alarma pero los nodos que actualmente maneja MERIS no tienen la capacidad de mostrar dicha alarma por limitaciones de hardware y por lo tanto implementará el envío de la alarma pero no su interpretación en el nodo.

[32] El director de maniobra es quien está a cargo de la toma de decisiones en las operaciones de rescate

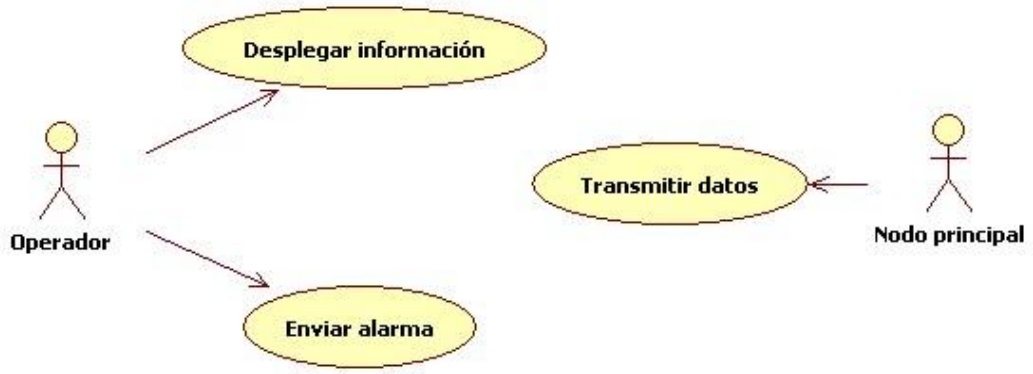


Figura 20. Modelo de casos de uso

Ahora bien, en referencia al NP, en primer lugar el sistema le permitirá asociarse a la REMM[33], condición principal necesaria para poder transmitir la información desde el NP hacia la PDA. En segundo lugar, el sistema le permitirá establecer un canal de comunicación para transmitir la información almacenada concerniente a las víctimas o pacientes MERIS. Es importante resaltar que está establecido que cada NP debería tener la oportunidad de enviar información en un periodo de tiempo menor a un segundo, entre cada turno de transmisión, denominando este plazo de 1s como el peor caso.

3.2.1 DESCRIPCIÓN EXTENDIDA DE LOS CASOS DE USO

A continuación se mostrará una serie de tablas que describen de manera detallada cada uno de los casos de uso identificados. Los casos de uso relacionados al actor operador son:

Caso de Uso	**DESPLEGAR INFORMACIÓN**
Actores	Operador (iniciador)
Tipo	Primario, abstracto
Propósito	Permitir el despliegue de la información en la PDA enviada por los nodos principales.
Resumen	El operador podrá ver la información, concerniente a los signos vitales de los pacientes, recolectada por los nodos dentro de la WSN, la cual ha sido entregada al respectivo NP y almacenada localmente. El operario tendrá la posibilidad de visualizar la información de acuerdo a diferentes criterios: • *Todos los nodos:* el sistema mostrará la información de todos los nodos sin clasificar bajo ningún criterio. • *Estado verde:* el sistema mostrará la información de los nodos que han sido clasificados como **verde** de acuerdo a las

[33] Red de etapa de monitoreo de MERIS, ver capitulo 2

	convenciones de los organismos de socorro y las pautas establecidas en el sistema. • *Estado amarillo:* el sistema mostrará la información de los nodos que han sido clasificados como **amarillo**. • *Estado rojo:* el sistema mostrará la información de los nodos que han sido clasificados como **rojo**.
Referencias cruzadas	Funciones R1.1, R1.2
Precondiciones	
Ninguna	
Flujo principal	
• Este caso de uso empieza cuando el actor Operador ingresa a la aplicación de monitoreo, automáticamente, el sistema rastreará los NP conectados a REMM e iniciará la ejecución del protocolo PMR comenzando el despliegue de la información de todos los nodos de la WSN (E1), iniciando por defecto en el modo de visualización *Todos los nodos.* • Si el operador selecciona *Estado verde,* subflujo S1 • Si el operador selecciona *Estado amarillo,* subflujo S2 • Si el operador selecciona *Estado rojo,* subflujo S3	
Subflujos	
S1: Estado verde • El sistema muestra la información de todos los nodos de la WSN encontrados en la categoría verde (E1), de acuerdo a las convenciones de los organismos de socorro y la configuración establecida en el sistema. *S2: Estado amarillo* • El sistema muestra la información de todos los nodos de la WSN que están en la categoría amarillo (E1) *S3: Estado rojo* • El sistema muestra la información de todos los nodos de la WSN hallados en la categoría rojo (E1)	
Excepciones	
• **E1:** El sistema genera una ventana de error mostrando que no ha podido establecer la conexión con ningún nodo principal por lo que no hay información para desplegar.	

Tabla 4. Descripción detallada caso de uso Desplegar información

Caso de Uso	**ENVIAR ALARMA**
Actores	Operador
Tipo	Opcional, abstracto
Propósito	Enviar una alarma a un nodo especifico
Resumen	En caso que por concepto del operador un paciente merezca cuidado especial por parte de los organismos de socorro, envía una alarma a dicho nodo para que inicie una señal visual o auditiva que llame la

	atención de los socorristas.
Referencias cruzadas	Función R1.3
Precondiciones	
Ninguna	
Flujo principal	
• Este caso de uso empieza cuando el actor Operador presiona la opción para enviar una alarma para un nodo en especial (E1) • El nodo en cuestión inicia una señal visual o auditiva	
Excepciones	
• E1: Debe mostrarse una advertencia en caso de que el nodo no responda con una confirmación la recepción del mensaje.	

Tabla 5. Descripción detallada caso de uso Enviar alarma

Las descripciones de los casos de uso relacionados al NP son:

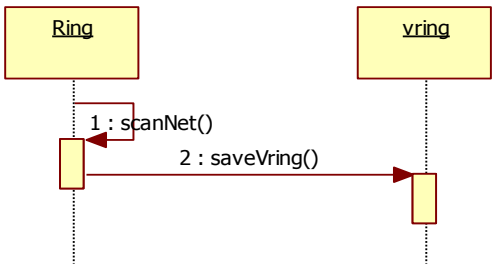
Caso de Uso	**TRANSMITIR DATOS**
Actores	Nodo principal (iniciador)
Tipo	Primario, abstracto
Propósito	Enviar información de los pacientes a la aplicación de monitoreo
Resumen	Una vez el NP ha sido registrado en el sistema y recibe el token o testigo de parte de la PDA, el NP puede comenzar la transmisión de la información almacenada referente a los pacientes.
Referencias cruzadas	Funciones R1.1, R2.1
Precondiciones	
• Debe haberse recibido el testigo para poder empezar a transmitir	
Flujo principal	
• Este caso de uso empieza cuando el actor Nodo principal recibe el testigo de parte del sistema dándole vía libre para transmitir durante un *tiempo acotado* definido (E1). En caso de no ser suficiente dicho tiempo para transmitir, subflujo S1	
Subflujos	
S1: Información pendiente El testigo es entregado a la PDA, pausando la transmisión de datos hasta que reciba de nuevo el testigo para seguir con el envío, continuando con el mismo procedimiento hasta finalizar la transmisión.	
Excepciones	
• E1: En caso de no recibir el testigo después de un periodo de tiempo determinado debe asumirse que ha sido desconectado del anillo virtual por lo que debe buscar una reconexión. El nodo debe realizar periódicamente una petición de conexión a la PDA, que permita asociarse al anillo virtual sin realizar un desperdicio de potencia.	

Tabla 6. Descripción detallada caso de uso Transmitir información

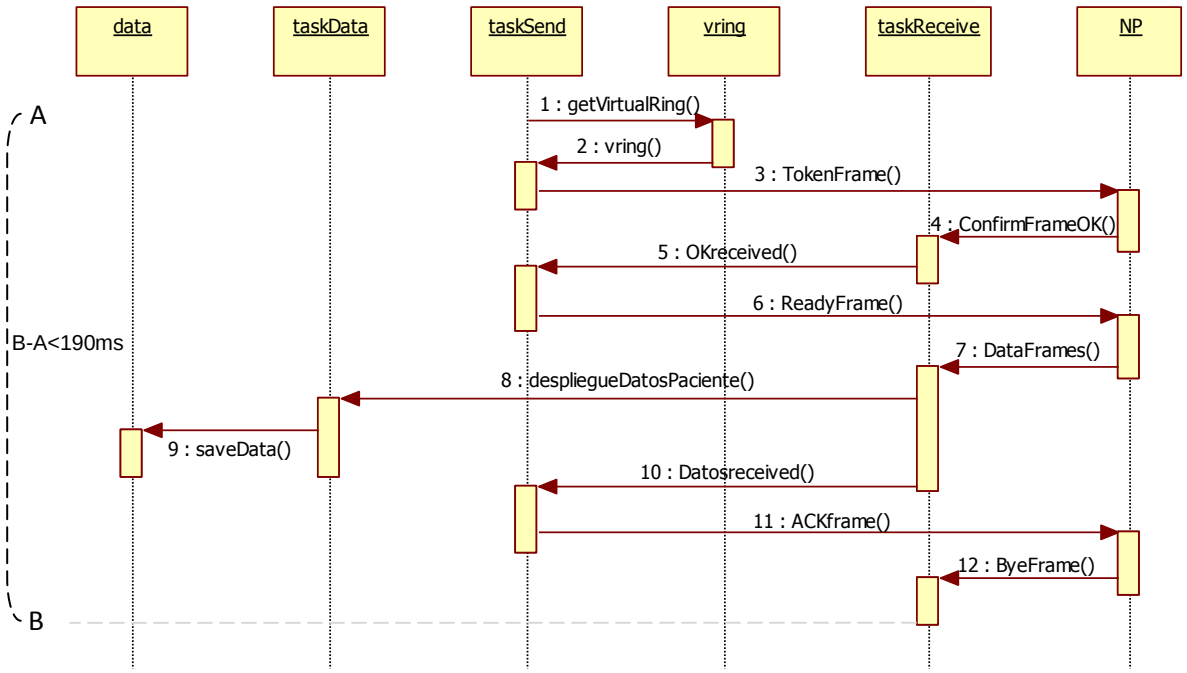
3.2.2 DIAGRAMAS DE SECUENCIA

De acuerdo a los casos de uso anteriores fueron identificados para la PDA dos procesos de tiempo promedio (proceso encargado de la interfaz gráfica, proceso de escaneo de red) y uno de tiempo real (proceso encargado de PMR), que permitirán el despliegue y la comunicación con los NP; de la misma manera dentro del proceso encargado de la ejecución del protocolo PMR, fueron identificadas tres tareas (tarea de datos, tarea de envío de tramas, tarea de recepción de tramas), que cumplen el funcionamiento descrito en el diagrama de secuencia ilustrado en la figura 21.

a.



b.



c.

