UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Título: Arquitectura de Sistemas Telemétricos.
Autores: Pablo Andrés Saccani.
Director: Silvia Gordillo.
Codirector: -
Asesor profesional: -
Carrera: Licenciatura en Sistemas.

Resumen

Las aplicaciones telemétricas han crecido masivamente en las últimas dos décadas, han crecido tanto como lo ha hecho Internet y la red de comunicación GSM. Estas componentes, junto con sus protocolos de comunicación, permitieron que los ingenieros reutilicen canales de comunicación abaratando de manera notoria los costos de estos sistemas.

Las primeras secciones describen las componentes intervinientes en los sistemas telemétricos, luego se implementa una solución genérica de comunicación que puede ser utilizada para cualquier tipo de sistema telemétrico de similares característica. La solución consiste en la creación de un esquema de sockets que dialogan con el transmisor mientras dura la comunicación. La comunicación establecida puede ser unidireccional o bidireccional.

También se implementa un simulador que permite representar diferentes escenarios de pruebas, sin la necesidad de contar con transmisores físicos.

Palabras Claves

Sistemas telemétricos, telemetría, sockets, emulador, seguimiento satelital, monitoreo, GSM, GPS, trama de datos, UDP, TCP, TCP-IP, modelo OSI, trilateración.

Conclusiones

La solución implementada puede ser utilizada como un canal de comunicación para enviar y recibir información prácticamente desde y hacia cualquier lugar del mundo, esto es, cualquier elemento que se monitoree localmente, ya sea por un medio cableado, WIFI, Bluetooth, etc., puede ser monitoreado o controlado desde sitios remotos con un costo muy bajo, sólo se requiere Internet y red GSM.

Trabajos Realizados

_Selección de bibliografía utilizada para: introducción a Sistemas Telemétricos, coordenadas GPS, trilateración, conexiones GSM.

_Búsqueda de equipos GSM emisores por GPRS para detallar diferentes características técnicas.

_Entrevistas con profesionales experimentados en telemetría (hidráulica, silos, seguimiento de móviles, depósitos fiscales).

_Desarrollo de esquema de sockets TCP que dialogan con transmisores y un simulador de equipos para pruebas.

Trabajos Futuros

_ Desarrollo de esquema de sockets UDP.

_Optimización en el protocolo de comunicación con ACK.

_Investigación sobre reparación de móviles a distancia.

_Integración de equipos GSM con monitores de siembra.

Fecha de la presentación: Junio 2014

Agradecimientos

A mis padres, por su educación, contención y confianza. A mi hermano, por la confianza en haber seguido la misma ruta.

A mis abuelas, en especial Nely por ser la persona que me marcó la vida con su cariño y ternura extrema. Por las rondas de mates y noches eternas que se quedaba junto a nosotros cuando íbamos a estudiar a su casa. Imposible no recordarla en algún instante del día.

A Jor e hijas, por los momentos robados dedicados al estudio y trabajo.

Al ex Centro Universitario Regional Junín (CURJ) por la posibilidad de realizar mi primera carrera Universitaria y permitir conocer compañeros que hoy son amigos incondicionales. También por haberme permitido conocer a Marcelo Guruceaga, mi socio.

A la Universidad Nacional de La Plata (UNLP) por permitir continuar con mis estudios y conocer la ciudad de La Plata.

A mi país por haberme permitido estudiar es una Universidad pública de alto nivel, espero estar devolviendo algo de todo ésto sábado a sábado desde hace ya más de 7 años.

Índice

CAPÍTULO I: INTRODUCCIÓN AL CONCEPTO DE TELEMETRÍA.

El objetivo de este capítulo es introducir el concepto de Telemetría, describir sus componentes y aplicaciones. También se esquematizan los puntos más relevantes de los sistemas telemétricos que se tratarán en detalle en los capítulos siguientes.

Para introducir el concepto de Telemetría se han seleccionado definiciones de diversas fuentes con el objetivo de establecer un análisis comparativo y redactar una definición integral.

Resulta convenientemente útil aclarar que su etimología proviene de la conjunción de las palabras TELE (a distancia) y METRÍA (medición).

Definiciones:

"*Sistema de medida de magnitudes físicas que permite transmitir éstas a un observador lejano.*" (Real Academia Española[1]).

"La Telemetría es el uso de equipos eléctricos o electrónicos para detectar, acumular y procesar datos físicos en un lugar, para después transmitirlos a una estación remota donde pueden procesarse y almacenarse." (BRICEÑO MÁRQUEZ, 2005: 226).

"La telemetría, se entiende como una tecnología que permite la medición remota de magnitudes físicas y el posterior envío de la información hacia el operador del sistema." (MARCANO/MARCANO, 2012:6).

De acuerdo a las definiciones precedentes, podemos describir a la Telemetría como:

Un sistema de medida de magnitudes físicas como por ejemplo: el voltaje, la corriente, la presión, el torque, coordenadas geoespaciales, etc., que permite transmitir los datos captados a una estación remota donde puedan ser procesados y almacenados. Cuando se hace referencia a estación remota, no necesariamente tiene que estar distante, sino que la Telemetría también es utilizada en distancias próximas donde las variables a monitorear son de difícil acceso, por ejemplo biomedicina, y/o se encuentran el lugares peligrosos como pueden ser desactivación de explosivos o pozos para la obtención de petróleo o hidrocarburos.

Una extensión de los Sistemas Telemétricos son los Sistemas de Telecontrol, donde además de procesar datos distantes, permiten enviar comandos para modificar las condiciones de operación. El control y envío de comandos lo puede ejercer un operador o un sistema automático.
En la automatización de procesos industriales un concepto muy utilizado es el de Sistemas SCADA (Supervisor y Control and Data Adquisition). [1]

[1] Vigésima segunda edición, en www.rae.es, consultado el 15/12/2012.

Elementos constitutivos de un sistema de telemetría.

Un sistema de Telemetría está compuesto básicamente por los siguientes componentes:

- Transmisor.
- Medio de Transmisión.
- Receptor.

Transmisor: Transductor y Unidad Terminal.

Un transductor es un equipo encargado de convertir un fenómeno físico cambiante a una señal eléctrica proporcional. Casi todos los fenómenos físicos disponen de equipos (transductores) que pueden convertir en señales eléctricas los valores. Entre los fenómenos más importantes tenemos: temperatura, presión, flujo, velocidad, aceleración, torque, posición angular, fuerza, humedad, voltaje, posición geoespacial.

La unidad terminal es un dispositivo que adapta los datos medidos por el transductor de manera que puedan ser transmitidos como señal codificada utilizando algún tipo de canal de transmisión.

Medio de transmisión.

Con relación a los canales o medios de transmisión, los más comúnmente utilizados para la medición remota son:

Par Trenzado: se trata de dos hilos de cobre entrelazados (por ejemplo: cable telefónico).

Cable Coaxial: se trata de dos conductores, donde uno es el eje central y el otro cubre al aislante del primero en forma de cubierta cilíndrica.

Fibra Óptica: se basa en un medio cristalino que permite la propagación de la luz, la cual no se dispersa sino que se mantiene dentro de la fibra por las características ópticas especiales de la misma.

Radio: Se refiere a la transmisión de información mediante ondas electromagnéticas. Son los sistemas de radio comunes dedicados a un servicio específico. Uno de los medios de transmisión utilizados en la actualidad para monitorear elementos remotos, móviles o fijos, es la tecnología celular. Dicha tecnología ha evolucionado en generaciones, cada una con un conjunto de características que la definen. La primera generación involucró a los servicios analógicos, la segunda a los servicios digitales y la tercera se centró en los servicios multimedia.

Criterio	Primera Generación	Segunda Generación	Tercera Generación
Servicios	Voz	Voz y Mensajería corta	Voz y Datos
Calidad del Servicio (QoS)	Baja	Alta	Alta
Nivel estandarización	Bajo	Fuerte	Fuerte
Velocidad de transmisión	Baja	Baja	Alta
Tipo de Conmutación	Circuitos	Circuitos	Paquetes (IP)

Tabla 1. Comparación entre diferentes de generaciones móviles (PACHÓN DE LA CRUZ, 2004:15).

Cada generación tuvo una arquitectura que la representa: GSM (Global System for Mobile communications) representa a la segunda, GPRS (General Packet Radio Service) a la segunda y media, es decir, a la transición entre la segunda y tercera generación que es UMTS (Universal Mobile Telecommunications System).

GSM: principios de los años ochenta existían estándares diferentes en diversos países con los correspondientes problemas de incompatibilidades. En 1982 el CEPT (Conference of European Postand Telecommunications) estableció un grupo paneuropeo que se denominó GSM (Groupe Speciale Mobile) con el objetivo de desarrollar un nuevo sistema inalámbrico. En 1989, la responsabilidad fue transferida al ETSI (European Telecommunications Standards Institute) que denominó al proyecto como Global System for Mobile Communications (GSM).

El proyecto GSM tenía las siguientes premisas:

- Itinerancia (roaning) internacional.
- Soporte para la introducción de nuevos servicios.
- Eficiencia espectral.
- Compatibilidad con ISDN (Integrated Services Digital Network). [2]

La evolución de GSM posee tres fases de evolución, la fase 1, en la que se produjeron sus especificaciones; la fase 2, en la que se propuso la inclusión de servicios de datos y de fax; y finalmente, la Fase 2+, en la que se realizan mejoras sobre la codificación de voz y se implementan servicios de transmisión de datos avanzados, entre ellos GPRS (General Packet Radio Service) y EDGE (Enhanced Data rates for GSM Evolution).

GSM es un sistema de conmutación de circuitos, diseñado originalmente para voz, al que posteriormente se le adicionaron servicios de datos:

- Mensajes cortos: entrega de mensajes de texto de hasta 160 caracteres.
- Datos GSM: que permite una tasa de transferencia de 9.6 kbps. [3]

GSM-GPRS: es una red de datos que utiliza la infraestructura de la red GSM para permitir la transmisión de paquetes de datos a tasas que fluctúan entre los 9.6 y los 171 Kbps (alta velocidad). Aunque se intenta reutilizar la red GSM existente tanto como sea posible, resulta necesario adicionar algunos

nuevos elementos de red, interfaces y protocolos, para manejar este nuevo tipo de tráfico y construir de esta manera una red móvil celular de paquetes.

UMTS: ha sido presentada como la culminación de la convergencia de Internet y las redes móviles, en ella, los usuarios tendrán la posibilidad de acceder a contenidos y servicios multimedia de banda ancha independientemente del lugar donde se encuentren.

La especificación de UMTS fue dividido en fases hasta alcanzar el objetivo final: una red integrada de servicios multimedia independientes de la posición del usuario:

- En la fase 1: se logra una evolución lógica desde las arquitecturas de segunda generación.
- En la fase 2: lo que se logra es una completa revolución: reemplazar la componente de conmutación de circuitos por una red basada completamente en conmutación de paquetes denominada IP UMTS network architecture. En esta propuesta, el protocolo IP adquiere cada vez mayor importancia hasta convertirse en el protocolo para el transporte, tanto de la información del usuario (contenido multimedia), como de la información de control y de señalización, de ahí la denominación de una red «todo IP».

UMTS proveerá servicios de voz y datos, en eso coincide con la red GSM/GPRS. Estos servicios serán provistos a diferentes tasas según el ámbito en el que se ofrezcan, en conexiones satelitales y servicios rurales en exteriores, la tasa será de 144 Kbps; en servicios urbanos en exteriores, la tasa será de 384 Kbps; mientras que en servicios de interiores o de exteriores de bajo rango de distancias se podrán alcanzar tasas de hasta 2 Mbps, en esto difiere con la red GSM/GPRS. (PACHÓN DE LA CRUZ, 2004).

Receptor.

El equipo receptor es un dispositivo capaz de decodificar la señal recibida de la unidad remota y mostrarla en algún formato adecuado para su análisis y almacenamiento.

Ejemplos de sistemas telemétricos.

La telemetría tiene un amplio campo de uso, entre los que podemos mencionar (MARCANO/MARCANO, 2012):

- En el área militar, en todo lo relacionado a operaciones de medición de dispositivos o equipos.
- En el área de telecomunicaciones, para las comunicaciones satelitales y monitoreo de equipos de comunicaciones.
- En el área médica, en lo relacionado con la biomedicina (sensores biológicos que se introducen en el cuerpo humano encargados de transmitir información a detectores externos).
- En monitoreo del medio ambiente mediante estaciones meteorológicas: precipitaciones, orientación del viento, humedad, presión atmosférica, etc.
- En la supervisión de niveles de líquidos de represas, ríos, contenedores, oleoductos y tuberías (medición de parámetros como: temperatura, presión, caudales).
- En fábricas, oficinas y residencias, para el monitoreo del uso de energía o fenómenos derivados como la temperatura.

- Monitoreo satelital de vehículos, donde además de datos geoespaciales tomados por un GPS, se capturan datos del entorno como por ejemplo: caudalímetro, botón de pánico, apertura de puertas, temperaturas del motor, etc.[2]

[2] En www.skycontrol.com.ar, consultado el 10/01/2013.

Sistemas Telemétricos

El siguiente gráfico muestra los componentes intervinientes en sistemas telemétricos distantes y denota la secuencia de la información desde que es obtenida por el transductor y enviada por la unidad terminal hasta ser recepcionada y almacenada por el receptor. También involucra cómo diferentes aplicaciones, que pueden ser utilizadas en dispositivos fijos o móviles, consumen la información almacenada. Ejemplos de estos sistemas telemétricos son: monitoreo satelital de móviles, silos, cisternas, digestores, etc.

En los siguientes capítulos se estudiarán los componentes intervinientes vinculando explicaciones teóricas con experiencias prácticas. Se llegará a un nivel detallado y ejemplos de implementación en las secciones donde intervienen las componentes software.