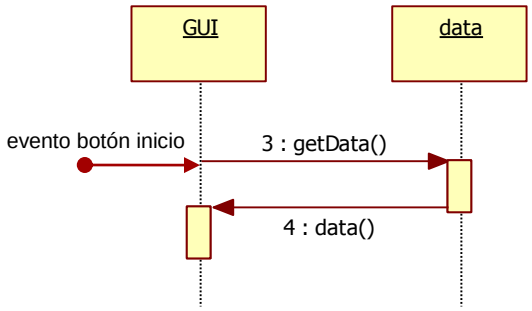


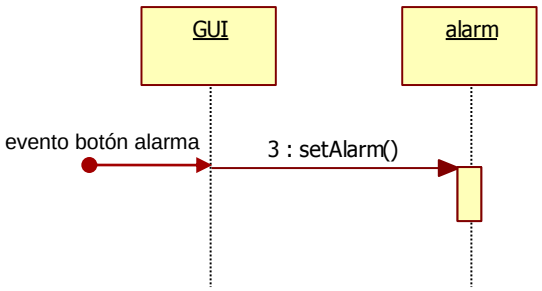
Figura 21. Diagrama de secuencia caso de uso desplegar información. a) Escaneo de red y almacenamiento de anillo virtual. b) Interacción entre tareas de tiempo real para la ejecución del protocolo. c) Despliegue de información en la interfaz gráfica.

El evento iniciador ocurre cuando el operador presiona el botón de inicio de la aplicación, iniciando la consulta del recurso que contiene la información del paciente para luego ser desplegada en la interfaz gráfica de usuario. Dicho recurso (data en la figura 21 c.) es creado y escrito con anterioridad de la siguiente manera:

Un proceso encargado del escaneo de la red (ver figura 21 a.) reconoce las direcciones IP y MAC de cada NP conectado y luego crea una tabla que representará el anillo virtual que define los turnos de envío de token almacenándola en un recurso del sistema (*mailbox*, FIFO ó memoria compartida). Otro proceso (ver figura 21 b.), por su parte, consulta el anterior recurso y toma la decisión de iniciar el envío de token al primer NP del anillo virtual, una vez la trama haya sido enviada, una tarea encargada de recibir la información de la red será activada para escuchar hasta que el NP responda con la trama correspondiente. De esta manera, es iniciado el envío y recepción de mensajes continuo de acuerdo al protocolo PMR hasta que la trama de datos es recibida por la tarea de recepción, en este instante, la tarea encargada de los datos almacena la información de esta trama en el recurso nombrado inicialmente. Posteriormente, una tarea encargada del envío de datos, avisa al NP por medio de la trama ACK que los datos han llegado correctamente, de esta manera, la trama BYE es usada por el NP para despedirse.

Teniendo en cuenta que el plazo para el despliegue de la información de todos los NP conectados a REMM es de 1s, se establece el plazo entre el envío del token a un NP hasta su despedida en 190ms, porque el máximo de NP es 5 y debe dejarse un margen de tiempo para el procesamiento de tareas de tiempo promedio del sistema.

a.



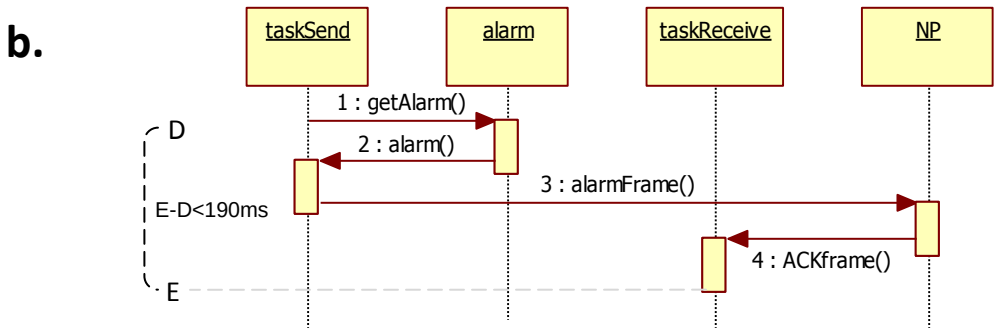


Figura 22. Diagrama de secuencia caso de uso enviar alarma. a) Registro de orden de alarma. b) Envío de alarma registrada.

Si el operador desea enviar una alarma al sensor conectado al paciente, presiona el botón correspondiente al evento e inmediatamente esta orden de alarma es almacenada en un recurso (alarm en la figura 22) para que, en el proceso encargado de PMR, la tarea que envía las tramas envíe la trama de alarma una vez su NP correspondiente tenga el turno para el envío del token, el NP por su parte confirma la recepción con un ACK. El plazo desde el envío de la alarma hasta que la PDA asegura la recepción de la misma es de 190ms, igual al plazo en el caso de enviar un token.

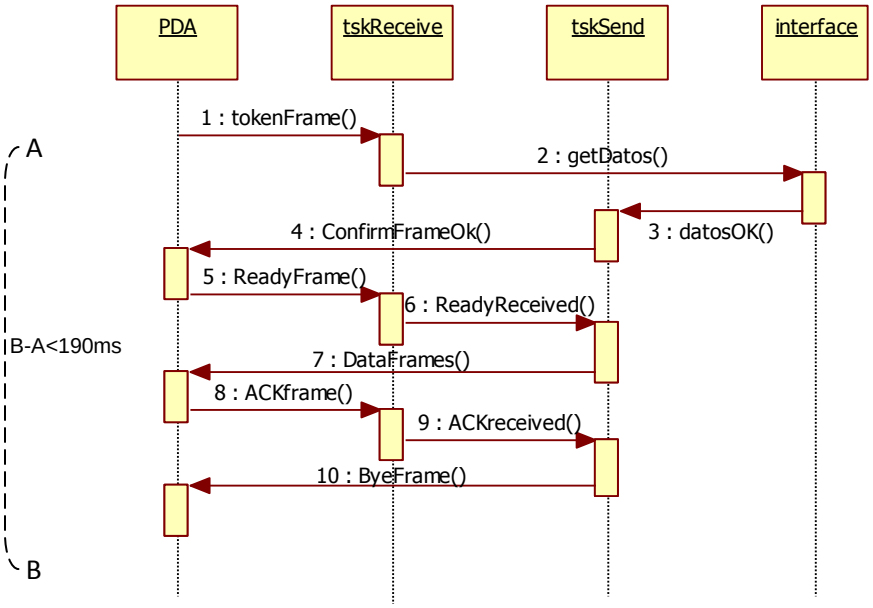


Figura 23. Diagrama de secuencia caso de uso transmitir información

En el NP, por su parte, fueron identificadas dos tareas (recibir y enviar) y un recurso importante para la simulación del funcionamiento del NP (ver figura 23). En primer lugar debe existir una interfaz entre los procesos encargados del protocolo PMR y los demás procesos que hacen parte del NP, esto con el objetivo de no afectar la planificación de los procesos que no corresponden a este trabajo de grado, en segundo lugar la tarea de

recepción de tramas que siempre debe estar activa para recibir el permiso de transmisión o token, una vez la PDA, el iniciador del proceso de transmisión del NP, envíe el token, la tarea de recepción percibe este suceso y le pide a la interfaz la información de los pacientes con el objetivo de confirmarle a la PDA si existen datos o no y almacenarlos en un buffer para su posterior envío, en caso de que existan datos, la tarea de envío de tramas envía un mensaje de confirmación hacia la PDA generando a continuación un intercambio de tramas de acuerdo al protocolo PMR.

3.2.3 DIAGRAMA DE CLASES

Así entonces, fueron planteados los diagramas de clases[34] para cada dispositivo, teniendo en primer lugar el diagrama de clases de la PDA dividido en dos tipos de clases (ver **¡Error! No se encuentra el origen de la referencia.**), la primera corresponde a las clases con los procesos de tiempo promedio o encargadas de la creación, despliegue e interacción de la interfaz gráfica de usuario[35] junto con la creación del anillo virtual utilizado por el protocolo PMR y la segunda corresponde a las clases con los procesos de tiempo real, encargadas de la ejecución del protocolo PMR.