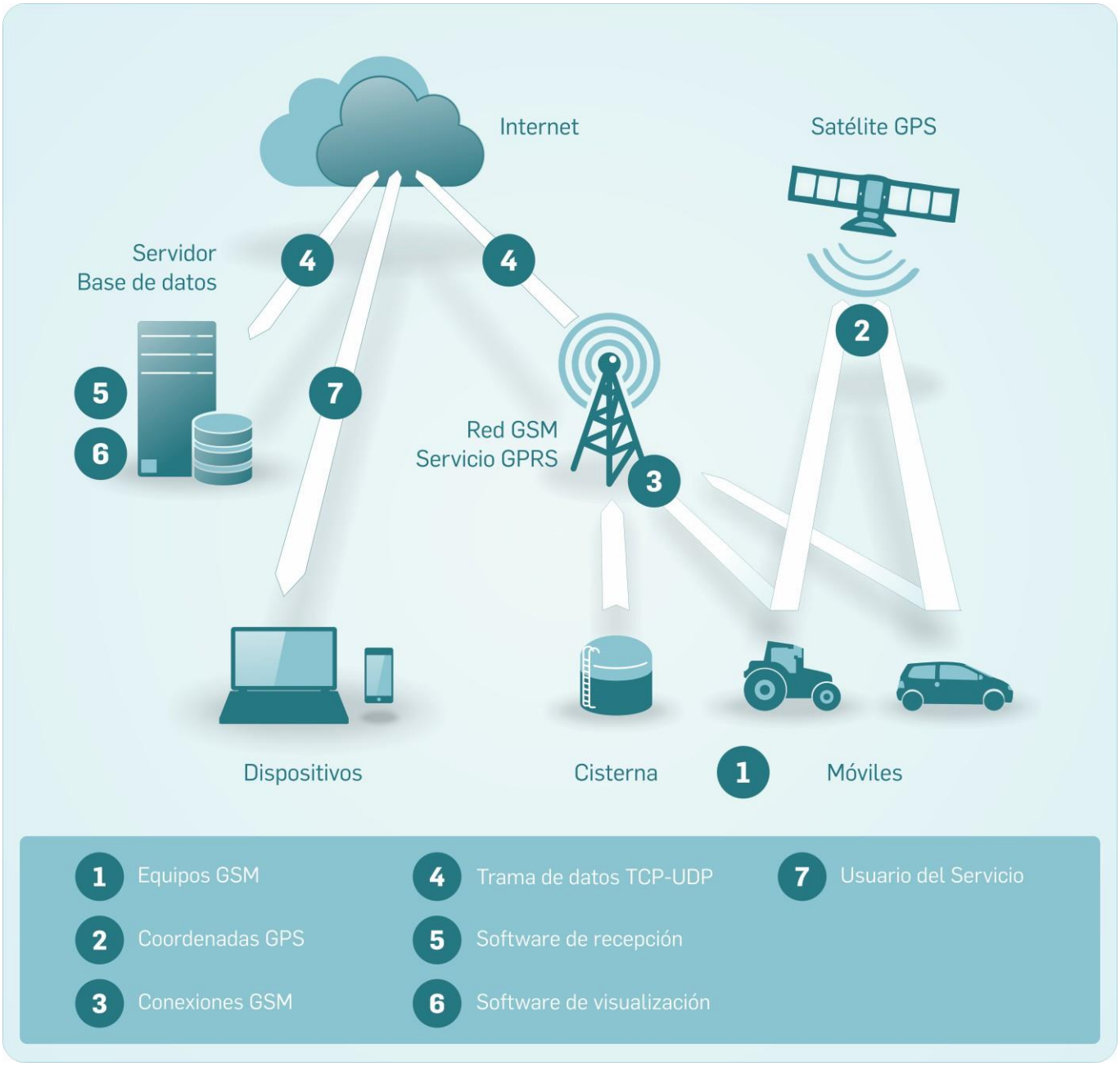


Figura 1.1: componentes intervinientes en sistemas telemétricos distantes.

CAPÍTULO II: HARDWARE Y COMUNICACIONES.

En este capítulo se presentan las componentes principales de los equipos GSM y las comunicaciones utilizadas en los sistemas telemétricos.

Los emisores móviles utilizan el GPS como componente primordial para obtener la georreferencia. En este capítulo se describe cómo funciona el GPS y cómo la triangulación permite determinar la ubicación.

También se describe brevemente cómo funciona la red GSM y la capa de transporte del modelo OSI, en especial los protocolos TCP y UDP.

EQUIPOS GSM.[4]

Identificación de componentes.

Los componentes externos que integran un equipo GSM son (de izquierda a derecha en la imagen):

- Alimentación
- GPS con base magnética
- Antena GSM con cable coaxial

Figura 2.1. Equipo GSM SK1000 para monitoreo satelital (suministrada por SkyControl SH)

Los principales componentes internos son:

Módulo GPS (amarillo): utiliza trama NMEA por puerto serie (estándar). Distorsión por temas de seguridad, por ejemplo, georeferenciación de misiles. Es un servicio gratuito. No se satura por el incremento de dispositivos debido a que es un servicio de Radio Frecuencia (RF) unidireccional.

SIM Card (naranja con un número).

Microprocesador (izq. de la SIM card).

Memoria Flash (debajo entre el microprocesador y la SIM Card): almacén de datos cuando no está disponible el servicio GPRS.

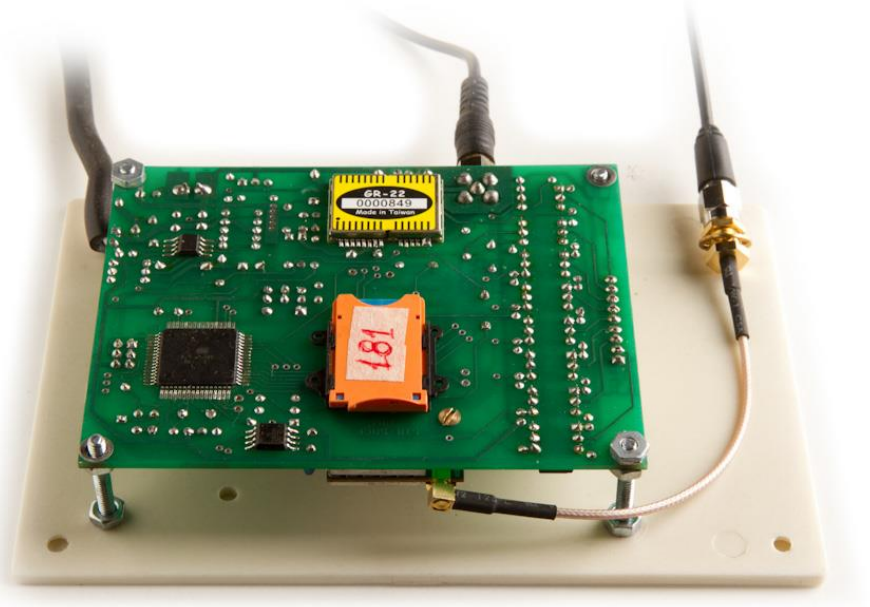


Figura 2.2. Plaqueta GSM SK1000 para monitoreo satelital (suministrada por SkyControl SH)

Conexión del emisor con el medio y el receptor.

1. Cuando se enciende el equipo el módem se registra en la red de la prestadora. Lo anterior ocurre si se tiene un SIM conectado con el servicio GPRS activo. El microprocesador principal consulta periódicamente por el estado del registro.
2. Una vez registrado el microprocesador principal le instruye al módem para conectarse con el GATEWAY de la prestadora. Si este paso resulta exitoso, se le asigna una IP al equipo (IP privada, pertenece a la prestadora del servicio) obteniendo acceso a Internet.
3. Se abre una conexión socket con un servidor TCP o UDP que esté a la escucha (IP+PUERTO).
4. Comienza el envío de tramas.

EL MÓDEM MANEJA LAS CAPAS DE TRANSPORTE Y RED. EL MICROPROCESADOR CENTRAL LA CAPA DE APLICACIÓN.

Conceptos.

GSM / GPRS: GSM (Groupe Special Mobile) es la red, GPRS (Servicio General de Paquetes de Radio) es uno de los servicios que da la red. Otro es el SMS, MMS.

Watchdog Timer: es un temporizador de 16 bit que puede ser usado como watchdog (perro guardián) o por intervalos de tiempo. La principal función del watchdog timer (WDT) es reiniciar el procesador después de que ocurra una falla o problema de software, o después de un intervalo de tiempo determinado generado por el programador, en cuyo caso se reinicia el procesador o el programa en ejecución.

Si el watchdog timer no se emplea en ninguna subrutina puede ser configurado como un temporizador de intervalos y puede generar interrupciones en los intervalos de tiempo seleccionados.

Características de Equipos GSM.

Se presentan puntos relevantes de varias especificaciones técnicas de equipos GSM utilizados para transferir información desde una entidad móvil o fija:

- Microcontrolador con puertos serie.
- Módem GSM-GPRS con antena interna o externa.
- Tarjeta SIM.
- Memoria interna no volátil.
- Leds de señalización.
- Entradas / salidas digitales y analógicas.
- Gabinetes rígidos e impermeables.
- Inclusión (o no) de batería interna.
- En el caso de equipos móviles:
- GPS con antena interna o externa.

Ejemplos de características de Equipos GSM en el mercado.

A continuación se muestran las características técnicas del equipo SK1000 del Sistema de Monitoreo Satelital SkyControl[3]:

- Microcontrolador RISC con 2 puertos serie, sistema de supervisión de la alimentación y watchdog.
- GPS con batería de backup y antena externa con base magnética.
- Módem GSM-GPRS cuatribanda con antena externa autoadhesiva.
- Memoria no volátil para registro de recorridos en ausencia de cobertura GPRS. Hasta 12000 puntos de registro.
- Tarjeta SIM intercambiable, funciona con todas las operadoras GSM.
- Parámetros operativos configurables según el móvil: Transmisiones ante variaciones de tiempo, posición y velocidad del móvil.
- Led de señalización de estado del GPS y transmisiones GPRS.
- Admite la conexión de un pulsador de pánico / alerta.
- Entradas Digitales Optoacopladas: 8. Entradas Analógicas: 2.
- Amplio rango de alimentación: 9 a 26 Vcc. Consumo promedio: < 100mA en 12Vcc

[3] En www.skycontrol.com.ar/equipos/, extraído el 03/03/2013.

El siguiente es otro ejemplo de equipos utilizados por depósitos fiscales para el seguimiento de contenedores y mercadería. Además de tener mejoras en la comunicación como contingencia y algoritmos con ACK, dispone de trabas que bloquean y desbloquean desde un sistema remoto central: [5]

- Placa CPU basada en microcontrolador Atmel de última generación (ATmega / ATxmega).
- Memoria interna para almacenar configuraciones (no volátil).
- Memoria externa de datos para almacenar recorridos (no volátil).
- Entradas y Salidas (Analógicas y Digitales) según requerimientos de la aplicación.
- Conectividad USB para configuración local y descarga de datos.
- Indicadores luminosos de estado: servicio activo, nivel de baterías, recepción GPS, enlace GSM/GPRS.
- Receptor GPS de alta sensibilidad con antena incorporada.
- Módem GSM/GPRS con antena incorporada.
- Operación GSM/GPRS con doble SIM (redundancia).
- Comunicación con el Servidor: actualización periódica y automática del posicionamiento, comandos de apertura y cierre de viaje, generación de alarmas ante apertura de la carga o bien del gabinete.
- Gabinete apto intemperie (estanqueidad + protección UV). Conectores externos para fibra óptica y carga de batería. Sistema de montaje rápido en lateral de contenedores o chasis.
- Incluye Batería de 6V, Electrolito Absorbido: autonomía mínima de 3 días (ajustable a lógica del negocio)
- Modulador y Demodulador Óptico.
- Cuerda con fibra monomodo en su interior, tramo de 20 mts.
- Dimensiones: 15 x 10 x 5 cm.

Precintos con GPS para monitorear el seguimiento de contenedores.

Figura 2.3. Precinto de seguridad para contenedores (en www.multistore.com.ar, extraído el 15/05/2013)

Equipos para el monitoreo de mercadería.

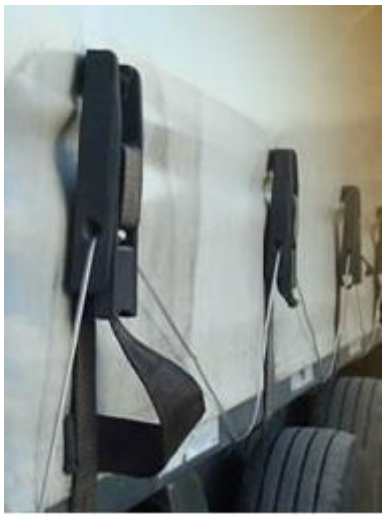


Figura 2.4. Soga de seguridad para traslado de mercadería (suministrado por Depósito Fiscal Murchinson)

Evolución y mejoras.

A medida que los sistemas de telemetría utilizados para monitoreo satelital fueron evolucionando, incorporaron mejoras relacionadas con mecanismos de redundancia, cuyo objetivo radicó en disminuir la pérdida de información. Entre los puntos coincidentes de mejora podemos nombrar:

Gabinetes impermeables y rígidos útiles para equipos que están en contacto con el agua, zonas húmedas y móviles con mucha vibración.

Memoria interna para almacenamiento de datos: utilizada por equipos instalados en unidades móviles. Si al momento de enviar los datos se encuentra con ausencia de cobertura GPRS los deposita en la memoria interna, al encontrar señal se envían los datos. Con esto se logra el almacenamiento de recorridos históricos completos.

Doble SIM. Posibilita instalar dos SIM de diferentes telefónicas en forma simultánea, si no se puede transmitir las tramas por uno se intenta transmitir por el otro.

Posibilidad de guardar los recorridos en memoria externa extraíble. Útil para auditar recorridos y reconstruir aquellos que por alguna circunstancia no han podido ser transferidos.

Existen elementos a monitorear que pueden no tener señal GSM. Para estos casos existen equipos que se comunican directamente con el satélite, sin pasar por antenas GSM.

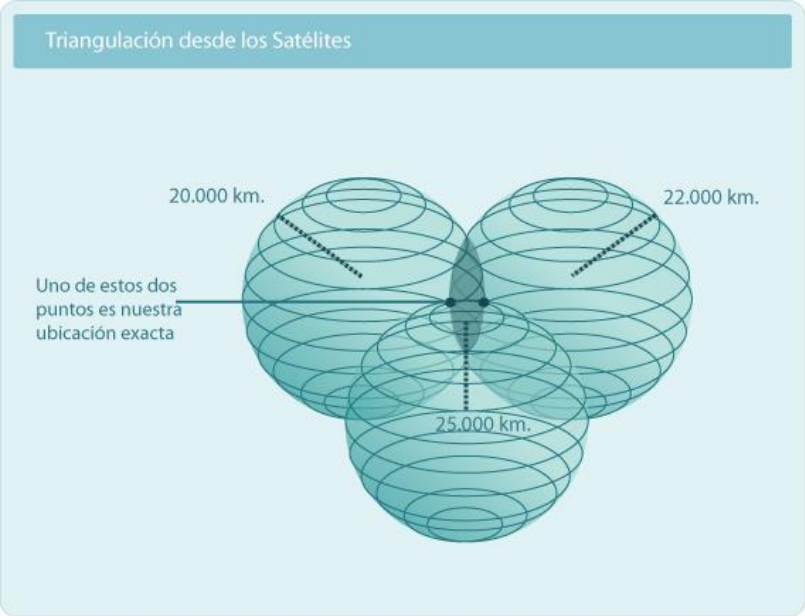


Figura 2.5. Triangulación.

Coordenadas GPS.

El Sistema de Posicionamiento Global (G.P.S.) fue desarrollado para determinar posiciones en tierra, mar, aire o en el espacio, partiendo de las posiciones conocidas de una constelación de satélites. El sistema fue desarrollado, instalado y actualmente operado por el Departamento de Defensa de los Estados Unidos.

Funcionamiento del Sistema GPS:

Uso de la triangulación para determinar la ubicación. [6]

Gracias al posicionamiento estratégico de tres satélites en el espacio, el Sistema GPS permite una medición exacta de un objeto en cualquier parte del planeta a partir de la triangulación de información entre los dispositivos mentados.

La idea es la siguiente: la distancia entre el primer satélite y el punto del que queremos saber las coordenadas es de 20.000km, el universo de puntos con un satélite se reduce a una esfera. A continuación se mide la distancia a un segundo satélite y nos resulta que estamos a 22.000km, por lo que el universo de puntos se reduce a un círculo. Siguiendo con la misma técnica hacia un tercer satélite, si nos encontramos a 25.000km, la intersección entre las 3 esferas reduce aún más el universo de puntos a 2. En la práctica, para mediciones en el planeta, uno de los puntos se puede descartar por estar muy alejado. En consecuencia, utilizando 3 satélites cuyas posiciones son conocidas, podemos ubicar la posición de un punto en el planeta. [7]

Obteniendo la distancia entre un satélite y la ubicación.

Para determinar la distancia se utiliza la medición de la velocidad de una señal emitida por el satélite y recepcionada por el receptor GPS. La fórmula utilizada es muy sencilla y conocida:

Velocidad (km/h) * Tiempo (h) = Distancia (km)

En este caso, por ser una señal de radio, la velocidad utilizada es la de la luz que es aproximadamente 300.000 km/seg. Queda obtener el tiempo exacto para obtener la distancia precisa a cada satélite y utilizando trilateración obtendremos el lugar donde estamos posicionados.

La señal que utilizan los satélites y el receptor es conocida con el nombre "Código Pseudo Aleatorio" (Pseudo Random Code). Físicamente se trata de una secuencia muy complicada de bits 0 o 1, la palabra Aleatorio está relacionada porque la señal es tan complicada que parece un ruido eléctrico generado por el azar. El satélite y el receptor GPS, deben ser capaces de sincronizar sus Códigos Pseudo Aleatorios para que el sistema funcione.

Esta señal es importante por los siguientes motivos:

- La complejidad del código ayuda a asegurar que el receptor de GPS no se sintonice accidentalmente con alguna otra señal.
- Cada uno de los satélites tiene su propio y único Código Pseudo Aleatorio, cuya complejidad garantiza que el receptor no se confunda de satélite. De esa manera, también es posible que todos los satélites trasmitan en la misma frecuencia sin interferirse mutuamente.

Los diseñadores del sistema GPS encontraron una solución que permite resolver el problema con relojes mucho menos precisos en los GPS receptores, y es lo que permite el bajo coste de los mismos.

El secreto para obtener un tiempo perfecto es efectuar una medición satelital adicional.

Con relojes imperfectos, una cuarta medición, efectuada como control cruzado, no intersectará con los tres primeros. De esa manera la computadora de nuestro GPS detectará la discrepancia y atribuirá la

diferencia a una sincronización imperfecta con la hora universal. Dado que cualquier discrepancia con la hora universal afectará a las cuatro mediciones, el receptor buscará un factor de corrección único que siendo aplicado a sus mediciones de tiempo hará que los rangos coincidan en un solo punto.

Una consecuencia es que cualquier GPS debe ser capaz de sintonizar al menos cuatro satélites de manera simultánea. En la práctica, casi todos los GPS en venta actualmente, acceden a más de seis y hasta a doce.

Composición del Sistema GPS: (HOFMANN-WELLENHOF, LICHTENEGGER, COLLINS, 1997)

El GPS se divide en tres segmentos: **segmento espacial, segmento de control** y **segmento usuario**.

Figura 2.6. Constelación de satélites GPS (en sitio oficial de la NASA).

Video puesto a disposición para la descarga en el sitio de la NASA (extraído el 03/04/2014): How Do Global Positioning Systems, or GPS, Work?
Ver video

El **segmento espacial** contiene los satélites emisores de las señales, conocidos como Constelación NAVSTAR (NAVigation Satellite Timing And Ranging), que consta de un mínimo de 24 satélites dispuestos en 6 planos orbitales. Dispone además de algunos satélites de recambio, por si alguno de los que están en funcionamiento fallasen.

Los satélites están a una altura de 20.200 kilómetros, y actúan como un punto de referencia conocido, transmitiendo información con dos frecuencias de referencia L1=1575.42 MHz y L2=1227.60 MHz. Sobre estas frecuencias se modulan 2 códigos, llamados C/A y P. El código C/A, (Clear/Access o Course/Acquisition), está disponible para todos los usuarios mientras que el código P (Precision-code), se reserva para usos militares.

Los satélites están distribuidos de manera que garanticen al menos 4 satélites visibles desde cualquier punto del mundo, las 24 horas del día. La energía eléctrica que requieren para su funcionamiento la adquieren a partir de dos paneles compuestos de celdas solares adosados a sus laterales.

Figura 2.7. The Jason-2 satellite orbits Earth. Imagen puesta a disposición en el sitio de la NASA (extraído el 21/04/2014): sección forstudent, What Is a Satellite?

El **segmento de control** es quien gobierna el sistema, a través de 5 estaciones situadas en Tierra con gran precisión. Estas estaciones son Hawai, Colorado Springs, Isla de Ascensión en el Atlántico Sur, Diego García en el Índico y Kwajalein en el Pacífico Norte. Estas estaciones realizan un seguimiento continuo de los satélites y pueden realizar cambios en la información transmitida por los satélites.

Por último, el **segmento usuario** está constituido por todos los equipos utilizados para la recepción de las señales emitidas por los satélites y empleados para el posicionamiento, para la navegación o para la determinación del tiempo con precisión.

Fiabilidad de los datos.

Debido al carácter militar del sistema GPS, el Departamento de Defensa de los EE.UU. se reservaba la posibilidad de incluir un cierto grado de error aleatorio, que podía variar de los 15 a los 100 metros. La llamada disponibilidad selectiva (S/A)[4] fue eliminada el 2 de mayo de 2000. Aunque actualmente no aplique tal error inducido, la precisión del sistema GPS depende del número de satélites visibles en un momento y posición determinados.

Con un elevado número de satélites siendo captados (7, 8 ó 9 satélites), y si éstos tienen una geometría adecuada (están dispersos), pueden obtenerse precisiones inferiores a 2,5 metros en el 95% del tiempo. Si se activa el sistema DGPS (o GPS diferencial), la precisión mejora siendo inferior a un metro en el 97% de los casos.

DGPS o GPS diferencial.

El DGPS o GPS diferencial, es un sistema que proporciona a los receptores de GPS correcciones de los datos recibidos de los satélites GPS, con el fin de proporcionar una mayor precisión en la posición calculada.

El fundamento radica en el hecho de que los errores producidos por el sistema GPS afectan por igual (o de forma muy similar) a los receptores situados próximos entre sí. Los errores están fuertemente correlacionados en los receptores próximos.

Un receptor GPS fijo en tierra (referencia) que conoce exactamente su posición puede calcular los errores producidos por el sistema GPS, comparándola con la suya, conocida de antemano. Este receptor transmite la corrección de errores a los receptores próximos a él, y así estos pueden, a su vez, corregir también sus errores.

- Con el DGPS se pueden corregir en parte los errores debidos a:
- Disponibilidad selectiva (eliminada a partir del año 2000).
- Propagación por la ionosfera - troposfera.
- Errores en la posición del satélite (efemérides).
- Errores producidos por problemas en el reloj del satélite.

Para consultar los valores en metros que pueden producir los errores: *Ver ANEXO 1: "Medición de errores del Sistema GPS".*

Hemos utilizado una presunción que no es del todo exacta para calcular la distancia, la velocidad de la luz sólo es constante en el vacío. Una señal de GPS pasa a través de partículas cargadas en su paso por la ionosfera y luego al pasar a través de vapor de agua en la tropósfera pierde velocidad. Hay maneras de minimizar este tipo de error que exceden las limitaciones de este trabajo.

[4] Disponibilidad selectiva: Degradación intencionada de la señal GPS con el fin de evitar la excesiva precisión de los receptores GPS comerciales modernos.

Para que las correcciones DGPS sean válidas, el receptor tiene que estar relativamente cerca de alguna estación DGPS; generalmente, a menos de 1000 km. Las precisiones que manejan los receptores diferenciales son de centímetros, por lo que pueden ser utilizados incluso en ingeniería.

Integración con la telefonía móvil.

Actualmente dentro del mercado de la telefonía móvil la tendencia es la de integrar tecnología GPS dentro de dispositivos móviles. El uso y masificación del GPS está particularmente extendido en los teléfonos móviles inteligentes y tabletas, lo que ha hecho surgir todo un ecosistema de software para este tipo dispositivos, así como nuevos modelos de negocios que van desde el uso del terminal móvil para la navegación tradicional hasta la prestación de los llamados Servicios Basados en la Localización (LBS).

Un buen ejemplo del uso del GPS en la telefonía móvil son las aplicaciones que permiten conocer la posición de amigos cercanos sobre un mapa base. Para ello basta con tener la aplicación respectiva para la plataforma deseada (Android, Bada, IOS, WP, Symbian) y permitir ser localizado por otros.

Aplicaciones.

El GPS es utilizado en múltiples campos como la geodesia, geofísica, geodinámica, astronomía, meteorología, topografía o cartografía. También se utiliza en la navegación marina, aérea o terrestre, en la sincronización del tiempo, para controlar flotas y maquinaría, en la localización automática de vehículos, la exploración y en los deportes de aventura.

Conexiones GSM. [8]

La segunda generación de la tecnología celular comenzó con una variedad amplia de estándares, los proveedores europeos sufrieron las consecuencias de una diversidad de normas incompatibles entre sí. El reconocimiento de este problema fue un factor que impulsó el desarrollo del estándar GSM para las comunicaciones móviles.

En 1982, cuando aparecieron los primeros servicios celulares comerciales, la CEPT (Conference Europenne de Postal et Telecommunications) tomó la iniciativa de poner en marcha un grupo de trabajo, llamado Groupe Special Mobile (GSM).

“El resultado del trabajo desarrollado por este grupo es el sistema GSM (sistema global para comunicaciones móviles), que comprende la estandarización de servicios, de las interfaces funcionales entre subsistemas y de la arquitectura de protocolos empleado, basado en el uso de estándares mundiales (en 1989 la responsabilidad de la estandarización pasó de manos del CEPT al Instituto Europeo de Estándares en Telecomunicaciones, ETSI).

A través de la creación de un sistema paneuropeo, el ETSI se planteó alcanzar los siguientes objetivos:

a) proveer mejor calidad de servicio que la que los sistemas analógicos otorgaban.

b) proveer servicio telefónico en toda Europa (crear roaning).

c) proveer transmisión de datos originados por fax, correos electrónicos, transferencia de archivos, etcétera.

También se consideró la necesidad de crear un sistema con bajos costos de implementación y de gran potencial para incrementar la eficiencia espectral, con una mejor calidad subjetiva de voz y la factibilidad de usar los teléfonos celulares con la tecnología vigente." (DAVID MUÑOZ RODRIGUEZ, 2002: 227).

Después de pruebas de campo y de la selección del método de acceso Time Division Multiple Access (TDMA), los gobiernos de 18 países firmaron en 1988 un acuerdo de intenciones. En este documento los firmantes se comprometían a cumplir las especificaciones, a adoptar este estándar único y a poner en marcha un servicio comercial GSM, que ofrece seguimiento automático de los teléfonos móviles en su desplazamiento por todos los países. Conforme se desarrolló, GSM mantuvo el acrónimo, aunque en la actualidad signifique Global System for Mobile communications.

"En un futuro se espera que exista roaning en cualquier país en el que exista una red GSM, de manera que el usuario mantenga siempre la misma identidad. A la vez, se espera que exista roaning entre los usuarios de GSM de la banda de los 900MHz y DCS 1800 (la versión GSM en los 1800MHz)." (DAVID MUÑOZ RODRIGUEZ, 2002: 228).

Arquitectura de la red GSM [9]

Las redes de telefonía móvil utilizan zonas circulares, denominadas celdas, para cubrir zonas geográficas. Cada celda está rodeada por 6 celdas continuas, motivo por el cual se utiliza un hexágono para su gráfica. Para evitar interferencias, dos celdas contiguas no pueden utilizar la misma frecuencia. Cada celda tiene una estación base denominada BTS (Estación Base Transceptora).

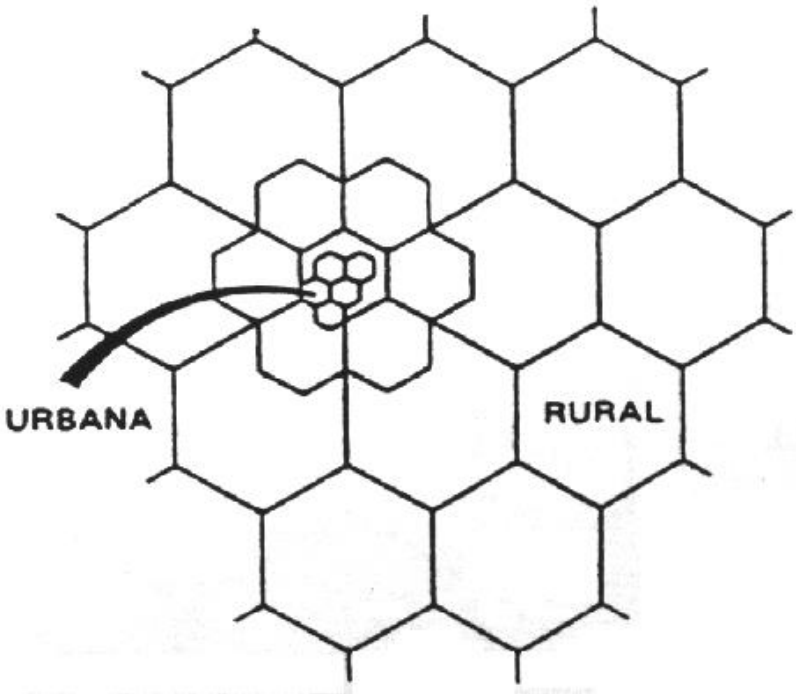


Figura 2.8. Esquema de celdas (Dirección General de Protección Civil y Emergencia, Mrio. del Interior, Gobierno de España. Publicado en www.proteccioncivil.org, extraído el 01/05/2014)

Cuanto menor es el radio de cobertura de una celda, mayor es el ancho de banda disponible. Por tal razón, en zonas urbanas muy pobladas hay celdas con radios de unos cientos de metros, mientras que en zonas rurales pueden alcanzar hasta 30km.

Tipos de células:

Macrocélulas rurales: con cobertura omnidireccional en un radio que puede llegar hasta los 30 km en función de la orografía. Utilizan tres antenas omnidireccionales: la central transmite y las dos de los laterales reciben.

Figura 2.9. Esquema de celdas (Dirección General de Protección Civil y Emergencia, Mrio. del Interior, Gobierno de España. Publicado en www.proteccioncivil.org, extraído el 01/05/2014)

Microcélulas urbanas: la cobertura en la célula se divide en tres sectores, lo cual permite la utilización de un mayor número de radiocanales y por tanto eleva el número de usuarios. El radio de cobertura de cada célula es mucho menor, del orden de 4 a 5 km. En cada sector se utilizan tres antenas cuya funcionalidad es la misma que en el caso de las macrocélulas.

Figura 2.10. Esquema de celdas (Dirección General de Protección Civil y Emergencia, Mrio. del Interior, Gobierno de España. Publicado en www.proteccioncivil.org, extraído el 01/05/2014)

Picocélulas: Se instalan en ubicaciones en las que se requiere un número adicional de canales o bien en puntos donde la cobertura de las microcélulas se hace difícil, como el caso de interiores (aeropuertos, estaciones de tren, etc.). El alcance es muy limitado, del orden de las centenas de metros.

Figura 2.11. Esquema de celdas (Dirección General de Protección Civil y Emergencia, Mrio. del Interior, Gobierno de España. Publicado en www.proteccioncivil.org, extraído el 01/05/2014)

En una red GSM la terminal del usuario se llama **estación móvil**. Las estaciones móviles se identifican por un número único de 15 dígitos denominado **IMEI** (Identificador Internacional de Equipos Móviles) y requieren de manera obligatoria una tarjeta SIM para conectarse a la red GSM.

Una tarjeta **SIM** (Módulo de Identificación del Abonado) es una tarjeta inteligente desmontable que contiene información de suscripción del usuario, parámetros de red y directorio telefónico. Esto permite al usuario mantener su información después de cambiar su teléfono. También puede cambiar de operador de telefonía, manteniendo el mismo equipo simplemente cambiando la tarjeta SIM. Algunos operadores introducen un bloqueo para que el teléfono utilice un sólo tipo de tarjeta SIM, o sólo una tarjeta SIM emitida por la compañía donde se compró el teléfono, práctica conocida como bloqueo de SIM. Esto se hace porque los equipos están subvencionados con el abono del servicio.