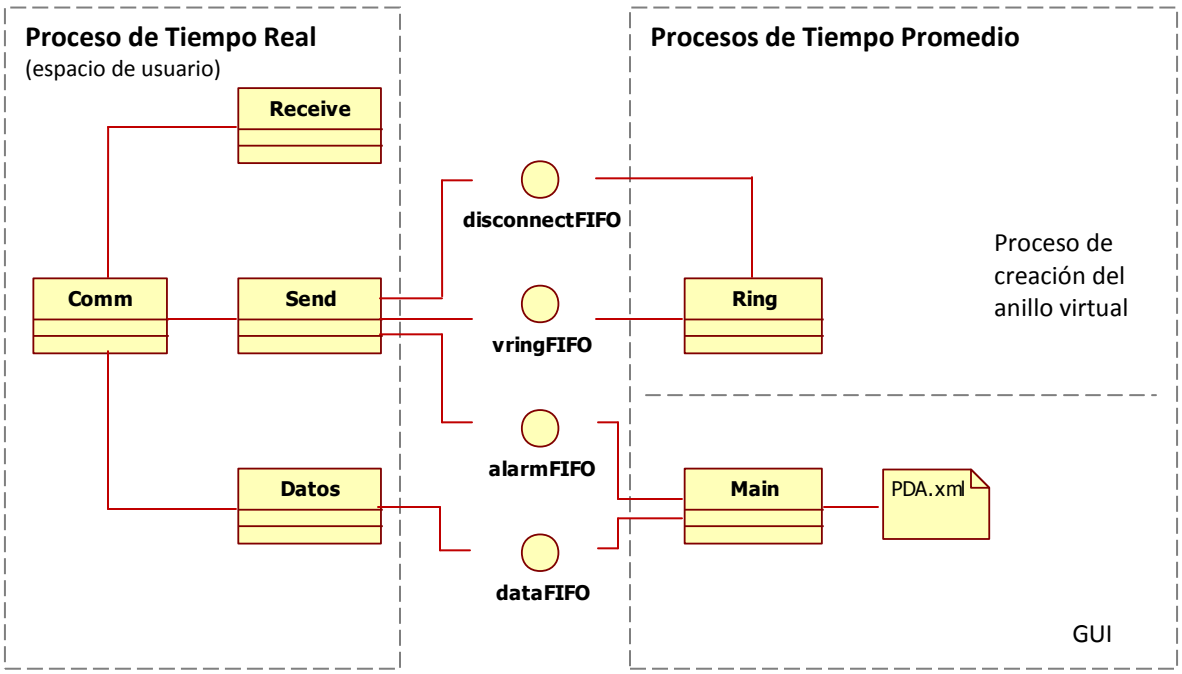


Figura 24. Diagrama de clases de la PDA

[34] Las clases hacen referencia a los hilos que son instanciados para cumplir las tareas establecidas.
[35] Estas clases se generan automáticamente con Glade. La sección 3.4 describe el funcionamiento de la GUI.

El proceso de tiempo real utiliza mensajería entre tareas, *mailboxes* y semáforos para su comunicación y sincronización respectivamente. La sincronización por medio de semáforos es la ilustrada en la figura 25 y está a cargo de la tarea de envío porque es la que tiene el control sobre los turnos de envío de testigo y por lo tanto el acceso a la información del anillo virtual.

RTAI, con el fin de no utilizar el mismo mecanismo para la comunicación entre procesos que implementa el kernel de Linux, proporciona un módulo para el kernel denominado rtai_tbx basado en *mailboxes* que permite la comunicación entre procesos de tiempo real y es el utilizado en este trabajo de grado para la comunicación entre tareas de la PDA.

Por otro lado la interfaz de comunicación entre los procesos de tiempo real y tiempo promedio son tres FIFO, una con la tabla que contiene el anillo virtual, escrita por el proceso encargado de la creación del anillo virtual y leída por la tarea encargada del envío de tramas, la segunda es una FIFO que contiene toda la información actualizada de todos los pacientes de MERIS, escrita por la tarea de datos y leída por el proceso encargado de la interfaz gráfica para realizar el despliegue de dicha información y la tercera es la pila utilizada para gestionar el evento de desconexión de un nodo principal.

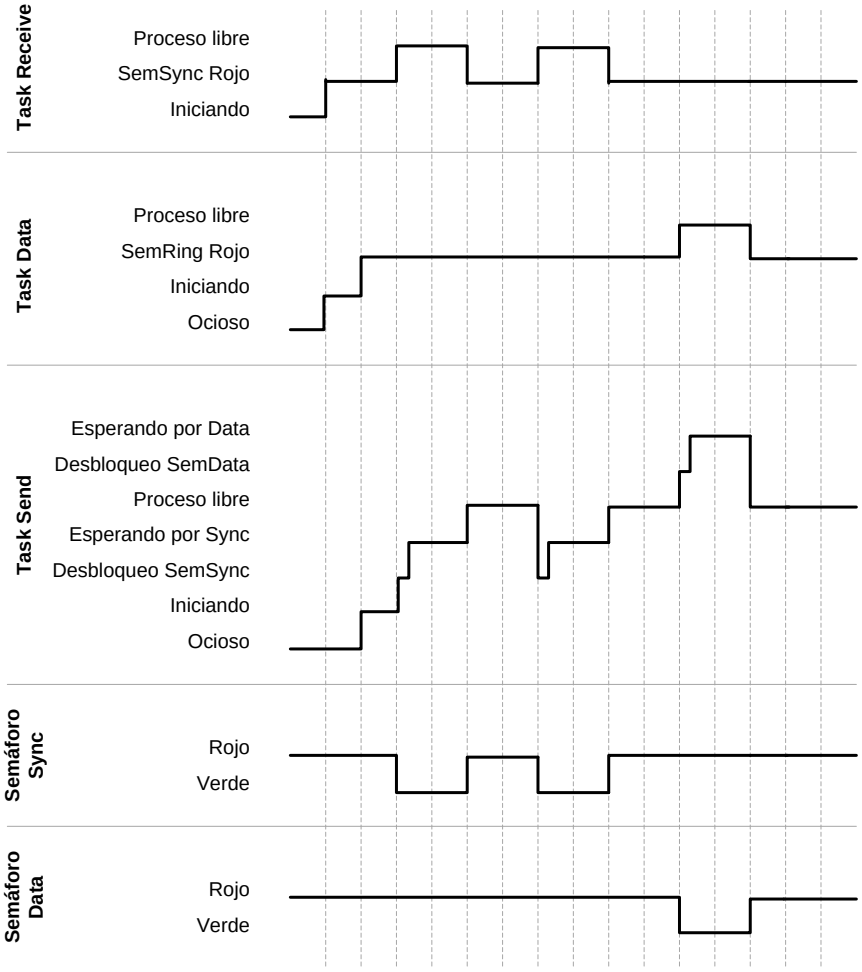


Figura 25. Sincronización entre tareas de tiempo real en la PDA

La creación del anillo virtual, realizado en tiempo promedio, es un escaneo de la red por medio del protocolo de resolución de direcciones (ARP - *Address Resolution Protocol*), el cual proporciona el servicio de obtención de direcciones de control de acceso (MAC) de los equipos conectados en la misma red física, interrogando a los equipos de la red para averiguar sus direcciones físicas y posteriormente crear una tabla de búsqueda entre las direcciones lógicas y físicas en una memoria caché denominada caché de ARP para usos posteriores [54]. De esta manera, es guardada esta tabla en una FIFO que será leída por la tarea de envío para conocer los NP conectados a REMM y el turno de permiso de comunicación. Teniendo en cuenta que debido al desplazamiento del operador los NP pueden desconectarse de la red, es necesario que el proceso encargado del escaneo, sea ejecutado constantemente sin afectar los otros procesos, planteando con esto la ejecución del mismo por medio de un hilo periódico de un segundo y el establecimiento de un tiempo muerto en el proceso de tiempo real (porque es un proceso de tiempo real reactivo no periódico) para permitir la atención de los procesos de tiempo promedio.

En cuanto a la gestión de envío de alarmas, fue establecido la utilización de una FIFO que almacene en tiempo promedio la orden de envío de alarma a un sensor cuando el socorrista lo establezca y de esta manera la tarea encargada del envío del token, en caso de encontrar esta orden, envía la o las alarmas en lugar de un token hacia el NP correspondiente al o los sensores, cuando sea el caso.

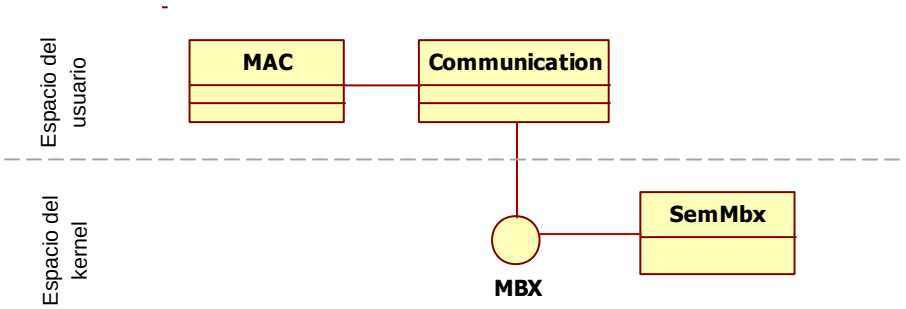


Figura 26. Diagrama de clases del NP

Para la simulación del comportamiento de los NP fue propuesto el diagrama de clases ilustrado en la figura 26. En primer lugar, es necesario no olvidar que el NP está compuesto de muchas tareas que se encargan de la comunicación y almacenamiento de información de los nodos sensores de MERIS y que no son parte de este trabajo de grado, por tal motivo, la propuesta establece como interfaz entre dicho procesamiento y las tareas encargadas de atender al protocolo PMR para la transmisión de información, un *mailbox* escrito por ellos, que esté disponible para la lectura en caso de recibir un token y de esta manera no afectará a las demás tareas del NP. En consecuencia, existen dos procesos, uno que simula la escritura del *mailbox* propuesto y que está en el espacio del kernel disponible para cualquier aplicación que desee acceder al recurso y el otro, en el espacio del usuario, encargado de la comunicación con la PDA y hace uso del recurso antes descrito.