Cada tarjeta SIM posee un número de identificación único y secreto denominado **IMSI** (Identificador Internacional de Abonados Móviles). Este código se puede proteger con una clave de 4 dígitos llamada código **PIN** (Número de Identificación Personal) que es el que solicita la terminal cuando se enciende. El **PUK** (Clave Personal de Desbloqueo) es un código de 8 dígitos que se utiliza para desbloquear la tarjeta SIM en caso de que el PIN sea ingresado erróneamente 3 veces consecutivas. Después de 10 intentos erróneos del PUK, la tarjeta SIM se bloquea de forma permanente, siendo necesario que el operador la sustituya por una nueva.

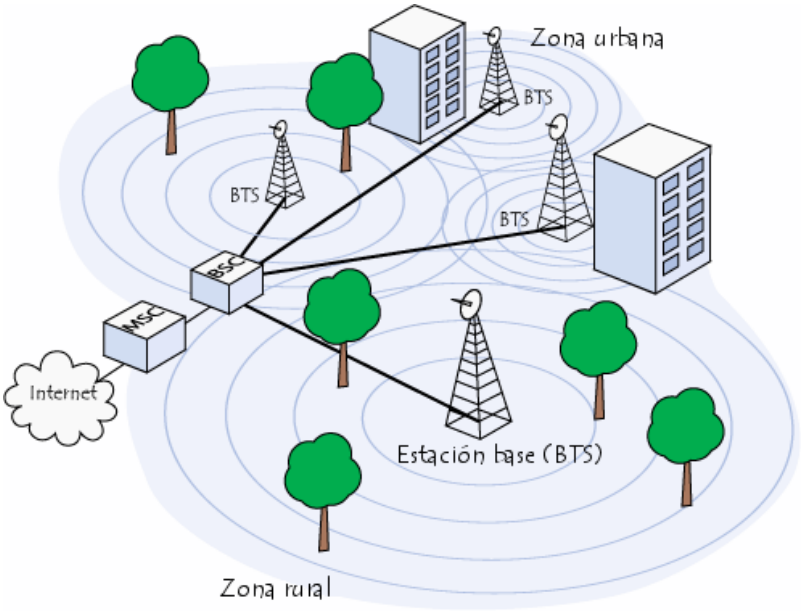


Figura 2.12. Publicación Estándar GSM (Sistema global de comunicaciones móviles) en http://es.kioskea.net/

La tarjeta SIM permite identificar a cada usuario independientemente de la terminal utilizada durante la comunicación con la estación base. Las comunicaciones entre una estación móvil y una estación base BTS (Estación Base Transceptora) se producen a través de un vínculo de radio.

Todas las BTS de una red celular están conectadas a un controlador **BSC** (Controlador de Estaciones Base) que administra la distribución de los recursos. El sistema compuesto por las estaciones bases + BSC se denomina **BSS** (Subsistema de Estaciones Base).

Por último, los controladores de estaciones base están físicamente conectados al **MSC** (Centro de Conmutación Móvil) que los conecta con la red de telefonía pública y con Internet, lo administra el operador de la red telefónica.

TRAMA DE DATOS TCP - UDP.

Escalando hacia la capa de aplicación del modelo de referencia OSI (Interconexión de Sistemas Abiertos) de la ISO (Organización Internacional de Normas), encontramos la capa de Red y Transporte. Los sistemas telemétricos actuales utilizan en la capa de red el protocolo IP debido a la conjunción de variables como fiabilidad, existencia de un contexto de servicios de red y servidores que pueden ser adquiridos a muy bajo costo. Respecto a la capa de transporte hay sistemas que utilizan UDP y otros que utilizan TCP como en el caso de los ejemplos que se exponen a continuación.

La ventaja que tiene UDP es el bajo costo de los mensajes, debido a que tienen poca información de control, no hay acuse de recibos y permiten el envío sin establecer previamente una conexión. Los sistemas telemétricos que utilizan este protocolo implementan soluciones de acuse de recibos y retransmisiones para asegurarse que los datagramas enviados lleguen a destino.

TCP garantiza que los datos serán entregados en su destino sin errores en el mismo orden en que se transmitieron y permite comunicarse con varias aplicaciones de un mismo host distinguiéndolas con el concepto de puerto. Las ventajas mencionadas hacen que los datagramas tengan un encabezado con mucha información y esto repercute en el coste del servicio. Actualmente es muy utilizado para sistemas telemétricos debido a que el costo por transferencia de datos de las compañías telefónicas es bajo, es un protocolo seguro y simplifica la codificación evitando controles y acuses de recibos.

A continuación se amplía información de estos protocolos de la capa de trasporte.

Modelo de referencia OSI. [10]

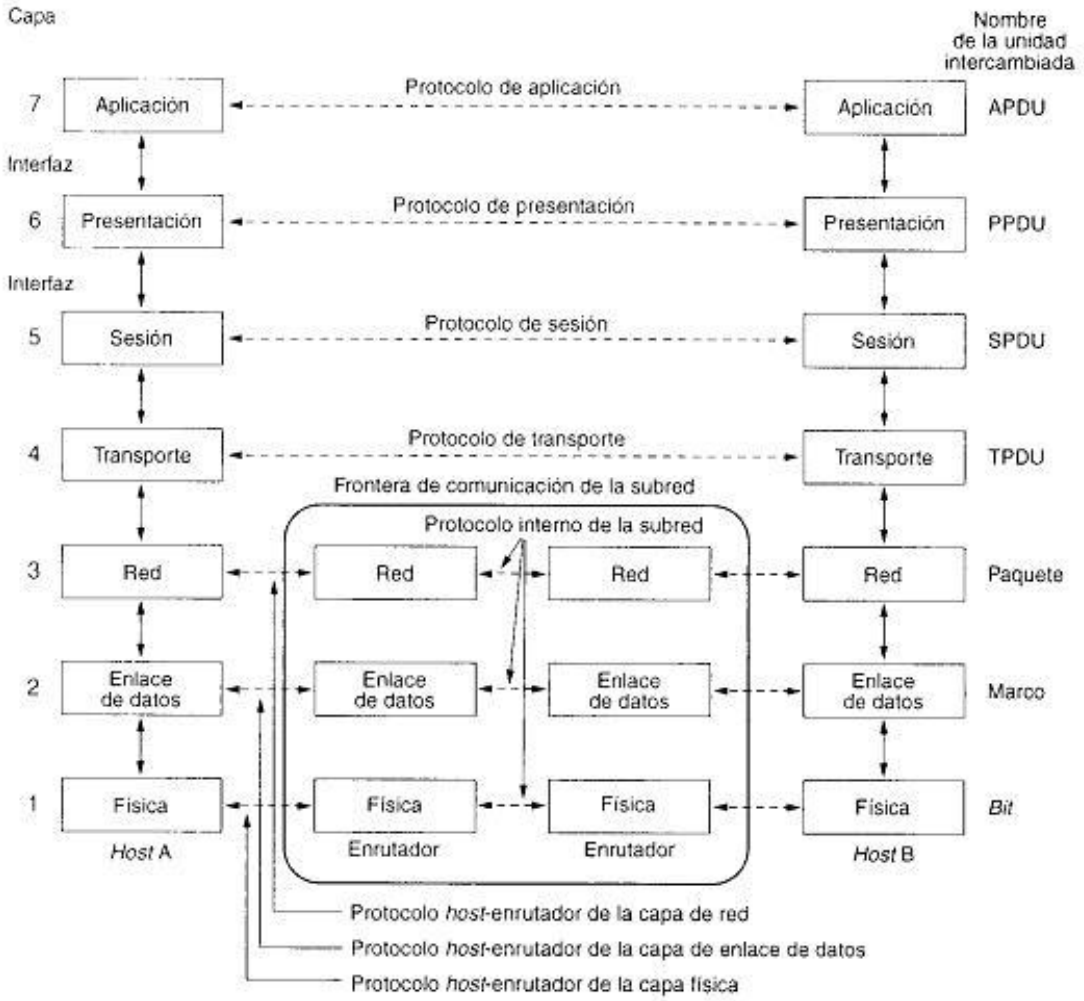
Fue desarrollado en 1984 por la Organización Internacional de Estándares (ISO), una federación global de organizaciones que representa aproximadamente a 130 países. El núcleo de este estándar es el modelo de referencia OSI, una normativa formada por siete capas que define las diferentes fases por las que deben pasar los datos para viajar de un dispositivo a otro sobre una red de comunicaciones.

Siguiendo el esquema de este modelo se crearon numerosos protocolos. El advenimiento de protocolos más flexibles donde las capas no están tan demarcadas y la correspondencia con los niveles no era tan clara, puso a este esquema en un segundo plano. Sin embargo es muy usado en la enseñanza como una manera de mostrar cómo puede estructurarse una "pila" de protocolos de comunicaciones.

El modelo especifica el protocolo que debe ser usado en cada capa, y suele hablarse de modelo de referencia, ya que es usado como una gran herramienta para la enseñanza de comunicación de redes.

Se trata de una normativa estandarizada útil debido a la existencia de muchas tecnologías, fabricantes y compañías dentro del mundo de las comunicaciones, y al estar en continua expansión, se tuvo que crear un método para que todos pudieran entenderse de algún modo, incluso cuando las tecnologías no coincidieran. De esta manera, no importa la localización geográfica o el lenguaje utilizado, todo el mundo debe atenerse a unas normas mínimas para poder comunicarse entre sí. Esto reviste gran importancia, principalmente, cuando hablamos de la red de redes, es decir, Internet.

Este modelo está dividido en siete capas:

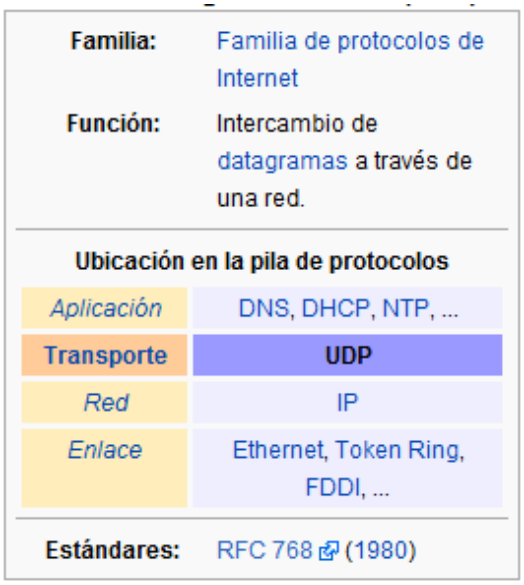


(TANENBAUM, 1997:29)

Capa de transporte.

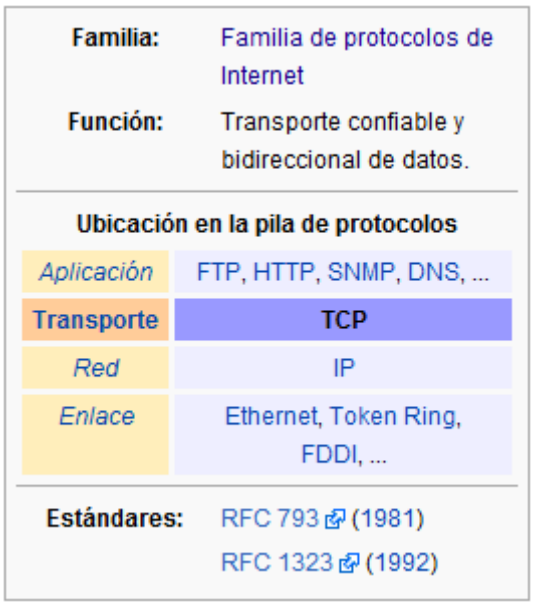
Capa encargada de efectuar el transporte de los datos (que se encuentran dentro del paquete) de la máquina origen a la de destino, independizándolo del tipo de red física que se esté utilizando. La PDU de la capa 4 se llama Segmento o Datagrama, dependiendo de si corresponde a TCP ó UDP. Sus protocolos son TCP y UDP; el primero orientado a conexión y el otro sin conexión. Trabajan, por lo tanto, con puertos lógicos y junto con la capa red dan forma a los conocidos como Sockets IP:Puerto (191.16.200.54:80).

User Datagram Protocol (UDP).



UDP es un protocolo del nivel de transporte basado en el intercambio de datagramas, cada paquete se trata de forma independiente, conteniendo cada uno la dirección de destino. La red puede encaminar cada fragmento hacia el equipo receptor por rutas distintas. No se garantiza que los paquetes lleguen en el orden adecuado ni que todos lleguen al destino. Permite el envío de datagramas a través de la red sin establecer una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento en su cabecera. Tampoco tiene confirmación ni control de flujo, por lo que los paquetes pueden adelantarse unos a otros; y tampoco se sabe si ha llegado correctamente, ya que no hay confirmación de entrega o recepción. Su uso principal es para protocolos como DHCP, BOOTP, DNS y demás protocolos en los que el intercambio de paquetes de la conexión/desconexión son mayores, o no son rentables con respecto a la información transmitida, así como para la transmisión de audio y video en tiempo real, donde no es posible realizar retransmisiones por los estrictos requisitos de retardo que se tiene en estos casos. [11]

Transmission Control Protocol (TCP).



TCP es un protocolo del nivel de transporte basado en el intercambio de datagramas. El protocolo garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron. También proporciona un mecanismo para distinguir distintas aplicaciones dentro de una misma máquina, a través del concepto de puerto.

TCP da soporte a muchas de las aplicaciones más populares de Internet (navegadores, intercambio de ficheros, clientes FTP, etc.) y protocolos de aplicación HTTP, SMTP, SSH y FTP.

El protocolo TCP tiene un sistema de acuse de recibo que permite al cliente y al servidor garantizar la recepción mutua de datos. [12]

Puertos

UDP y TCP utilizan puertos para permitir la comunicación entre aplicaciones. El campo de puerto tiene una longitud de 16 bits, por lo que el rango de valores válidos va de 0 a 65.535. El puerto 0 está reservado, pero es un valor permitido como puerto origen si el proceso emisor no espera recibir mensajes como respuesta.

Los puertos 1 a 1023 se llaman puertos "bien conocidos" y en sistemas operativos tipo Unix enlazar con uno de estos puertos requiere acceso como super usuario.

Los puertos 1024 a 49.151 son puertos registrados.

Los puertos 49.152 a 65.535 son puertos efímeros y son utilizados como puertos temporales, sobre todo por los clientes al comunicarse con los servidores.

CAPÍTULO III: Emulador, Componentes Software y Modelo de Datos

Este capítulo desarrolla las componentes software intervinientes en un sistema telemétrico y enumera puntos considerables del diseño.

También en este capítulo se desarrolla un emulador que permite generar tramas TCP/IP útiles para probar el software concurrente que recepciona la información.

Software de Recepción.

Un diseño de aplicaciones recomendable consiste en tener dos o más aplicaciones independientes. Una encargada de recepcionar las tramas enviadas por los equipos remotos, la otra es una aplicación que, a demanda del usuario o mediante temporizadores, recupera la información persistida por la primera y pone a disposición del usuario.

En esta sección detallaremos las funcionalidades de la primera y visualizaremos las mismas en código java.

Funcionalidades principales.

- Recepcionar la información enviada por los equipos.
- Realizar el checkum de las tramas.
- Interpretar los datos.
- Persistirlos o derivarlos a un servicio web.

Es importante que esta aplicación esté siempre activa, caso contrario se produce pérdida de información porque los equipos en muchas de las aplicaciones de telemetría transmiten datos 7x24x365.

Disponibilidad de la recepción de datos.