3.2.4 DIAGRAMA DE TIEMPO

Con el objetivo de cumplir el plazo entre el envío del token a un NP hasta su despedida (190ms), fueron establecidos diversos plazos de procesamiento y recepción de tramas para el proceso de tiempo real, como es ilustrado en la figura 27.

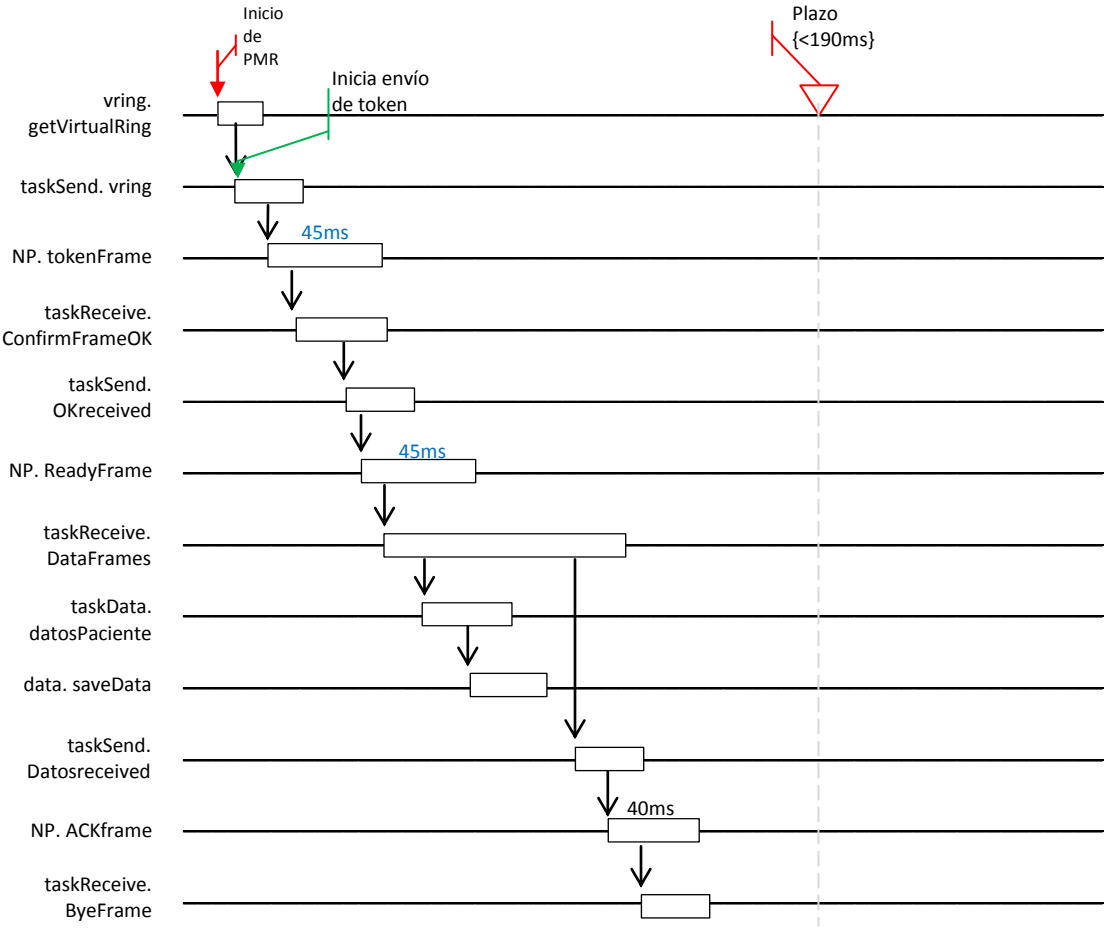


Figura 27. Diagrama de tiempo proceso de tiempo real

De esta manera, el valor de inicialización de los temporizadores utilizados por PMR es:

a. TIEMPO_ACOTADO

Considerando que la mayor duración de una transmisión entre un NP y la PDA es dada al ubicarse en el umbral de distancia de 50m y al transmitirse toda la información de los pacientes, el valor de TIEMPO_ACOTADO es establecido en 45ms.

b. MAX_TIEMPO_TOKEN_RECIBIDO

Considerando que el mayor tiempo de respuesta de confirmación de un NP hacia la PDA está dado al hallarse a una distancia de 50m, el valor de este tiempo es establecido en 45ms.

c. MAX_TIEMPO_OCIOSO

Considerando que el máximo tiempo ocioso es cuando todos los NP están en el umbral de distancia (50m) y tienen que transmitir toda su información, el valor de MAX_TIEMPO_OCIOSO es 950ms (190ms x 5NP).

d. MAX_TIEMPO_EN_ANILLO

Considerado este tiempo como el máximo plazo establecido como condición de despliegue de definido por el proyecto MERIS, es decir, 1 segundo.

3.3 INTERFAZ GRÁFICA DE USUARIO

Figura 28. Interfaz gráfica de usuario

De acuerdo a las funcionalidades descritas anteriormente, en la figura 28 puede visualizarse el diseño de la interfaz gráfica de usuario GUI[36]. Este diseño minimalista pretende ofrecer las funcionalidades del sistema de manera muy simple, evitando la

[36] GUI, por sus siglas en ingles

navegación a través de muchas pantallas pero a la vez manteniendo la usabilidad, intentando cumplir con algunos de los requerimientos no funcionales del sistema pero de gran importancia para la aplicación en general. Una característica importante es el ajuste de la interfaz a las propiedades gráficas del sistema operativo, lo que la hace amigable al usuario.

Está desarrollada en Glade (*Glade Interface Designer*), una herramienta GTK/GNOME orientada al desarrollo de interfaces gráficas independiente del lenguaje de programación utilizado en la lógica de la aplicación. Las versiones más recientes de glade generan un archivo XML en lugar del código fuente con el objetivo de almacenar los elementos de la interfaz gráfica diseñada, permitiendo la construcción de la interfaz en tiempo de ejecución gracias a la librería libglade [55]. Este trabajo de grado utilizó la versión 3 que maneja archivos XML haciendo de la interfaz gráfica un elemento más liviano, creado en tiempo de ejecución y totalmente portable.

Figura 29. Disposición de elementos de la GUI

La figura 29 presenta más detalladamente la distribución de los elementos que hacen parte de la GUI. En la barra de título está como ícono la Estrella de la Vida –SOL[37]- porque a nivel internacional representa a las unidades y personal de emergencia a nivel prehospitalario y su utilización no tiene inconvenientes con el uso de emblemas reservados.

La barra de menús conformado en primer lugar por “Archivo” que a su vez tiene la opción de salir de la aplicación deteniendo todos los procesos (tareas) y recursos (semáforos, *mailboxes*) iniciados para su funcionamiento, en segundo lugar “Ver” que contiene la opción que permite consultar los créditos de la aplicación y en último lugar se encuentra “Ayuda” que contiene las instrucciones básicas para el manejo de la aplicación.

Como uno de los requisitos funcionales es la posibilidad de realizar un filtro a la información presentada en pantalla por facilidad para los organismos de socorro y en particular para el operador, la barra de botones de clasificación contiene cuatro elementos para controlar la visualización de la información de los pacientes. Así entonces, en primer lugar un botón en forma de ojo ubicado en la parte derecha de la barra permite desplegar en orden crítico (primero los pacientes clasificados en rojo, luego los clasificados en amarillo y por último en verde) todos los nodos independiente del NP al que pertenezcan y la clasificación que tengan en el momento, en segundo lugar, existen tres botones de colores rojo, amarillo y verde que permiten la visualización de la información de acuerdo a la clasificación *triage* del paciente. El orden de los botones y del despliegue de la información fue establecido de esta manera (rojo, amarillo, verde) con el objetivo de mantener un estándar para los dos casos, resaltando que siempre estarán primero los pacientes en peor estado con el fin de evitar confusiones en caso de que el socorrista tenga daltonismo.

La barra de símbolos está conformada por los elementos que representan los signos vitales censados por MERIS establecidos en un orden especial sugerido por los organismos de socorro debido a las prioridades que el personal de emergencia maneja.

La tabla de información despliega por columnas el estado e identificadores de cada paciente, en la columna 1 se muestra el color correspondiente a la clasificación dada al paciente, en la columna 2 el identificador del nodo con el formato: Nodo NP-Sensor, en la columna 3 la frecuencia respiratoria en respiraciones por minuto, en la columna 4 la frecuencia cardiaca en latidos por minuto (lpm) y la saturación de oxígeno en la sangre en la columna 5.

La tabla también ofrece la funcionalidad de envío de alarma manual a un nodo en particular por medio de la pulsación del color que indica el estado del paciente (columna 1) correspondiente al nodo a quien será enviada la alarma, esto con el fin de evitar la introducción de más elementos que complicarían el manejo de la aplicación.

La barra de botones de ejecución e información está compuesta en primer lugar de un botón con el símbolo de inicio encargado de arrancar el despliegue de la información por primera vez, en segundo lugar, un botón identificado con el signo de interrogación que permite visualizar la ayuda o instrucciones de la aplicación y por último lugar, un botón

[37] The Star of Life [58], por sus siglas en ingles.

con el signo de admiración que permite visualizar los créditos de la aplicación (ver figura 30).

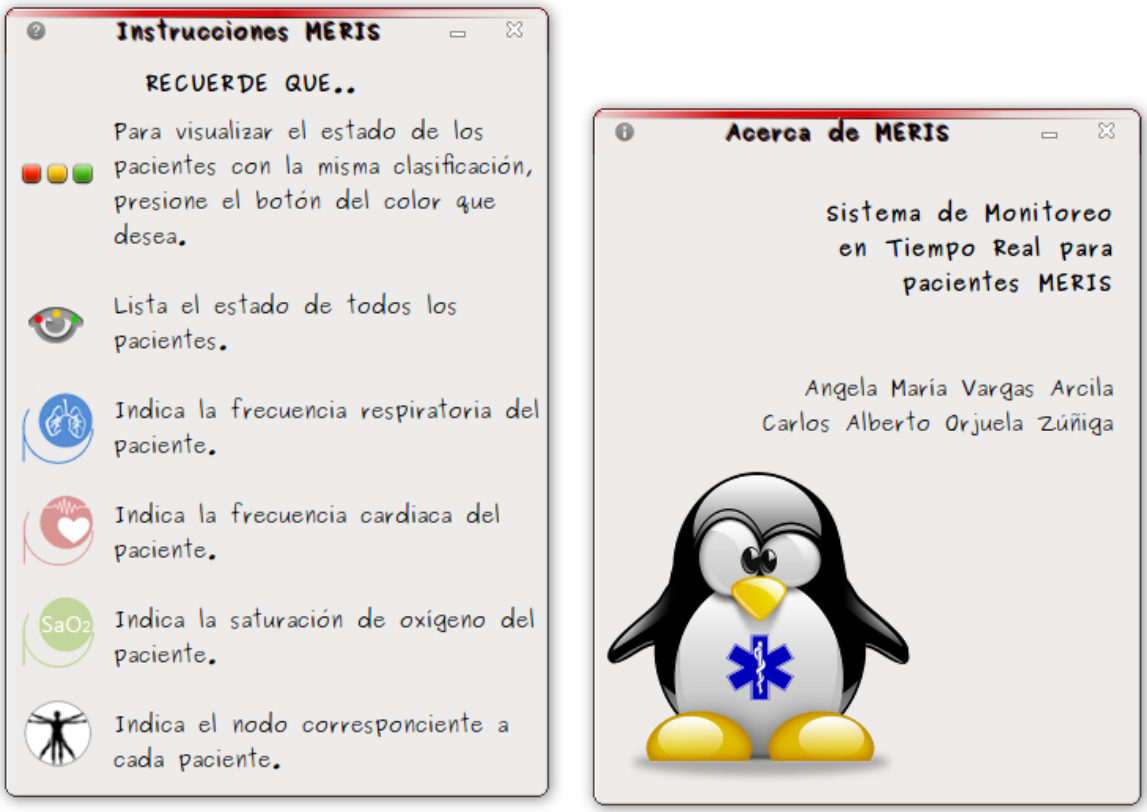


Figura 30. Ventanas de ayuda y créditos.

En otro orden de ideas, uno de los requerimientos no funcionales es la capacidad de advertir que un NP esta desconectado del anillo virtual (Función R2.2). Esto será determinado si el NP en cuestión no responde a dos envíos del testigo de parte de la PDA, entonces pasará a ser considerado como desconectado por lo que debe ser removido del anillo virtual y desplegarse un mensaje en el cual se muestre que uno de los nodos principales ha sido desconectado (Figura 31). Esto le da la posibilidad al operador de la PDA de acercarse nuevamente a una zona donde pueda ofrecer cobertura a dicho nodo.

Figura 31. Ventana para anunciar desconexión

Por último, el logo (ver figura 32) escogido para el sistema de monitoreo es Tux, la mascota oficial de Linux, debido a que el sistema de monitoreo fue desarrollado completamente en este sistema operativo. Además, de la estrella de la vida por las razones explicadas anteriormente.

Figura 32. Logo del sistema de monitoreo para pacientes MERIS

El diseño y plazos de tiempo del sistema establecidos por el presente capítulo, serán corroborados en el capítulo de pruebas, donde la verificación del buen funcionamiento y diseño de la etapa de monitoreo de los pacientes de MERIS será realizado.

4. PRUEBAS

El presente capítulo muestra una serie de pruebas que pretenden corroborar el correcto funcionamiento y cumplimiento de las condiciones planteadas en el diseño del sistema, como por ejemplo, los plazos de tiempo para cada una de las tareas y en particular, las ventanas de tiempo establecidas en la fase diseño del sistema.

En consecuencia, son calculados los tiempos de cómputo del sistema funcionando bajo diferentes condiciones de operación para luego realizar una comparación de dichos tiempos con los establecidos en la figura 27 y establecer el cumplimiento de los plazos y las modificaciones requeridas para mejorar el funcionamiento del sistema desarrollado.

Es importante resaltar al lector que los escenarios de pruebas incluyen casos en campo abierto e interiores, a distintas distancias entre la PDA y el o los NP, según sea el caso. Estas condiciones fueron definidas de acuerdo a los posibles escenarios de funcionamiento del sistema MERIS, como por ejemplo, un campo abierto con interferencias electromagnéticas y obstrucciones concernientes al terreno.

Por último, es presentado un análisis detallado de los resultados obtenidos y una comparación entre estos que muestre claramente el adecuado cumplimiento de las condiciones planteadas en la fase de diseño de la aplicación, así como los cambios pertinentes incluyendo trabajos futuros.

4.1 CONDICIONES DEL SISTEMA PARA LAS PRUEBAS

Antes que nada, es debido establecer las condiciones de ejecución de las pruebas. Fue necesario realizar la instalación del sistema operativo GNU/Linux con el correspondiente parche de RTAI (ver ANEXO A), para poder tener las facilidades y capacidades de un sistema operativo de tiempo real, en cada uno de los computadores portátiles, que sumaban un total de seis, dado que era necesario llevar al sistema a las condiciones de funcionamiento de máximo estrés, es decir, con el número máximo de nodos, a la máxima distancia de operación definida, que fue de 50 metros.

Los equipos poseen variadas características con respecto al hardware, lo que permite establecer en cierto grado la portabilidad del sistema, que fue desarrollado en uno de los equipos y portado a los demás sin variaciones de ningún tipo a nivel de programación. Algo importante de resaltar es que en la compilación del kernel de Linux con el parche de RTAI, fue utilizado solo uno de los núcleos del procesador en cada computador sin limitar la operación del sistema en equipos con múltiples procesadores, sino por el contrario, puede funcionar en aquellos con capacidades más restringidas.

Por otro lado, la locación elegida para hacer las pruebas en el exterior fue el parque de ingenierías, dado que era el lugar propicio para ello ya que había al momento de realizar las mismas, cinco redes inalámbricas WiFi, las cuales generaban cierta interferencia electromagnética. Además, la gran cantidades de árboles sembrados en esta área, presenta desafíos importantes, debido a obstrucciones naturales del terreno, así como interferencia a las señales de radio por la humedad contenida en los mismos.

En lo referente al sistema, dado que este utiliza una red tipo ad-hoc con una comunicación no orientada a la conexión, existe una alta probabilidad de pérdida de paquetes debido a condiciones de la red, por lo cual las diferentes operaciones de envío y recepción desde la PDA, fueron definidas como *no bloqueantes*, lo que permite determinar en todo momento una pérdida de algún paquete y dar el tratamiento respectivo, según sea el caso. Para esto, es establecido un tiempo límite de escucha, donde el sistema debe esperar respuesta de parte de un nodo en particular, y llegado dicho tiempo límite, realizar una retransmisión o ignorar el nodo en cuestión y continuar con el siguiente, dependiendo del tipo de respuesta esperado. Este *tiempo de expiración* fue definido en un segundo para las pruebas iniciales, lo bastante amplio para evitar truncar el proceso de comunicación y lograr así un promedio real de cada una de las etapas de la comunicación mostradas en la figura 27.

Para verificar los tiempos utilizados en cada proceso, fue tomada una marca de tiempo en varios puntos clave del sistema. Estas marcas de tiempo almacenadas en una pila FIFO[38], eran tomadas de esta y enviadas a un archivo por una tarea de tiempo promedio, independiente del sistema bajo prueba, para así interferir en lo mínimo posible en las condiciones de funcionamiento. La información almacenada era mostrada de la siguiente forma:

```
  25 Z 19795839
  26 N 88866726
* 27 C 27780386
* 28 D 52305603
* 29 Z 23977103
* 30 N 104141035
  31 T 1280267018
+ 32 C 52163965
+ 33 D 27641261
+ 34 Z 27782900
+ 35 N 107650145
+ 36 C 43816524
+ 37 D 14943261
+ 38 Z 30570409
+ 39 N 89408974
+ 40 C 9361538
+ 41 D 3240081
+ 42 Z 5584237
+ 43 N 18263799
+ 44 C 51761679
+ 45 D 43593590
+ 46 Z 32089040
+ 47 N 122756834
+ 48 T 346622471
  49 C 21076451
  50 D 2972729
```

Figura 33. Marcas de tiempo en nanosegundos

En la figura 33, las marcas de tiempo señaladas con un asterisco (*), muestran los tiempos tomados desde el inicio hasta el final de la comunicación con un nodo y aquellas

[38] First-In First-Out, política de “primero en entrar, primero en salir”

marcadas con el símbolo de suma (+), muestran los tiempos para todo el anillo virtual, en este caso, compuesto por cuatro nodos.