Este es un punto fundamental en los sistemas telemétricos para evitar pérdida de información. Existen sistemas donde la falla de esta aplicación es inadmisible, como es el caso de sistemas relacionados con la medicina, aeronáutica, etc. para estos casos el emisor, receptor y medio son especiales y generalmente desarrollados por el mismo fabricante (parte de un mismo producto).

Existen otros sistemas telemétricos donde la pérdida de información es importante, pero la inexistencia de un dato no pone en riesgo la vida, algunos ejemplos son monitoreo satelital de vehículos, telemetría de cisternas, digestores, plantas de silos, etc. De estos últimos mostraremos implementaciones y recomendaciones.

Para que dispositivos distantes pueden transmitir información desde sus equipos, un medio muy utilizado con el avance de las comunicaciones e internet es GPRS, donde los emisores transmiten tramas de datos especificando una dirección IP y puerto. La aplicación receptora generalmente escuchará con un socket en un puerto del servidor donde se ejecuta.

Mientras el servidor esté en funcionamiento la aplicación receptora tiene que estar activa. Existen dos mecanismos para que la aplicación sea ejecutada quedando en condiciones de comenzar a escuchar conexiones:

La aplicación receptora es una aplicación cliente y se configura el Sistema Operativo para ser ejecutada cuando el mismo inicia.

La aplicación es web y pertenece a un contenedor. Cuando éste inicia ejecuta la aplicación.

Cualquiera de las dos alternativas son buenas y aíslan al operador humano a monitorear continuamente la aplicación de posibles reinicios del sistema donde se está ejecutando.

El siguiente caso muestra un código de ejemplo del segundo punto donde un contenedor de servlet notifica a la aplicación que se está iniciando (método *contextInitialized*). Para esto se implementa la clase *ServletContextListener*, cuando el contenedor se detiene notifica a la aplicación y se ejecuta el método *contextDestroyed*.

```
1.  public class SocketListener implements ServletContextListener {
2.
3.      public void contextInitialized(ServletContextEvent arg0) {
4.
5.          //CONFIGURAMOS PARA UTILIZAR EL LOG4J
6.          ConstantesMS.REAL_PATH =
                                arg0.getServletContext().getRealPath("/");
7.          PropertyConfigurator.configure( ConstantesMS.REAL_PATH +
                                                    "log4j.properties");
8.
9.          //TOMAMOS EL SERVER SOCKET Y LO CORREMOS
10.         Server.getInstance().iniciar(2000, 600000);
11.
12.     }
13.     public void contextDestroyed(ServletContextEvent arg0) {
14.         //TERMINA LA EJECUCIÓN DEL HILO
15.         Server.getInstance().terminar();
16.     }
17. }
```

Cuando se ejecuta el método *contextInitialized* se instancia un objeto Server con información del puerto en el cual los sockets deben escuchar y el tiempo en milisegundos en que los mismos deben esperar luego de la última trasmisión.

Gestión de conexiones.

La clase Server que es el proceso principal donde los distintos equipos solicitan conexiones:

```
1.  public class Server extends Thread {
2.      //SERVER SOCKET
3.      private ServerSocket serverSocket = null;
4.
```

```
5.      //PARA EL SINGLETON
6.      private static Server instance = null;
7.
8.      private int puerto;
9.      private int timeOut;
10.
11.
12.     //SINGLETON -----------------
13.     public static Server getInstance(){
14.         if (instance == null){
15.             instance = new Server();
16.         }
17.         return instance;
18.     }
19.
20.     private Server(){
21.     }
22.     //---------------------------
23.
24.     public void iniciar(int puerto, int timeOut){
25.         //SETEAMOS EL PUERTO Y ARRANCAMOS EL HILO
26.         this.puerto = puerto;
27.         this.timeOut = timeOut;
28.         this.start();
29.     }
30.
31.
32.     public void run() {
33.         Socket socket = null;
34.
35.         try {
36.
37.             //LOG.
38.             log.info("SERVER SOCKET ESPERANDO CREACIÓN");
39.             serverSocket = new ServerSocket(this.puerto);
40.             //LOG.
41.             log.info("SERVER SOCKET CREADO");
42.
43.             while (true) {
44.
45.                 // LOG.
46.                 log.info("SERVER SOCKET ESPERANDO CONEXION");
47.                 // ACEPTO UNA CONEXIÓN DE UN CLIENTE
48.                 socket = serverSocket.accept();
49.                 // LOG.
50.                 log.info("SERVER SOCKET ACEPTO CONEXION");
51.
52.                 // LOG.
```

```
53.                     log.info("CONFIGURAMOS EL TIME OUT DE LECTURA DEL
                                NUEVO SOCKET [" + this.timeOut + "]");
54.                     // PONEMOS TIME OUT DE LECTURA!!!!!!!
55.                     socket.setSoTimeout(this.timeOut);
56.
57.                     // LOG.
58.                     log.info("SE LANZA NUEVO HILO DE EJECUCIÓN PARA
                                                ATENDER EL SOCKET");
59.                     // ATIENDO LA SOLICITUD DEL CLIENTE
60.                     (new HiloReceptor(socket)).start();
61.
62.             } catch (Throwable t) {
63.                 //TRATAMIENTO DE ERROR
64.             } finally {
65.                 try {
66.                     serverSocket.close();
67.                 } catch (Throwable t) {
68.                     //TRATAMIENTO DE ERROR
69.                 }
70.             }
71.         }
72.
73.         public void terminar(){
74.             try{
75.                 // LOG.
76.                 log.info( "SERVER SOCKET COMENZANDO DESCONEXION" );
77.                 //CERRAMOS EL SERVER SOCKET
78.                 serverSocket.close();
79.                 // LOG.
80.                 log.info( "SERVER SOCKET DESCONECTADO" );
81.
82.
83.                 //PRODUCE UNA EXCEPCION EN EL RUN()
84.                 Server.getInstance().interrupt();
85.                     Server.instance = null;
86.
87.             }catch(Throwable t){
88.                 //TRATAMIENTO DE ERROR
89.             }
90.         }
91.     }
```

Server es un servidor de conexiones al puerto 2000. Por cada equipo que requiere una conexión crea un socket, le asigna un timeout e instancia un nuevo hilo de ejecución (clase *HiloReceptor*) que es quien gestionará la comunicación entre el socket y el equipo.

Este esquema de administración de sockets no requiere mantenimiento, ya que cuando un equipo requiere conexión se le asigna un nuevo socket, luego cuando deja de transmitir y transcurre el tiempo

de timeout asignado al socket, se autodestruye. De esta manera la cantidad de sockets activos es equivalente a la cantidad de equipos que están transmitiendo más el remanente que aún no se han autodestruido.

Procesando las tramas de datos.

El socket encargado de recepcionar los datos de un equipo queda esperando por la lectura del primer byte de la trama. Una vez que esto ocurre realiza 4 tareas:

Lee la trama completa.

Realiza en checksum para comprobar la integridad de los datos.

Interpreta los datos.

Almacena los datos o se los informa a otra aplicación.

El diseño de la trama de datos es una componente fundamental dentro del protocolo de comunicación y su estructura e interpretación es definida generalmente por el fabricante del equipo. En el *ANEXO 2: "Trama de datos GSM"* se muestra la especificación de una trama de datos para un sistema de monitoreo satelital.

El siguiente código muestra las componentes principales para el procesamiento de las tramas de datos (clase HiloReceptor):

```
92.  public class HiloReceptor extends Thread
93.
94.      private Socket socket;
95.
96.  public void run() {
97.
98.      Vector vectorByte = new Vector();
99.
100.     try{
101.         //OBTENEMOS LOS INPUT - UOTPUT STREAM DEL SOCKET
102.         InputStream entrada = this.socket.getInputStream();
103.
104.
105.           DataInputStream entradaDatos = new
106.                 DataInputStream(entrada);
107.
108.         while (true){
109.
110.             //BORRAMOS CONTENIDO DEL VECTOR
111.             vectorByte.clear();
112.
113.             //LOG.
114.             log.info( "SOCKET ESPERANDO LEER DATO DE VERSIÓN DE
                 PLAQUETA | IP:" +this.socket.getInetAddress()+" |
                 PORT:"+this.socket.getPort() );
115.
```

```
116.                //LEO EL CAMPO DE FUNCIÓN PARA SABER LA VERSIÓN DE LA
                    PLAQUETA
117.                b = new Integer(entradaDatos.readUnsignedByte());
118.
119.                //LOG.
120.                log.info( "SOCKET LEYÓ DATO DE VERSIÓN DE PLAQUETA
v."+b+"                 | IP:" +this.socket.getInetAddress()+" |
                        PORT:"+this.socket.getPort());
121.
122.                //VERSIÓN DE LA PLAQUETA
123.                vectorByte.add(b) ;
124.
125.                //VERSIONES DIFERENTES DE EQUIPOS
126.                switch (b.byteValue()){
127.                    case 1: cargar_trama_v1(vectorByte, entradaDatos);
128.                    break;
129.                    case 2: cargar_trama_v2(vectorByte, entradaDatos);
130.                    break;
131.                }
132.
133.        }catch (SocketTimeoutException ste) {
134.            log.info( "SOCKET FINALIZÓ LA CONEXION POR TIME OUT | IP:"
            +this.socket.getInetAddress()+" |
            PORT:"+this.socket.getPort());
135.            //TRATAMIENTO DE ERROR SOCKET TIMEOUT EXCEPCION
136.
137.        }catch (IOException ioe) {
138.            log.info( "SOCKET FINALIZÓ LA CONEXIÓN POR CIERRE DE CONEXIÓN
            | IP:" +this.socket.getInetAddress()+" |
            PORT:"+this.socket.getPort());
139.            //TRATAMIENTO DE ERROR POR SOCKET EXCEPTION
140.
141.        }catch (Throwable t) {
142.            //TRATAMIENTO DE ERROR
143.        }finally{
144.            //CIERRA EL SOCKET
145.        }
146.
147.
148.
149.  private void cargar_trama_v2(Vector vectorByte, DataInputStream
150.                          entradaDatos) throws Exception{
151.
152.        TramaDTO tramaDTO = null;
153.
154.        try{
155.            //NRO DE SERIE (dos bytes --> b2 y b3)
156.            vectorByte.add( new
157.                Integer(entradaDatos.readUnsignedByte()) );
```

```
158.            vectorByte.add( new
159.                Integer(entradaDatos.readUnsignedByte()) );
160.
161.            //CANTIDAD DE REGISTROS QUE CONTIENE LA TRAMA (un byte --> b4)
162.            cantReg = new Integer(entradaDatos.readUnsignedByte());
163.            vectorByte.add( cantReg );
164.
165.            //TRAMA DE DATOS (20 BYTES * la cantidad de reg. que indica la
                                  trama)
166.            totalBytesReg = cantReg * ConstantesMS.LONG_TRAMA;
167.            for (i = 0; i<totalBytesReg; i++){
168.
169.            }
170.
171.            //LEO EL CHECKSUM DE LA TRAMA (bm)
172.            vectorByte.add( new
173.                Integer(entradaDatos.readUnsignedByte()) );
174.
175.            //CHEQUEAMOS LA TRAMA
176.            if (this.checksumOk(vectorByte, 256)){
177.
178.                //PARSEO DEL VECTOR PARA OBTENER LA TRAMA
179.                tramaDTO = this.parsear_Trama_Datos_v2(vectorByte);
180.
181.                //INVOCA A WEB SERVICE
182.                this.invocarWS(tramaDTO);
183.
184.                //GUARDAMOS EN BASE DE DATOS
185.                FachadaDao. putRegistroDatosDTO_v2(tramaDTO);
186.            }
187.
188.        }catch (Throwable t) {
189.        //TRATAMIENTO DE ERROR
190.        }
```

Chequeo de la trama.

El método *checksumOk(vectorByte, 256)* comprueba la integridad de los datos enviados en la trama. Existen numerosos métodos de comprobación. En el ejemplo que se expone se utilizó el siguiente procedimiento:

Sumatoria en base decimal de todos los bytes de la trama.

Divide por 256.

El cociente se compara con el último dato de la trama.

Estructurar datos de la trama.

El Método *parsear_Trama_Datos_v2(vectorByte)* retorna un objeto de la clase TramaDTO que se corresponde con los datos estructurados enviados por el equipo. Estos datos pueden ser enviados a otra aplicación mediante, por ejemplo, la invocación de un web service o persistidos en una base de datos.

Tratamiento de excepciones.

El socket receptor de la trama de datos puede concluir la comunicación con el emisor por dos motivos. En primera instancia porque el emisor finalizó de forma normal la comunicación. En este caso, se levanta una excepción del tipo java.net.SocketException, al ser capturada esta excepción el receptor puede cerrar la conexión. El otro motivo se debe a que el emisor deja de transmitir información, en cuyo caso, se levanta una excepción del tipo java.net.SocketTimeoutException una vez transcurridos los milisegundos seteados por el SocketServer al momento de instanciar el Socket que atenderá la transmisión de un equipo.

En muchos de los diseños de sistemas de telemetría como digestores, cisternas, sistemas hidráulicos, monitoreo satelital de móviles, etc. una vez establecida una conexión, en circunstancias normales del medio de comunicación, no debiera desconectarse. Las excepciones capturadas por “time out” son frecuentemente utilizadas para evaluar el medio de comunicación.

El código expuesto tiene un diseño muy interesante respecto al mantenimiento autosuficiente de la aplicación. Pueden ocurrir dos problemas de mantenimiento en estos tipos de sistemas que tienen que dar servicio 7x24. El primero está relacionado a iniciar la aplicación ante reinicios inesperados del servidor (equipo o WebServer) y el segundo con la administración de los hilos que contienen cada uno de los sockets que dialogan con un equipo. El primer problema se ha solucionado utilizando un Listener que se ejecuta cuando se inicializa el Webserver, el cual inicializa un hilo con un SocketServer que es quien recibe los pedidos de nuevas conexiones. El segundo se soluciona seteando un tiempo de timeout a cada uno de los sockets que atienden a cada uno de los equipos, un socket ante un corte en la conexión se muere pasado el tiempo de timeout. Un reconexión del equipo hace que sea atendido por un nuevo socket.

Logeo de eventos.

Se utiliza la librería Log4j para logear los eventos más significativos en la administración de sockets. Se logea tanto la creación y finalización del ServerSocket, como la creación y finalización de los Sockets que establecen las comunicaciones con cada uno de los equipos. Además se registran las tramas crudas recibidas desde los equipos antes de ser procesadas, lo que permite tener una auditoría de datos que es muy útil cuando existen inconvenientes con los equipos y/o comunicaciones ya que permiten analizar la problemática aislando gran parte al software.

Emulador de Equipos.

Con la intención de no disponer físicamente de equipos (plaquetas, GPS, diferentes sensores, etc.) en las etapas de desarrollo y testing, se ha desarrollado una aplicación que permite emular una o varias instancias de equipos.

Disponer del emulador tiene los siguientes beneficios:

Permite hacer pruebas de stress simulando gran concurrencia de equipos transmitiendo simultáneamente (muchas veces no se dispone en stock de la cantidad de equipos que se desea probar).

Permite cambiar fácilmente parámetros evitando tener que conectar los equipos a un puerto serie e ir cambiando uno por uno.

Económico, ya que no se deben tener chips activos y enviando información para las pruebas.

Pantalla de inicio.