El significado de cada una de las letras lo muestra la Tabla 7:

IDENTIFICADOR	TRAMA
C	Confirmación OK
D	Datos
Z	Despedida
N	Final Nodo
T	Final Anillo
R	Re sincronización
E	Expiración
X	Confirmación Cancel

Tabla 7. Indicadores marcas de tiempo

Estos indicadores concuerdan con los mensajes definidos en la tabla 3 y la figura 14, mostrados en la página 322. En primera instancia, la letra **C** indica una confirmación de parte del nodo diciendo que tiene datos y está listo para transmitir, esta puede ser reemplazada por una **X**, lo que indicaría que el nodo no tiene ninguna información almacenada de los pacientes. En segunda instancia, la letra **D** indica el tiempo que toma desde que la PDA indica que está listo para recibir los datos de los pacientes y el momento en que le llega esta información. En tercera instancia, una vez la PDA recibe los datos, envía un ACK hacia el nodo y recibe un mensaje de despedida por parte de dicho nodo, lo que es indicado con la letra Z. En cuarto lugar, la letra **N** muestra la terminación del proceso de comunicación con un nodo y finalmente la letra **Z** indica que todo el anillo virtual ha sido "recorrido" iniciando nuevamente la comunicación con el primer nodo en cola. Los mensajes indicados con las letras **R** y **E** son casos excepcionales en los cuales es recibido un mensaje a destiempo por lo que es iniciado un proceso de re sincronización o el tiempo de espera de un mensaje a terminado, respectivamente.

4.2 DESCRIPCIÓN DE PRUEBAS

4.2.1 PRUEBA EXTERIOR DISTANCIA MÁXIMA, UN NODO

La primera de las pruebas fue hecha al aire libre, en el parque de ingenierías como fue indicado anteriormente, a una distancia de 50 metros entre la PDA y un solo nodo, moviendo periódicamente de lugar la PDA, para simular un desplazamiento de la persona a cargo de la misma, eso sí, manteniendo siempre la distancia estipulada. 80000 marcas de tiempo fueron tomadas, las cuales fueron almacenadas en el archivo denominado *salida* en los archivos anexos a la presente monografía[39].

[39] Los archivos que contienen los resultados de las pruebas realizadas, se encuentran anexados solamente en formato digital debido a su extensión para la impresión.

4.2.2 PRUEBA EXTERIOR 30 METROS, UN NODO

La segunda prueba pretendía mantener las condiciones anteriores, pero a una distancia menor para comprobar así, como era afectado el funcionamiento del sistema por la distancia. La distancia ahora fue conservada en 30 metros, en la misma locación. Fueron tomadas también 80000 marcas de tiempo, las cuales fueron almacenadas en los archivos *salida3.1* y *salida3*.

4.2.3 PRUEBA EXTERIOR MÁXIMA DISTANCIA, TODOS LOS NODOS

En la tercera prueba era requerido llevar el sistema a sus parámetros de operación máximos, por lo que la idea inicial era realizar la comunicación con 5 nodos principales. Debido a algunas fallas en algunos de los computadores portátiles, relacionadas con el hardware del mismo y ajenos al sistema MERIS, y debido a la imposibilidad de conseguir un nuevo equipo, fue necesario hacer la prueba con un nodo menos, pero como es posible observar nuevamente, el tiempo adicional generado por la adición de un nodo al anillo virtual no excede los límites impuestos en las etapas de diseño del sistema. En esta prueba es importante resaltar la comprobación de los procesos de desconexión y reconexión de algunos nodos demostrando que esto no afecta en absoluto el correcto funcionamiento del sistema. En esta prueba no fue posible realizar la misma cantidad de marcas de tiempo debido a que la batería de los equipos terminó antes de poder finalizar la prueba, pero de igual forma una cantidad representativa de medidas fue recopilada, ella fue realizada a 50 metros alrededor de la PDA.

4.2.4 PRUEBA INTERIOR, UN NODO

El objetivo de esta prueba era probar el sistema en el interior de una estructura, en este caso, el edificio de la facultad de Ingeniería Electrónica y Telecomunicaciones de la Universidad del Cauca. Esta locación nos ofrecía el lugar excelente para probar dado que la estructura es bastante robusta generando más desafíos para el funcionamiento del sistema. La prueba duró un poco más de 3 horas con una toma total de 70300 marcas de tiempo a 37 metros, con dos muros de por medio entre la PDA y el nodo. Los resultados fueron almacenados en el archivo *salida4*.

Aquí finalizó la primera fase de pruebas, donde un estudio detallado de los resultados permitió reconocer un error de programación donde era realizada una retransmisión de los paquetes, indiferente del tipo de paquete perdido, lo cual fue corregido para que solo ocurriera en caso de que dicho paquete fuera el de TOKEN disminuyendo así los tiempos finales. Así mismo, los tiempos de espera en la escucha fueron ajustados de acuerdo a los promedios obtenidos y reducidos a 45 milisegundos (resultado mostrado a continuación) para que las restricciones de tiempo sean cumplidas incluso en caso de pérdida de paquetes.

4.2.5 PRUEBA EXTERIOR, SISTEMA OPTIMIZADO

La segunda fase de pruebas fue realizada con los cambios estipulados, pero debido a la imposibilidad de contar con todos los equipos nuevamente para realizar la prueba con cuatro nodos y la PDA fue hecha con 2 nodos, pero de igual forma la información recabada es lo suficientemente convincente para determinar el correcto funcionamiento del sistema y el cumplimiento de las condiciones establecidas. La cantidad de marcas de tiempo tomadas fue mayor de 80000.

4.3 CÁLCULO DE RESULTADOS

Es importante decir que debido a que los cambios hechos para la segunda fase de pruebas, en los resultados de la pruebas de la primera fase existían ciertos picos de tiempo (mayores de un segundo), los cuales no correspondían al funcionamiento final del sistema que incluía los ajustes finales, principalmente en los tiempos de expiración, fue empleado el mecanismo de media geométrica el cual ajusta estos tiempos pico a los resultados reales de funcionamiento final. Esto es demostrado con los resultados de la prueba final donde ya no fue utilizada la media geométrica sino únicamente la media aritmética demostrando la validez de los resultados de las pruebas anteriores.

Las fórmulas para el cálculo de la media geométrica y aritmética son:

$$\bar{x} = \sqrt[n]{\prod_{i=1}^{n} x_i} = \sqrt[n]{x_1 \cdot x_2 \cdots x_n}$$

$$\bar{x} = \frac{\sum_{i=1}^{n} a_i}{n} = \frac{a_1 + \cdots + a_n}{n}$$

Figura 34. Media geométrica y media aritmética respectivamente.

Las tablas siguientes tablas muestran los resultados de las pruebas realizadas:

PRIMERA FASE										
PRUEBA	**PROMEDIO C (nanosegundos)**		**PROMEDIO D (nanosegundos)**		**PROMEDIO Z (nanosegundos)**		**PROMEDIO N (nanosegundos)**		**PROMEDIO T (nanosegundos)**	
	A	**G**	**A**	**G**	**A**	**G**	**A**	**G**	**A**	**G**
4.2.1	11.847.097	9.623.785	11.421.771	9.240.357	9.859.211	6.811.684	33.337.949	27.976.399	80.919.753	46.676.250
4.2.2	18.928.316	13.343.165,14	17.491.726	12.479.655 ,33	52.810.966	40.200.866 ,6	16.253.594	10.007.427 ,46	121.275.818	64.412.608,96
4.2.3	29.448.972	16.790.166,36	27.575.016	15.043.989 ,43	28.225.295	14.695.240 ,49	85.301.918	57.636.764 ,72	677.123.239	442.102.169,44
4.2.4	42.257.095	27.919.527	40.732.786	27.168.674	39.527.782	26.124.122	122.387.039	88.003.526	202.530.141	118.555.144

Tabla 8. Resultados primera fase pruebas

SEGUNDA FASE					
PRUEBA	**PROMEDIO C (nanosegundos)**	**PROMEDIO D (nanosegundos)**	**PROMEDIO Z (nanosegundos)**	**PROMEDIO N (nanosegundos)**	**PROMEDIO T (nanosegundos)**
	ARITMÉTICO	**ARITMÉTICO**	**ARITMÉTICO**	**ARITMÉTICO**	**ARITMÉTICO**
4.2.5	5.706.855,00	5.032.876,00	4.817.585,00	15.492.564,00	48.330.471,00

Tabla 9. Resultados segunda fase pruebas

Donde **A** se refiere al promedio aritmético y **G** al promedio Geométrico

Específicamente hablando de la comparativa entre la prueba realizada con cuatro nodos y aquella con dos, vemos que aunque la relación del número de nodos es de 1 a 2, la relación de tiempos es casi de 1 a 10, demostrando la significativa mejora que tuvo el funcionamiento del sistema, con los cambios discutidos anteriormente.

Con los anteriores resultados de la Tabla 8 y la Tabla 9, es comprobado de manera inequívoca como las modificaciones realizadas permiten operar al sistema dentro de los parámetros de diseño establecidos, principalmente el parámetro de comunicación **dentro de una ventana de un segundo** y muestran de igual forma la validez del presente proyecto.

5. CONCLUSIONES, RECOMENDACIONES Y TRABAJOS FUTUROS

5.1 CONCLUSIONES

- Las redes Ad-Hoc aunque son las más adecuadas en situaciones de emergencia por su fácil y rápido despliegue, todavía son demasiado vulnerables para soportar sistemas de tiempo real, debido a la no implementación de nuevos protocolos a nivel de red orientados a la transmisión de tráfico de tiempo real, y ajustables a casos particulares como el caso de REMM.