Consta de un formulario con los siguientes campos:

- IP: es la dirección IP dónde se encuentra el servidor que recepciona las tramas.
- Puerto: número de puerto donde el servidor escucha las tramas de datos.
- Delay: es el intervalo de tiempo en el envío de las tramas.
- Trama: trama de dato a enviar. Al ser un campo de texto libre permite cambiar fácilmente datos en las tramas y poder colocar tramas de diferentes versiones de equipos. Para una mayor claridad, la aplicación soporta separar los datos de la trama con el carácter "+", el mismo es opcional y al momento de ser recepcionado la aplicación lo descarta. El siguiente es un ejemplo de trama donde se ha utilizado el signo para separar los datos de cabecera (versión del equipos, identificación del móvil y cantidad de registros) y comprobación de los registros de datos. Ver *ANEXO 2: "Trama de datos GSM")*:

002+000127+001+01101001601104500003403301702106005500405100004006212800000+216

Primer segmento (002): versión de la plaqueta.

Segundo segmento (000127): identificación de la plaqueta que está enviando.

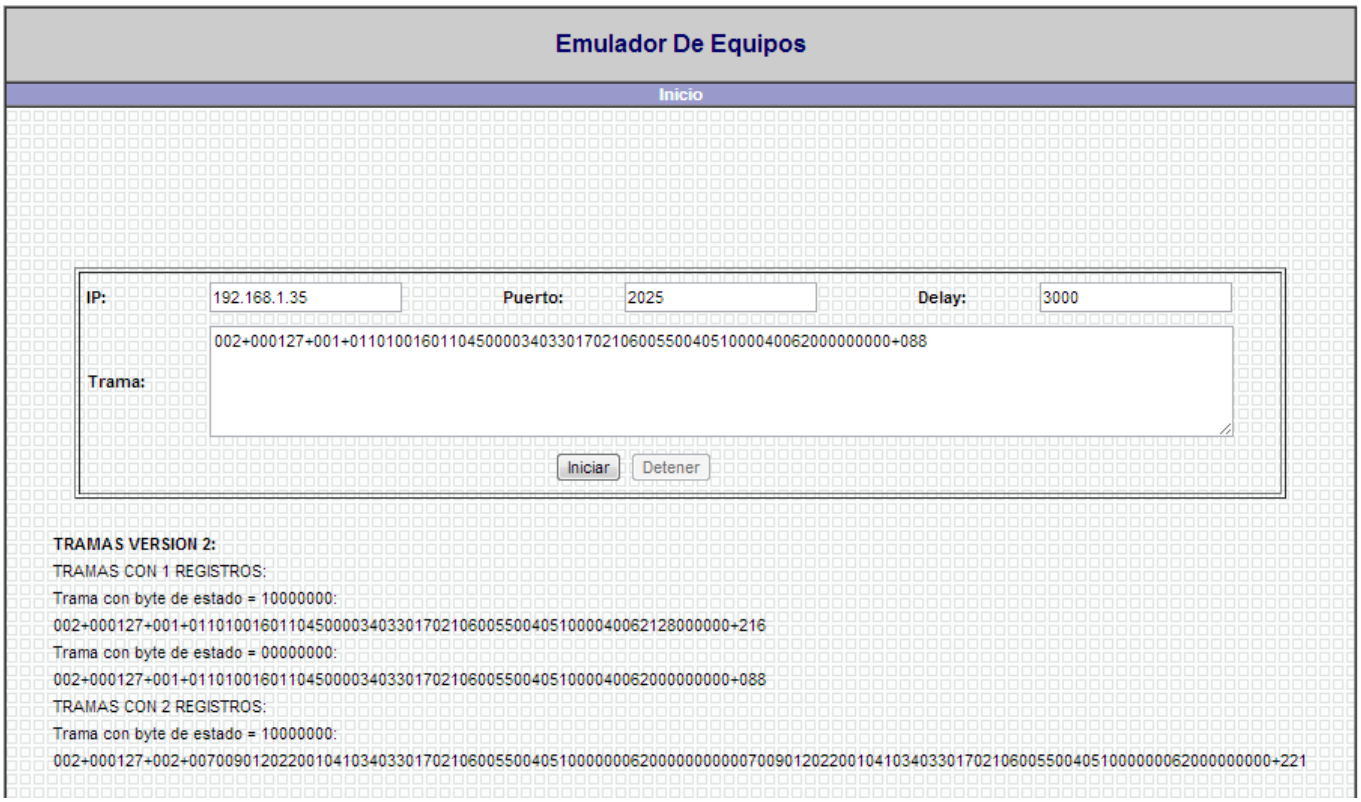
Tercer segmento (001): cantidad de registros de datos que tienen la trama.

Cuarto segmento (01101001601104500003403301702106005500405100004006212800000): registro de datos.

Quinto segmento (216): datos para el checksum.

El formulario posee un botón Iniciar / Iniciar Otro y un botón Detener. Cada vez que se presiona Iniciar se crea un nuevo hilo con un socket (equipo simulado) listo para que la aplicación que recepciona las tramas acepte la comunicación. Cuando se presionó el botón Iniciar el título del botón cambia a Iniciar Otro y se permiten crear más instancias con los mismos parámetros o diferentes (tramas, timeout, etc). El botón Detener hace que todos los equipos simulados dejen de transmitir datos. Si están transmitiendo una trama en el momento que se presiona el botón Detener, esa trama se termina de enviar.

Como ayuda a los desarrolladores y encargados de hacer pruebas, en la parte inferior, se visualizan tramas modelos que se pueden copiar / pegar en los campos del formulario. Hay tramas con el checksum correcto y con más de un registro de datos.



Envío de tramas de datos.

Al presionar el botón Iniciar / Iniciar Otro se ejecuta el siguiente Action (struts framework):

```
1.  public ActionForward execute(ActionMapping arg0, ActionForm form,
    HttpServletRequest arg2, HttpServletResponse arg3) throws Exception {
2.
3.      String ip;
4.      int puerto;
5.      int delay;
6.      String trama = "";
7.      String[] vTrama;
8.      String resp = null;
9.      EmuladorForm emuladorForm;
10.
11.     try{
12.         //CASTEAMOS EL FORMULARIO CON LOS DATOS INGRESADOS
13.         emuladorForm = (EmuladorForm) form;
14.
15.         //OBTENEMOS LOS DATOS INGRESADOS EN EL FORMULARIO QUITANDO LOS
              + PARA LA TRAMA
16.         ip = emuladorForm.getIp();
17.         puerto = emuladorForm.getPuerto();
18.         delay = emuladorForm.getDelay();
19.         vTrama = emuladorForm.getTrama().split("\\+");
20.         for (String elem : vTrama){
21.             trama += elem;
22.         }
23.
```

```
24.             //INSTANCIONAMOS UN NUEVO HILO (EQUIPO SIMULADO)
25.             Cliente.Iniciar();
26.             new Cliente(ip, puerto, delay, trama);
27.             arg2.setAttribute("iniciar","");
28.
29.             //COLOCAMOS MARCA PARA QUE CAMBIE EL TITULO DEL BOTON
                INICIAR POR INICIAR OTRO
30.             resp = "sucess";
31.
32.         }catch(Throwable t){
33.             new LogExcepcion(t);
34.             resp = "pageError";
35.         }
```

Recuperada la trama de datos se quitan, si existen, los caracteres + y se instancia la clase Cliente que es la encargada de enviar la trama de datos al servidor. Cada vez que se presiona el botón Iniciar Otro se crea una nueva instancia de la clase Cliente, al ser esta clase un hilo se crean tantos como equipos en simultáneo deseamos tener. Podemos decir que con el botón Iniciar / Iniciar Otro tenemos una fábrica de equipos emulados.

```
36.  public class Cliente extends Thread {
37.
38.         private String ip = null;
39.         private int puerto;
40.         private int delay;
41.         private String trama;
42.         private static int estado = 1; //1 EJECUTAR 0 TERMINAR
43.
44.
45.         public Cliente(String ip, int puerto, int delay, String trama){
46.             this.iniciar(ip, puerto, delay, trama);
47.         }
48.
49.         public void iniciar(String ip, int puerto, int delay, String trama){
50.             //SETEA LAS VARIABLES DE INSTANCIA CON LA IP, PUERTO, DELAY Y
                  TRAMA
51.             this.ip = ip;
52.             this.puerto = puerto;
53.             this.delay = delay;
54.             this.trama = trama;
55.
56.             //INICIALIZA EL HILO → EJECUTA MÉTODO RUN()
57.             this.start();
58.         }
```

La variable “estado” es utilizada como condición de corte de todos los hilos instanciados. Si se instancian varios objetos de la clase Cliente (equipo emulado), y se cambia el valor de la variable estado a cero, provocará que todos los hilos terminen, ya que estarán consultando el valor de dicha variable. Esto se logra definiendo la variable “estado” como variable de clases.

El método run() realiza las siguientes tareas:

Crea un socket y establece un canal de salida.

Instancia y carga con la trama un vector dinámico, para saber a priori la longitud consulta la cantidad de registros que tiene la trama.

Realiza iteraciones mientras la variable “estado” tenga asignado el valor 1 (uno), dentro del loop realiza dos acciones: envía por el socket la trama de datos y duerme el hilo una determinada cantidad de milisegundos. Dormir el hilo con el método sleep hace que no compita por recursos con otros procesos mientras está dormido.

Todos los hilos terminan cuando se presiona el botón Detener que asigna el valor 1 (uno) a la variable “estado”. Al ser una variable de clase y estar todas las instancias consultándola en la proposición del loop, hace que todos los hilos finalicen.

```
59.     public void run() {
60.     String trama;
61.     Socket socketCliente;
62.     int[] tramaByte;
63.     int i;
64.     String strByte;
65.     String strCantReg;
66.     int cantReg;
67.
68.     try{
69.         trama = this.trama;
70.
71.         socketCliente = new Socket(ip, puerto);
72.         OutputStream salida = socketCliente.getOutputStream();
73.
74.         //PROCESAMOS LA TRAMA
75.         //4 primeros corresponden al encabezado
76.         //20 de la trama
77.         //1 checksum
78.
79.         //Obtenemos la cantidad de registros
80.         strCantReg = trama.substring(9, 12);
81.         cantReg = (new Integer(strCantReg)).intValue();
82.
83.         //INSTANCIAMOS EL VECTOR PARA GUARDAR LA TRAMA DE DATOS
84.         tramaByte = new int[4+ConstantesED.LONG_TRAMA*cantReg+1];
85.
86.         //CARGAMOS LOS DATOS EN EL VECTOR DE TRAMA
87.         for (i = 0; i<4+ConstantesED.LONG_TRAMA*cantReg+1; i++){
88.             strByte = trama.substring(i*3, i*3+3);
89.             tramaByte[i] = (new Integer(strByte)).intValue();
90.         };
91.
```

```
92.             //MIENTRAS NO SE DESHABILITE EL HILO MANDA TEMPORALMENTE
                TRAMAS AL SERVIDOR
93.             while (Cliente.estado == 1){
94.                 for (i = 0; i<tramaByte.length; i++){
95.                     salida.write(tramaByte[i]);
96.                     salida.flush();
97.                 }
98.                 Cliente.sleep(delay);
99.             }
100.
101.            //CIERRA LA COMUNICACIÓN CON EL SOCKET DEL SERVIDOR
102.            socketCliente.close();
103.
104.        }catch(Throwable t){
105.                new LogExcepcion(t);
106.            }
107.        }
```

Esquema integrador de aplicaciones.

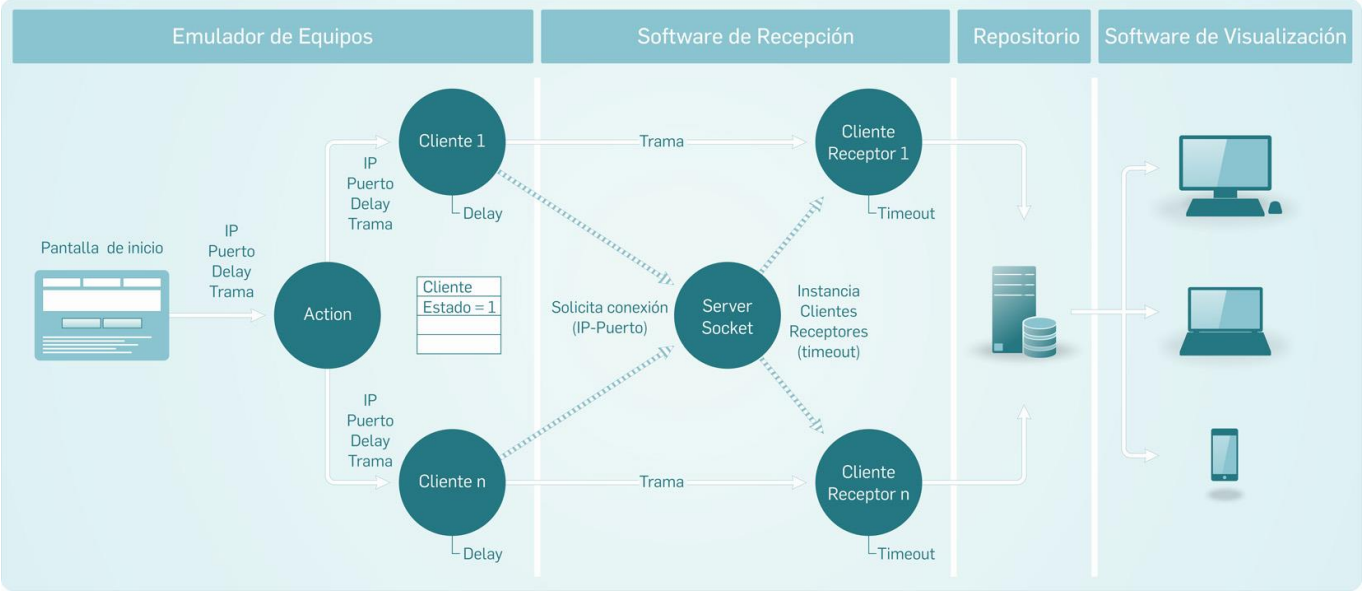


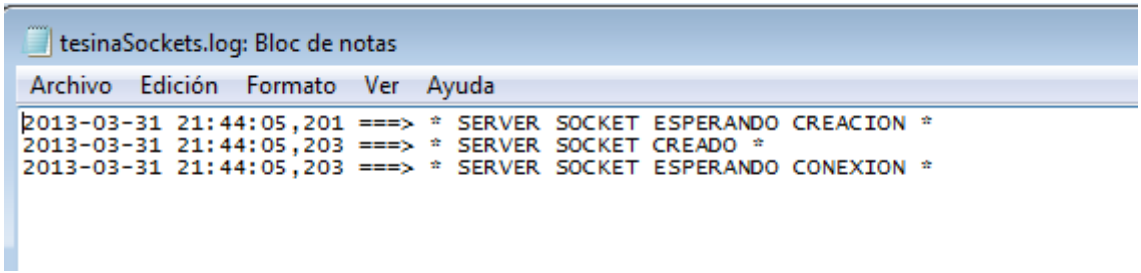
Figura 3.1. Flujo de información a través de esquema de aplicaciones.

Ejemplo 1: Recepción de tramas de un Equipo simulado.

El siguiente ejemplo muestra el caso más simple que se puede plantear. Habrá un Cliente (equipo emulado) enviando tramas cada 10 segundos, cada una de las cuales contendrá un registro de datos. El ServerSocket del software receptor guardará en un archivo de log los eventos más importantes e instanciará un Cliente Receptor ante la solicitud de conexión del Cliente. El Cliente Receptor interpreta la trama y la guarda en un archivo de log a modo de poder visualizar los resultados.

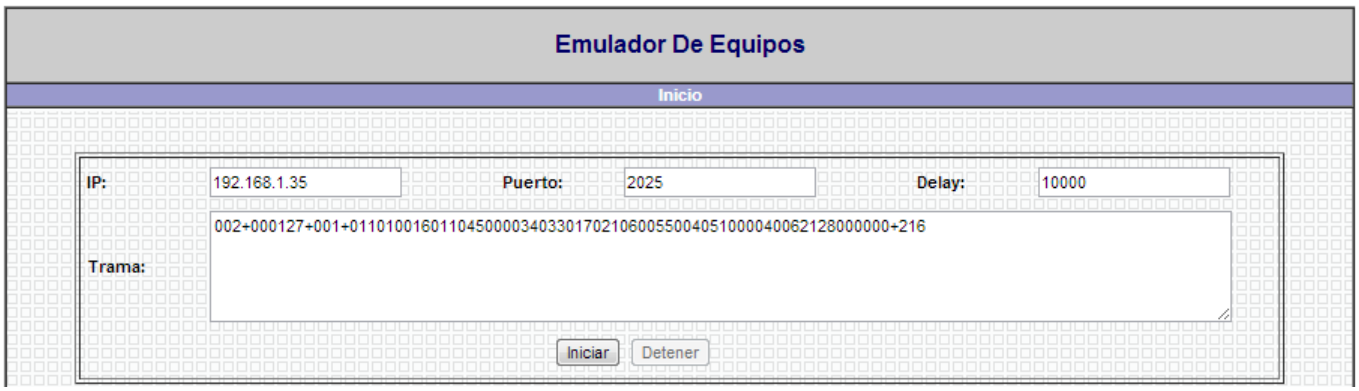
Implementación de las aplicaciones.

Cuando se despliega la aplicación o reinicia el servidor, el listener instancia la clase Server Socket que guarda la siguiente información en los archivos de log.



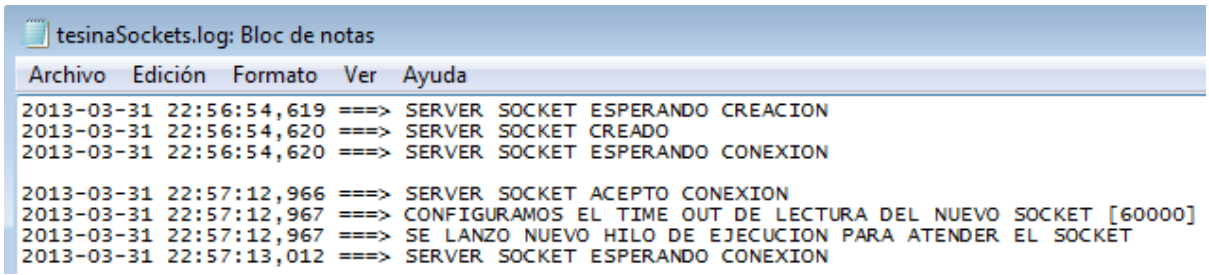
Pantalla de inicio de la aplicación Emulador de Equipos.

La identificación del equipo / plaqueta emulado es 127 correspondiente al tercer byte de la trama. En el *ANEXO 2: Trama de datos GSM* se visualiza la estructura de la trama completa.



Log del Server Socket luego de aceptar la conexión.

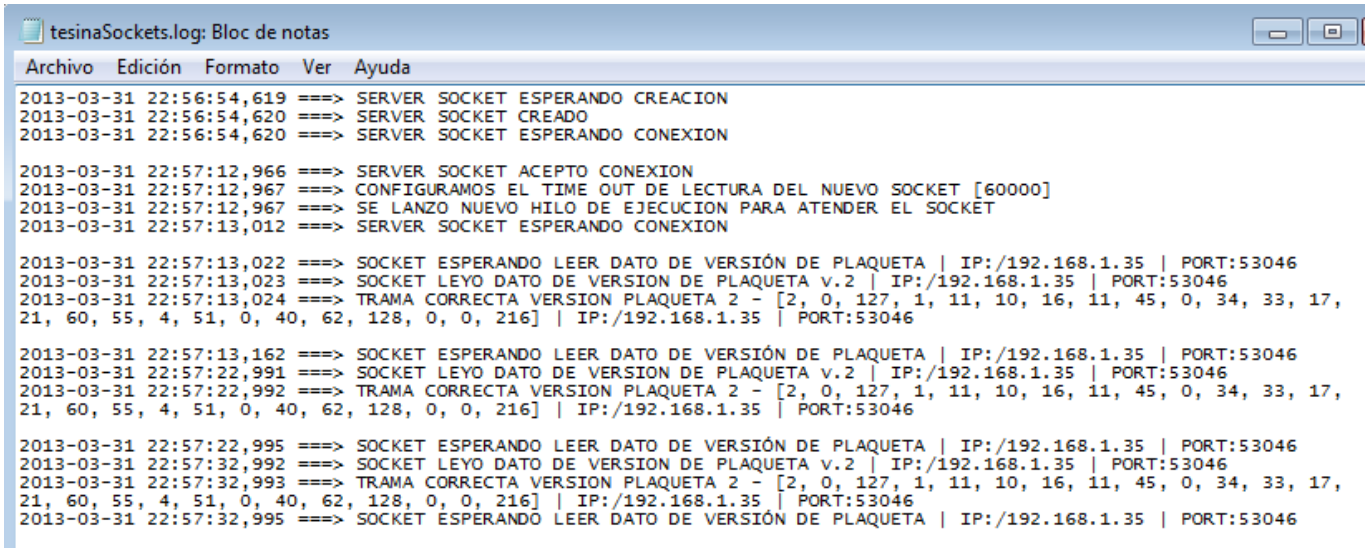
Al presionar el botón Inicio se instancia un cliente que le solicita al Server Socket una conexión para comenzar a enviarle información. El Server Socket la siguiente información en el archivo de log.



Log del Socket Servidor luego de recepcionar las tramas.

El Hilo Receptor instanciado para atender el Cliente guarda la siguiente información en el archivo de log. luego de recibir cada trama de datos (en el ejemplo se lleva recibido 3 tramas).

Se puede observar que el intervalo entre las tramas es de 10 segundos aproximadamente. El intervalo de tiempo que se asignó de delay al Cliente es el siguiente:



Log del Cliente Receptor al recibir las tramas.

La clase del DAO (Data Access Object) encargada de persistir la información, luego de ser chequeada e interpretada, la guarda en un archivo de log. Se utilizó un archivo de log. por la facilidad en la persistencia, pero en los sistemas de telemetría en esta instancia generalmente se persiste en una base de datos o invoca a un WebService.

tesinaTramas.log: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
2013-03-31 22:57:13,150 ===> Versión de la Plaqueta: 2
2013-03-31 22:57:13,151 ===> Identificador de Plaqueta: 127
2013-03-31 22:57:13,151 ===> Cant. Tramas:1
2013-03-31 22:57:13,151 ===> Trama 1
2013-03-31 22:57:13,161 ===>    Fecha: 16/10/2011
2013-03-31 22:57:13,161 ===>    Hora: 08:45
2013-03-31 22:57:13,162 ===>    Latitud: 34.55728833333333
2013-03-31 22:57:13,162 ===>    Longitud: 60.918458333333334
2013-03-31 22:57:13,162 ===>    Velocidad: 4.0
2013-03-31 22:57:13,162 ===>    Direccion: 124
2013-03-31 22:57:13,162 ===>    Bits de estados: 128
2013-03-31 22:57:13,162 ===> ---------------------------------------
2013-03-31 22:57:22,993 ===> Versión de la Plaqueta: 2
2013-03-31 22:57:22,993 ===> Identificador de Plaqueta: 127
2013-03-31 22:57:22,993 ===> Cant. Tramas:1
2013-03-31 22:57:22,993 ===> Trama 1
2013-03-31 22:57:22,993 ===>    Fecha: 16/10/2011
2013-03-31 22:57:22,994 ===>    Hora: 08:45
2013-03-31 22:57:22,994 ===>    Latitud: 34.55728833333333
2013-03-31 22:57:22,994 ===>    Longitud: 60.918458333333334
2013-03-31 22:57:22,994 ===>    Velocidad: 4.0
2013-03-31 22:57:22,994 ===>    Direccion: 124
2013-03-31 22:57:22,994 ===>    Bits de estados: 128
2013-03-31 22:57:22,995 ===> ---------------------------------------
2013-03-31 22:57:32,993 ===> Versión de la Plaqueta: 2
2013-03-31 22:57:32,993 ===> Identificador de Plaqueta: 127
2013-03-31 22:57:32,993 ===> Cant. Tramas:1
2013-03-31 22:57:32,994 ===> Trama 1
2013-03-31 22:57:32,994 ===>    Fecha: 16/10/2011
2013-03-31 22:57:32,994 ===>    Hora: 08:45
2013-03-31 22:57:32,995 ===>    Latitud: 34.55728833333333
2013-03-31 22:57:32,995 ===>    Longitud: 60.918458333333334
2013-03-31 22:57:32,995 ===>    Velocidad: 4.0
2013-03-31 22:57:32,995 ===>    Direccion: 124
2013-03-31 22:57:32,995 ===>    Bits de estados: 128
2013-03-31 22:57:32,995 ===> ---------------------------------------
```

Log del Socket Servidor luego de finalizar la conexión por el Cliente.

Al presionar el botón Detener el Cliente cierra la conexión del socket emisor y el Hilo Receptor, al detectar que se finalizó normalmente la conexión, cierra su socket y el hilo finaliza.

tesinaSockets.log: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
2013-04-01 00:10:53,457 ===> SOCKET ESPERANDO LEER DATO DE VERSIÓN DE PLAQUETA | IP:/192.168.1.35 | PORT:54227
2013-04-01 00:11:03,456 ===> SOCKET FINALIZÓ LA CONEXIÓN POR CIERRE DE CONEXIÓN | IP:/192.168.1.35 | PORT:54227
```

Ejemplo 2: Recepción de tramas en simultáneo de tres Equipos simulados.

El siguiente ejemplo muestra tres Clientes (equipos emulados) enviando tramas cada 3, 6 y 9 segundos.

Implementación de las aplicaciones.

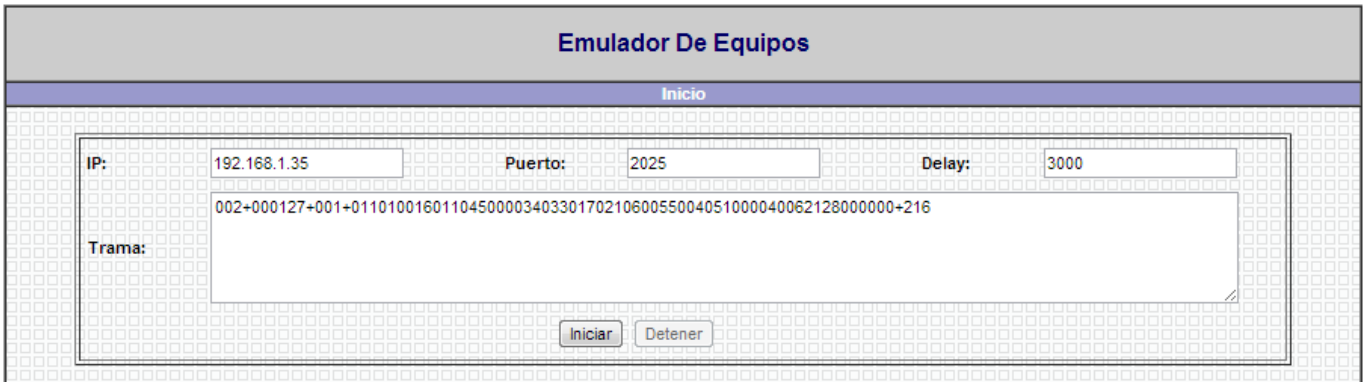
tesinaSockets.log: Bloc de notas

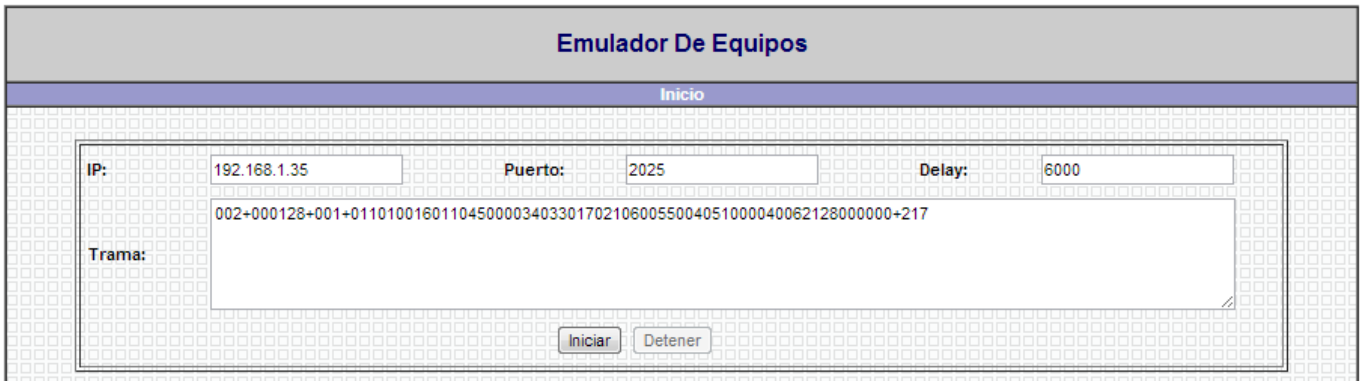
Archivo Edición Formato Ver Ayuda

```
2013-03-31 21:44:05,201 ===> * SERVER SOCKET ESPERANDO CREACION *
2013-03-31 21:44:05,203 ===> * SERVER SOCKET CREADO *
2013-03-31 21:44:05,203 ===> * SERVER SOCKET ESPERANDO CONEXION *
```

Pantalla de inicio de la aplicación Emulador de Equipos.

La identificaciones de los equipos / plaquetas emulados son 127 (frecuencia de envío de 3 segundos) y 128 (frecuencia de envío de 6 segundos) correspondiente al tercer byte de la trama. En el *ANEXO 2: "Trama de datos GSM"* se visualiza la estructura de la trama completa.