- El software libre, como el caso de RTAI, ha hecho un gran aporte a los sistemas de tiempo real, porque ofrece un buen rendimiento en el tiempo y control directo del hardware en sistemas que brindan un control del *software transparente y susceptible de modificación*[40]*.*

- La etapa de monitoreo en tiempo real para MERIS, no sólo agrega funcionalidad a este proyecto, sino que se resume en una ayuda potencial a los organismos de socorro, porque agiliza y optimiza la asignación del personal de socorro y atención del paciente.

- WiFi es una tecnología necesaria en los sistemas de tiempo real cuando éstos están orientados a la atención médica en situaciones de emergencia, pero debido a su inmadurez a nivel de red en cuanto a tiempo real se refiere, la alternativa para su adecuado manejo en estos sistemas, es por un lado, contrarrestar cualquier retardo o desconexión de la red por medio de una correcta planificación del sistema y métodos de control de fallos, y por otro lado, hacer que el hardware de la interfaz inalámbrica priorice la atención a los procesos de tiempo real sobre cualquier otro, esto es logrado teniendo un completo sistema operativo de tiempo real ó agregando un complemento de tiempo real que por medio de drivers controle la tarjeta de red.

- Dado que en el sistema existe la necesidad de realizar ciertas operaciones que no cuentan con un soporte propio desde el sistema operativo, tales como acceso a la red y aún más importante, el despliegue de información sobre la interfaz gráfica, se muestra la forma más eficiente para realizarlas, ofreciendo una interacción entre las tareas de tiempo real y aquellas de tiempo promedio que involucran a las operaciones descritas anteriormente, principalmente, a través de pilas tipo FIFO, y dejando un lapso de tiempo para la atención de estas mismas.

- Para realizar la depuración del sistema fueron tomados entre cinco mil y diez mil ciclos, los cuales comprendían el proceso completo de comunicación entre la PDA y todo el anillo virtual establecido, indiferente del numero de nodos que lo

[40] Frase de Lawrence Lessig, en el libro Software Libre para una Sociedad Libre de Richard Stallman

compusieran, considerando que los resultados mostrados son lo suficientemente fehacientes como para comprobar el correcto funcionamiento y validez del presente trabajo.

- Los sistemas operativos basados en Linux proporcionan un mejor rendimiento que cualquier otro sistema operativo basado en software privativo porque son sistemas sostenidos por una comunidad confiable y capaz de ofrecer un desarrollo y actualización constante del sistema.

- La utilización de software libre es totalmente gratificante al cooperar con el desarrollo de nuevo software que ofrezca libertad al usuario para su manejo, ejecución, distribución y modificación (aspectos esenciales para un buen desarrollo de software), sin restricciones ni obligaciones de comunicar su utilización a una entidad específica.

- Es importante acudir a los foros especializados existentes en la red, ya que éstos pueden ofrecer un soporte importante y válido al momento de solucionar problemas que surjan en las diferentes fases del proyecto, en particular en el desarrollo del mismo.

- La interacción entre el usuario final de una aplicación y su diseñador, es esencial cuando dicha aplicación está orientada a optimizar una actividad realizada por el usuario, de esta manera es posible crear software con características de usabilidad y totalmente intuitivo.

- A la hora de desarrollar un proyecto, es importante asegurarse de que las tecnologías a manejar sean lo suficientemente maduras para una adecuada y rápida ejecución del proyecto, así como un dimensionamiento correcto en cuanto al tiempo de terminación del proyecto.

5.2 RECOMENDACIONES

Durante la investigación y desarrollo este trabajo de grado fueron conocido y creados factores importantes para el proyecto final MERIS resumidos en los siguientes ítems:

- Para una correcta implementación final del proyecto MERIS, es recomendable la utilización del sistema operativo Mobilinux en la PDA, debido a que combina perfectamente características importantes para el desarrollo de una adecuada aplicación dedicada a la atención médica, por ejemplo, incorpora un sistema de gestión de batería permitiendo alargar la duración de la misma incluso cuando estén ejecutándose aplicaciones de alto consumo, para el caso particular de MERIS ofrece soporte para WiFi y otros tipos de conectividad, ofrece también mayor velocidad que los dispositivos Symbian o Microsoft (hasta tres veces más rápido) y respuesta en tiempo real.

- Teniendo en cuenta que el tiempo es el factor más crítico en aplicaciones orientadas a atenciones prehospitalarias, se debe tener en cuenta que la interfaz

gráfica para el usuario debe mantener poca navegabilidad con el objetivo de hacer más fácil la lectura del estado de los pacientes para el personal de socorro.

- Una vez el paciente ha sido remitido a una atención hospitalaria, es importante para las entidades de atención tener una referencia del comportamiento de dicho paciente durante su atención prehospitalaria, por eso, es recomendable que el sistema final MERIS cree un historial del estado del paciente en la PDA, por medio de la SD *card* (para mayor capacidad) o en su defecto, en otro de los módulos del proyecto.

- Con el fin de no afectar la planificación de los NP, es recomendable que su implementación final conserve el manejo de un *mailbox* como interfaz entre el módulo de despliegue de información de pacientes y el resto de procesos del NP (obtención de la posición de cada sensor con respecto al NP, obtención de la información de cada paciente, etc).

- Un factor importante en cualquier sistema es la seguridad de la red, por ende, todas las tramas del protocolo PMR enviadas desde los NP hacia la PDA contienen un campo correspondiente a la dirección MAC de cada NP, es recomendable entonces la utilización de este campo para una implementación más adecuada de la seguridad en conjunto con la tabla contenedora del anillo virtual que relaciona esta dirección con la IP.

- Es recomendable que tanto el identificador como la IP de cada NP sean establecidos manualmente por el personal de instalación, con el fin de mantener un identificador único no aleatorio que no cambie cada vez que MERIS sea utilizado, evitándole confusiones a los socorristas. En este punto, es necesario mantener un registro de los nodos instalados con su respectivo identificador e IP asignada que sea consultado cada vez que un nuevo NP sea instalado para MERIS.

5.3 TRABAJOS FUTUROS

- Las redes inalámbricas aún no tienen la suficiente madurez para asegurar una comunicación con restricciones de tiempo real, especialmente las redes tipo Ad-Hoc, que son las más adecuadas para situaciones de emergencia. De esta manera, es identificado un primer trabajo futuro que implica el estudio y desarrollo de protocolos de tiempo real a nivel de enlace de datos para interfaces inalámbricas.

- RTnet es el único complemento para RTAI que permite el comportamiento de los dispositivos de red en tiempo real, pero su madurez es solamente para redes Ethernet, de esta manera se identifica un segundo trabajo futuro que consiste en el desarrollo de sus controladores de forma tal que permitan utilizar la interfaz inalámbrica en tiempo real.

- Las interfaces gráficas son elementos esenciales de los sistemas para interactuar con sus usuarios, pero al mismo tiempo no son atendidas dentro de un plazo establecido para garantizar su ejecución con restricciones de tiempo. Por lo anterior es propuesto un trabajo futuro que busque un mecanismo de atención en tiempo real a las interfaces de usuario convirtiéndolas en procesos de tiempo real no de tiempo promedio.

- Los entornos de desarrollo son herramientas importantes a la hora de elaborar software porque facilitan la depuración del código y la detección de errores. En cuanto al desarrollo de aplicaciones de tiempo real, éstos entornos no soportan su depuración debido a que mantienen una estrecha relación con el núcleo del sistema operativo sobre el que son ejecutadas las aplicaciones; de esta manera se identifica otro trabajo futuro enfocado en la elaboración de entornos de desarrollo para aplicaciones de tiempo real que mejore las condiciones de desarrollo de ellas.

- Realizar el despliegue en la PDA de la ubicación del paciente con referencia al NP correspondiente, utilizando el campo de posición enviado en la trama de datos.

- El presente trabajo de grado, durante la implementación del protocolo PMR, reservó una trama de alarma utilizada para enviar un aviso al sensor del paciente. Como trabajo futuro es necesario interpretar dicho aviso de una manera visual en el sensor y si es necesario utilizar el campo CA (de 4 bytes) de la trama con el fin de ubicar rápidamente a un paciente y para manejos más detallados de alarmas.

BIBLIOGRAFÍA

[1] Powel, B.; Harel, D., *Real-Time UML: Developing Efficient Objects for Embedded Systems*, 2nd Edición, Estados Unidos de América: Adison-Wesley, 1999.

[2] Parra, A.; Niehaus, D., *Dispositivo KURT, KU Tiempo Real. III, Congreso Hispalinux virtual, Noviembre de 2000.* Madrid: Unversidad Carlos III de Madrid, 2000.

[3] Spuri, M.; Butazzo, G., "Scheduling Aperiodic Task in Dynamic Priority", *The Journal of Real Time Systems,* Vol. 10, pp. 179-210, Marzo 1996.

[4] Powel, B., *Doing Hard Time: Developing Real-Time Systems with UML, Objects, Frameworks, and Pattern.* Estados Unidos de América: Adison-Wesley, 2001.

[5] Flynn, I; McIver, A., *Sistemas operativos.* Mexico: Cengage Learning Editores, 2001.

[6] Kalinsky, D., "Context Switch", *Embedded.com*, Febrero 2001. Disponible en web: http://www.embedded.com/story/OEG20010222S0038. [Citado: Noviembre 28, 2008]

[7] Welling, D., *Concurrent and Real-Time Programming in Java.* Chichester: John Wiley & Sons Ltd, 2004.

[8] Bishop, M., "Race Conditions, Files, and Security Flaws; or the Tortoise and the Hare Redux", Davis: University of California, Department of Computer Science, Septiembre 1995.

[9] Kalinsky, D., "Basic concepts of real-time operating systems", *LinuxDevices.com*, Noviembre 2003. Disponible en web: http://www.linuxdevices.com/articles/AT4627965573.html. [Citado: Noviembre 28, 2008]

[10] ACCESS CO., LTD. "Descripción General ALP", *Access Linux Platform*, 2008. Disponible en web: http://alp.access-company.com/lang/es/overview/index.html. [Citado: Diciembre 15, 2008.]

[11] Google Inc., "Open Handset Alliance", 2007. Disponible en web: http://www.openhandsetalliance.com/. [Accedido Enero 28, 2008]

[12] Google Inc., "What is Android", *Android*, 2008. Disponible en web: http://developer.android.com/guide/basics/what-is-android.html. [Citado: Diciembre 13, 2008.]

[13] The Apache Software Foundation, "Apache License, Version 2.0". Disponible en web: http://www.apache.org/licenses/LICENSE-2.0. [Citado: Diciembre 13, 2008.]

[14] ARM Ltd, “ARM Linux Mobile Platform”, *ARM the architecture for the digital world.* Disponible en web: http://www.arm.com/markets/mid/linux.html. [Citado: Diciembre 10, 2008]

[15] Hicks, J.; Blundell, P.; France, G.; Nelson, R.; Hovland, E., “The Familiar Project”, *handhelds.org,* 2007. Disponible en web: http://familiar.handhelds.org/. [Citado: Diciembre 11, 2008]

[16] Hicks, J.; Blundell, P.; France, G.; Nelson, R.; Hovland, E., “Familiar Linux Distribution”, *Handhelds.org - Open source for handheld device,* 2008. Disponible en web: http://handhelds.org/moin/moin.cgi/FamiliarDistribution. [Citado: Diciembre 11, 2008]

[17] LiMo Foundation, “Bienvenido a LiMo”, *LiMo Foundation.* Disponible en web: http://www.limofoundation.org/es/bienvenido-a-limo.html. [Citado: Diciembre 15, 2008]

[18] LiMo Foundation, “Why Now”, *LiMo Foundation.* Disponible en web: http://www.limofoundation.org/es/why-now.html. [Citado: Diciembre 14, 2008]

[19] MontaVista Software, Inc, “New MontaVista Mobilinux 5.0: The World's Most Advanced Mobile Operating System”, *montavista*. Disponible en web: http://www.mobilinux.com/product_detail_mob.php. [Citado: Diciembre 14, 2008]

[20] Motorola, Inc., “An Open and Flexible Platform for Motorola Mobile Devices”, *MOTOMAGX,* Junio 2008. Disponible en web: http://www.motorola.com/mot/doc/6/6864_MotDoc.pdf. [Citado: Diciembre 14, 2008]

[21] Motorola, Inc., “MOTODEV, the Motorola developer network”, *MOTODEV*. Disponible en web: http://developer.motorola.com/docstools/motodevstudio/linux/. [Citado: Diciembre 14, 2008]

[22] Openmoko, Inc., “Openmoko™ - Open. Mobile. Free”, *openmokowiki,* 2006. Disponible en web: http://www.openmoko.com/about.html. [Citado: Diciembre 13, 2008]

[23] Openmoko, Inc., “Supported devices”, *openmokowiki,* 2006. Disponible en web: http://wiki.openmoko.org/wiki/Openmoko-supported_hardware. [Citado: Diciembre 13, 2008]

[24] Openmoko, Inc., “Alarm daemon”, *openmokowiki,* 2006. Disponible en web: http://wiki.openmoko.org/wiki/Alarm_daemon. [Citado: Diciembre 13, 2008]

[25] Symbian Foundation Limited, “Symbian”, *Symbian Foundation.* Disponible en web: http://www.symbian.com/. [Citado: Diciembre 14, 2008]

[26] Symbian Foundation Limited, “Why is a different operating system needed?”, *Symbian.* Disponible en web: http://www.symbian.com/files/rx/file6383.pdf. [Citado: Diciembre 14, 2008]

[27] Canonical Ltd., “Ubuntu Mobile Internet Device (MID) Edition”, *ubuntu.* Disponible en web: http://www.ubuntu.com/products/mobile. [Citado: Diciembre 15, 2008]

[28] Microsoft Corporation, “Windows Embedded CE Overview”, *Windows Embedded.* Disponible en web: http://www.microsoft.com/windowsembedded/en-us/products/windowsce/default.mspx**.** [Citado: Diciembre 14, 2008]

[29] MontaVista Software, Inc., “MontaVista Mobilinux 5.0 - datasheet”, *montavista.* 2007. Disponible en web: http://mvista.com/download/MontaVista-Mobilinux-5-datasheet.pdf. [Citado: Marzo 14, 2009]

[30] Jackson, J., “MontaVista’s Mobilinux: Building the Case for Open, Scalable, Low-Cost Client Software”, *Yankee Group,* Noviembre 2005.

[31] 3COM Corporation, “802.11 Wireless LANS: A technology overview”, Tech. Rep., 2005.

[32] LAN/MAN Committee of the IEEE Computer Society, “IEEE Std 802.11”, New York, Tech. Rep., 2007.

[33] Mosquera, V., “Redes inalámbricas”, Popayán: SENA, 2007.

[34] WNDW Project, *Redes Inalámbricas en los Países en Desarrollo,* 3ra Edición. Londres: Hacker Friendly LLC, 2008.

[35] Lehembre, G., “Seguridad WiFi – WEP, WPA y WPA2”, *Hakin9,* 2006. Disponible en web: http://www.hakin9.org/. [Citado: Marzo 14, 2009]

[36] Novella, M.; Ignacio, J., *Técnicas de Acceso al Medio,* Madrid: Universidad Carlos III de Madrid, 2002.

[37] Mohedano, N., *Análisis de técnicas de acceso al medio avanzadas basadas en colas distribuidas y mecanismos Cross-Layer para sistemas de comunicación inalámbricas*. Catalunya : Universitat Politécnica de Catalunya, 2007.

[38] Cisco Systems, Inc. “Capa de enlace de datos” en Aspectos Básicos de Networking. Cisco Networking Academy, 2007.

[39] Melgarejo, P., *Curso de Redes de Datos.* Chile: Universidad de Concepción, 2000.

[40] Ergen, M.; Lee, D.; Sengupta, R; Varaiya, P., "WTRP-Wireless Token Ring Protocol", *IEEE Transactions on Vehicular Technology,* Vol. 53.

[41] Lee, D., "Wireless Token Ring Protocol", tesis de maestría, University of California, Berkeley, 2001.

[42] Yigitbasi, N., *A control plane for prioritized real-time communications in wireless token ring networks.* Istanbul: Istanbul Teknik Üniversitesi, 2008.

[43] Donatiello, L.; Furini, M., "Ad Hoc Networks: A Protocol for Supporting QoS Applications", Proceedings of The International Parallel and Distributed Processing Symposium, 2003.

[44] Sheltami, T.; Mouftah, H., "Clusterhead Controlled Token for Virtual Base Station On-demand in MANETs", Proceedings of the 23rd International Conference on Distributed Computing Systems. Washington: IEEE Computer Society, 2003.

[45] Brenner, P., "A Technical Tutorial on the IEEE 802.11 Protocol", Tech. Rep., BreezeCOM Wireless Communication, 2000.

[46] López, D, "CICLOPE RT_DISPLAY: Diseño de un rótulo luminoso con fines docentes", tesis de pregrado, Universidad Politécnica de Madrid, Segovia, 2005.

[47] Camara, C, *Linux en tiempo real*, España: Universidad de Sevilla, 2004.

[48] Mantegazza, P., *RTAI Programming Guide 1.0,* Lindon: Dipartimento di Ingegneria Aerospaziale, Politecnico di Milano Real Time Application Interface, 2000.

[49] de Boer, H. "Porting the Xenomai real-time Firewire driver to RTAI", University of Twente, Netherlands, Tech. Rep. 024CE2007, 2007.

[50] Xenomai Project, "Xenomai: Real-Time Framework for Linux", Mayo 2009. Disponible en web: http://www.xenomai.org/index.php/Main_Page. [Citado: Mayo 10, 2009]

[51] García, L.; Lópex, M.; Lorenzo, J., "Entorno de pruebas en tiempo real para validación de controladores y sistemas FDI robustos", XXV Jornadas de Automática de la Universidad de Cádiz. Cádiz, España, 2004.

[52] Barbalace, A.; Luchetta, A.; Manduchi, G.; Moro, M.; Soppelsa, A.; Taliercio, C., "*Performance Comparison of VxWorks, Linux, RTAI and Xenomai in a Hard Real-time Application",* IEEE Transactions Nuclear Science, Vol. 55, pp. 435-439, Febrero 2008.

[53] IBM Company, “IBM Rational Unified Process (RUP)”, *IBM.* Disponible en web: http://www-01.ibm.com/software/awdtools/rup/. [Citado: Junio 1, 2009]

[54] Plummer, D., “An Ethernet Address Resolution Protocol or Converting Network Protocol Addresses”, *IETF Tools.* Disponible en web: http://tools.ietf.org/html/rfc826. [Citado: Junio 11, 2009]

[55] The Glade Project, “What is Glade?”, *Glade - A User Interface Designer.* Disponible en web: http://glade.gnome.org/. [Citado: Agosto 4, 2009]

[56] Ministerio de la Protección Social, *Guías Básicas de Atención Médica Prehospitalaria.* Bogotá, 2005.

[57] Rodríguez, A.; Peláez, M.; Jimenez, L., *Manual de triage prehospitalario.* Barcelona.

[58] National Highway Traffic Safety Administration, “Star of Life”, *NHTSA Office of EMS.* Disponible en web: http://www.ems.gov/vgn-ext-templating/ems/sol/index.htm. [Citado: Octuble 15 , 2009]