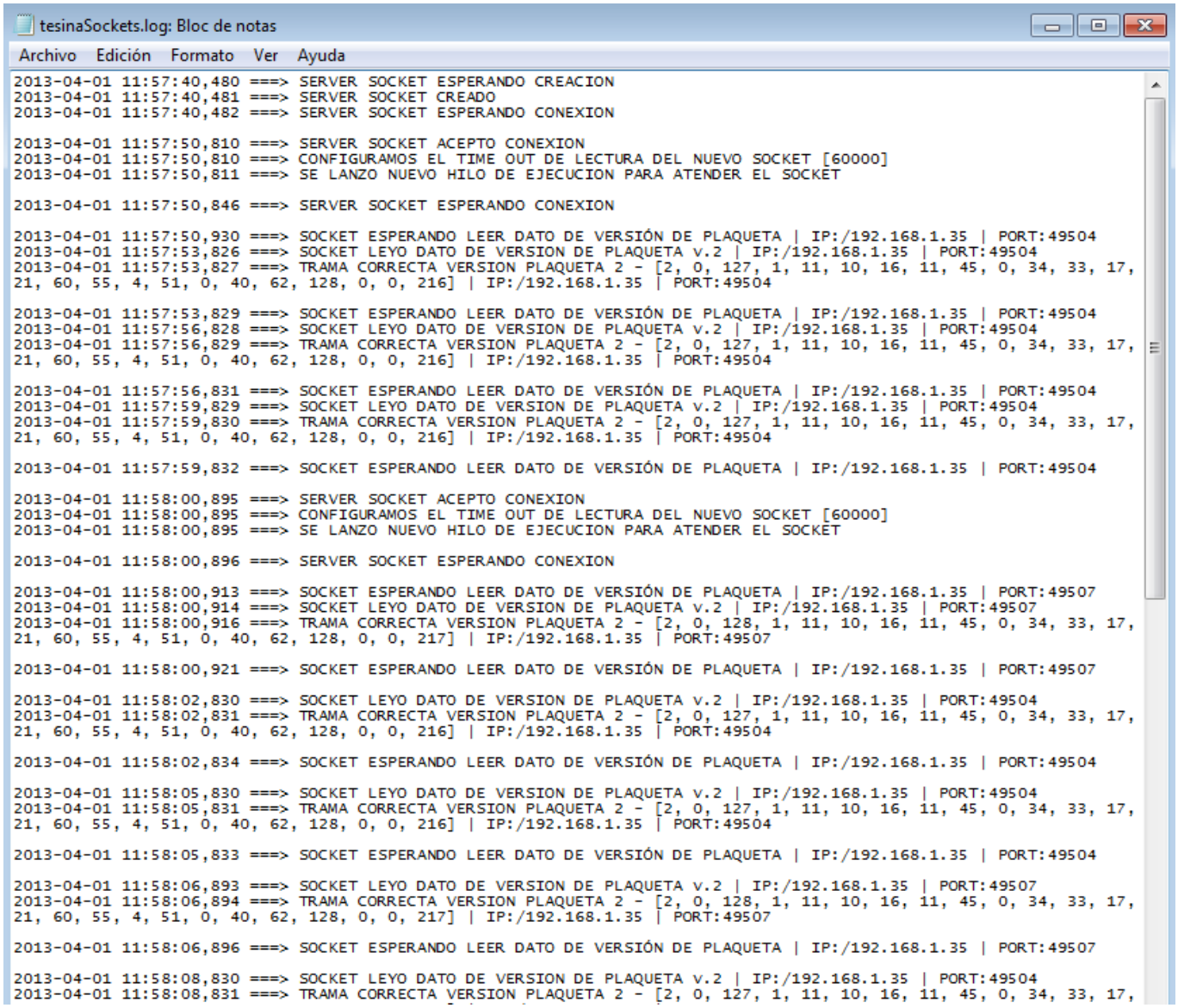


Log del Server Socket luego de aceptar las conexiones y recepcionar las tramas.

El Server Socket acepta la solicitud del Cliente cuya identificación de plaqueta emulada es la 127, le asigna un Hilo Receptor en el puerto 49504 con un tiempo de timeout de 1 minuto y queda esperando nuevas solicitudes de conexión.

El Hilo Receptor de la plaqueta 127 registra sus logs al recepcionar las tramas, y luego del tercer log se observa que el Server Socket recibe la solicitud del Cliente cuya identificación de plaqueta emulada es la 128, le asigna un Hilo Receptor en el puerto 49507 con un tiempo de timeout de 1 minuto y queda esperando nuevas solicitudes de conexión.

A partir de la instanciación de los Hilos Receptores para atender concurrentemente a los Clientes (equipos / plaquetas emuladas 127 y 128) se registran dos logs de la plaqueta emulada 127 (delay asignado de 3 segundos) cada un log de la plaqueta emulada 128 (delay asignado de 6 segundos)



tesinaSockets.log: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
2013-04-01 11:57:40,480 ===> SERVER SOCKET ESPERANDO CREACION
2013-04-01 11:57:40,481 ===> SERVER SOCKET CREADO
2013-04-01 11:57:40,482 ===> SERVER SOCKET ESPERANDO CONEXION

2013-04-01 11:57:50,810 ===> SERVER SOCKET ACEPTO CONEXION
2013-04-01 11:57:50,810 ===> CONFIGURAMOS EL TIME OUT DE LECTURA DEL NUEVO SOCKET [60000]
2013-04-01 11:57:50,811 ===> SE LANZO NUEVO HILO DE EJECUCION PARA ATENDER EL SOCKET

2013-04-01 11:57:50,846 ===> SERVER SOCKET ESPERANDO CONEXION

2013-04-01 11:57:50,930 ===> SOCKET ESPERANDO LEER DATO DE VERSIÓN DE PLAQUETA | IP:/192.168.1.35 | PORT:49504
2013-04-01 11:57:53,826 ===> SOCKET LEYO DATO DE VERSION DE PLAQUETA v.2 | IP:/192.168.1.35 | PORT:49504
2013-04-01 11:57:53,827 ===> TRAMA CORRECTA VERSION PLAQUETA 2 - [2, 0, 127, 1, 11, 10, 16, 11, 45, 0, 34, 33, 17,
21, 60, 55, 4, 51, 0, 40, 62, 128, 0, 0, 216] | IP:/192.168.1.35 | PORT:49504

2013-04-01 11:57:53,829 ===> SOCKET ESPERANDO LEER DATO DE VERSIÓN DE PLAQUETA | IP:/192.168.1.35 | PORT:49504
2013-04-01 11:57:56,828 ===> SOCKET LEYO DATO DE VERSION DE PLAQUETA v.2 | IP:/192.168.1.35 | PORT:49504
2013-04-01 11:57:56,829 ===> TRAMA CORRECTA VERSION PLAQUETA 2 - [2, 0, 127, 1, 11, 10, 16, 11, 45, 0, 34, 33, 17,
21, 60, 55, 4, 51, 0, 40, 62, 128, 0, 0, 216] | IP:/192.168.1.35 | PORT:49504

2013-04-01 11:57:56,831 ===> SOCKET ESPERANDO LEER DATO DE VERSIÓN DE PLAQUETA | IP:/192.168.1.35 | PORT:49504
2013-04-01 11:57:59,829 ===> SOCKET LEYO DATO DE VERSION DE PLAQUETA v.2 | IP:/192.168.1.35 | PORT:49504
2013-04-01 11:57:59,830 ===> TRAMA CORRECTA VERSION PLAQUETA 2 - [2, 0, 127, 1, 11, 10, 16, 11, 45, 0, 34, 33, 17,
21, 60, 55, 4, 51, 0, 40, 62, 128, 0, 0, 216] | IP:/192.168.1.35 | PORT:49504

2013-04-01 11:57:59,832 ===> SOCKET ESPERANDO LEER DATO DE VERSIÓN DE PLAQUETA | IP:/192.168.1.35 | PORT:49504

2013-04-01 11:58:00,895 ===> SERVER SOCKET ACEPTO CONEXION
2013-04-01 11:58:00,895 ===> CONFIGURAMOS EL TIME OUT DE LECTURA DEL NUEVO SOCKET [60000]
2013-04-01 11:58:00,895 ===> SE LANZO NUEVO HILO DE EJECUCION PARA ATENDER EL SOCKET

2013-04-01 11:58:00,896 ===> SERVER SOCKET ESPERANDO CONEXION

2013-04-01 11:58:00,913 ===> SOCKET ESPERANDO LEER DATO DE VERSIÓN DE PLAQUETA | IP:/192.168.1.35 | PORT:49507
2013-04-01 11:58:00,914 ===> SOCKET LEYO DATO DE VERSION DE PLAQUETA v.2 | IP:/192.168.1.35 | PORT:49507
2013-04-01 11:58:00,916 ===> TRAMA CORRECTA VERSION PLAQUETA 2 - [2, 0, 128, 1, 11, 10, 16, 11, 45, 0, 34, 33, 17,
21, 60, 55, 4, 51, 0, 40, 62, 128, 0, 0, 217] | IP:/192.168.1.35 | PORT:49507

2013-04-01 11:58:00,921 ===> SOCKET ESPERANDO LEER DATO DE VERSIÓN DE PLAQUETA | IP:/192.168.1.35 | PORT:49507

2013-04-01 11:58:02,830 ===> SOCKET LEYO DATO DE VERSION DE PLAQUETA v.2 | IP:/192.168.1.35 | PORT:49504
2013-04-01 11:58:02,831 ===> TRAMA CORRECTA VERSION PLAQUETA 2 - [2, 0, 127, 1, 11, 10, 16, 11, 45, 0, 34, 33, 17,
21, 60, 55, 4, 51, 0, 40, 62, 128, 0, 0, 216] | IP:/192.168.1.35 | PORT:49504

2013-04-01 11:58:02,834 ===> SOCKET ESPERANDO LEER DATO DE VERSIÓN DE PLAQUETA | IP:/192.168.1.35 | PORT:49504

2013-04-01 11:58:05,830 ===> SOCKET LEYO DATO DE VERSION DE PLAQUETA v.2 | IP:/192.168.1.35 | PORT:49504
2013-04-01 11:58:05,831 ===> TRAMA CORRECTA VERSION PLAQUETA 2 - [2, 0, 127, 1, 11, 10, 16, 11, 45, 0, 34, 33, 17,
21, 60, 55, 4, 51, 0, 40, 62, 128, 0, 0, 216] | IP:/192.168.1.35 | PORT:49504

2013-04-01 11:58:05,833 ===> SOCKET ESPERANDO LEER DATO DE VERSIÓN DE PLAQUETA | IP:/192.168.1.35 | PORT:49504

2013-04-01 11:58:06,893 ===> SOCKET LEYO DATO DE VERSION DE PLAQUETA v.2 | IP:/192.168.1.35 | PORT:49507
2013-04-01 11:58:06,894 ===> TRAMA CORRECTA VERSION PLAQUETA 2 - [2, 0, 128, 1, 11, 10, 16, 11, 45, 0, 34, 33, 17,
21, 60, 55, 4, 51, 0, 40, 62, 128, 0, 0, 217] | IP:/192.168.1.35 | PORT:49507

2013-04-01 11:58:06,896 ===> SOCKET ESPERANDO LEER DATO DE VERSIÓN DE PLAQUETA | IP:/192.168.1.35 | PORT:49507

2013-04-01 11:58:08,830 ===> SOCKET LEYO DATO DE VERSION DE PLAQUETA v.2 | IP:/192.168.1.35 | PORT:49504
2013-04-01 11:58:08,831 ===> TRAMA CORRECTA VERSION PLAQUETA 2 - [2, 0, 127, 1, 11, 10, 16, 11, 45, 0, 34, 33, 17,
```

Log del Cliente Receptor al recibir las tramas.

La misma situación a la comentada en el punto anterior se muestra en el DAO logeando las tramas interpretadas de ambos equipos emulados.

tesinaTramas.log: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
2013-04-01 11:57:53,827 ===> Versión de la Plaqueta: 2
2013-04-01 11:57:53,827 ===> Identificador de Plaqueta: 127
2013-04-01 11:57:53,827 ===> Cant. Tramas:1
2013-04-01 11:57:53,828 ===> Trama 1
2013-04-01 11:57:53,828 ===>    Fecha: 16/10/2011
2013-04-01 11:57:53,828 ===>    Hora: 08:45
2013-04-01 11:57:53,829 ===>    Latitud: 34.55728833333333
2013-04-01 11:57:53,829 ===>    Longitud: 60.918458333333334
2013-04-01 11:57:53,829 ===>    Velocidad: 4.0
2013-04-01 11:57:53,829 ===>    Direccion: 124
2013-04-01 11:57:53,829 ===>    Bits de estados: 128
2013-04-01 11:57:53,829 ===> ----------------------------------------
2013-04-01 11:57:56,829 ===> Versión de la Plaqueta: 2
2013-04-01 11:57:56,829 ===> Identificador de Plaqueta: 127
2013-04-01 11:57:56,829 ===> Cant. Tramas:1
2013-04-01 11:57:56,829 ===> Trama 1
2013-04-01 11:57:56,830 ===>    Fecha: 16/10/2011
2013-04-01 11:57:56,830 ===>    Hora: 08:45
2013-04-01 11:57:56,831 ===>    Latitud: 34.55728833333333
2013-04-01 11:57:56,831 ===>    Longitud: 60.918458333333334
2013-04-01 11:57:56,831 ===>    Velocidad: 4.0
2013-04-01 11:57:56,831 ===>    Direccion: 124
2013-04-01 11:57:56,831 ===>    Bits de estados: 128
2013-04-01 11:57:56,831 ===> ----------------------------------------
2013-04-01 11:57:59,830 ===> Versión de la Plaqueta: 2
2013-04-01 11:57:59,830 ===> Identificador de Plaqueta: 127
2013-04-01 11:57:59,830 ===> Cant. Tramas:1
2013-04-01 11:57:59,831 ===> Trama 1
2013-04-01 11:57:59,831 ===>    Fecha: 16/10/2011
2013-04-01 11:57:59,831 ===>    Hora: 08:45
2013-04-01 11:57:59,832 ===>    Latitud: 34.55728833333333
2013-04-01 11:57:59,832 ===>    Longitud: 60.918458333333334
2013-04-01 11:57:59,832 ===>    Velocidad: 4.0
2013-04-01 11:57:59,832 ===>    Direccion: 124
2013-04-01 11:57:59,832 ===>    Bits de estados: 128
2013-04-01 11:57:59,832 ===> ----------------------------------------
2013-04-01 11:58:00,917 ===> Versión de la Plaqueta: 2
2013-04-01 11:58:00,918 ===> Identificador de Plaqueta: 128
2013-04-01 11:58:00,919 ===> Cant. Tramas:1
2013-04-01 11:58:00,919 ===> Trama 1
2013-04-01 11:58:00,920 ===>    Fecha: 16/10/2011
2013-04-01 11:58:00,920 ===>    Hora: 08:45
2013-04-01 11:58:00,920 ===>    Latitud: 34.55728833333333
2013-04-01 11:58:00,920 ===>    Longitud: 60.918458333333334
2013-04-01 11:58:00,920 ===>    Velocidad: 4.0
2013-04-01 11:58:00,920 ===>    Direccion: 124
2013-04-01 11:58:00,921 ===>    Bits de estados: 128
2013-04-01 11:58:00,921 ===> ----------------------------------------
2013-04-01 11:58:02,831 ===> Versión de la Plaqueta: 2
2013-04-01 11:58:02,832 ===> Identificador de Plaqueta: 127
```

Log del Socket Servidor luego de finalizar la conexión por el Cliente.

tesinaSockets.log: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
2013-04-01 11:58:26,836 ===> SOCKET FINALIZÓ LA CONEXIÓN POR CIERRE DE CONEXIÓN | IP:/192.168.1.35 | PORT:49504
2013-04-01 11:58:30,898 ===> SOCKET FINALIZÓ LA CONEXIÓN POR CIERRE DE CONEXIÓN | IP:/192.168.1.35 | PORT:49507
```

Ejemplo 3: Mantenimiento autónomo de Hilos Receptores.

El objetivo de este ejemplo es simular un caso real de corte en la comunicación y reconexión. Se muestra cómo se autodestruyen los Hilos Receptores cuando no reciben tramas de datos en un tiempo mayor al asignado a su timeout y ante una nueva solicitud de conexión se le asigna otro Hilo Receptor. En sistemas de telemetría productivos, las causales por las que pueden ocurrir estas desconexiones sin cierre normal de sockets pueden darse por muchas circunstancias, entre las más comunes se encuentran pérdida en la señal de la red GSM o, en el caso de equipos móviles, el no envío de

información por determinación del equipo al no haber obtenido las coordenadas correctamente del GPS (por estar bajo techo, puente o entre edificios de gran altura).

Para simular esta situación se ha realizado una pequeña modificación a la clase Cliente, de tal modo que al presionar el botón Detener no cierre los sockets, sino que no envíe más tramas manteniendo establecida la conexión. Para implementar lo anterior, dormimos el hilo del Cliente el mismo valor de tiempo que le asigna el Server Socket a los Hilos Receptores.

Código de la clase Cliente modificado para no enviar tramas al presionar el botón detener.

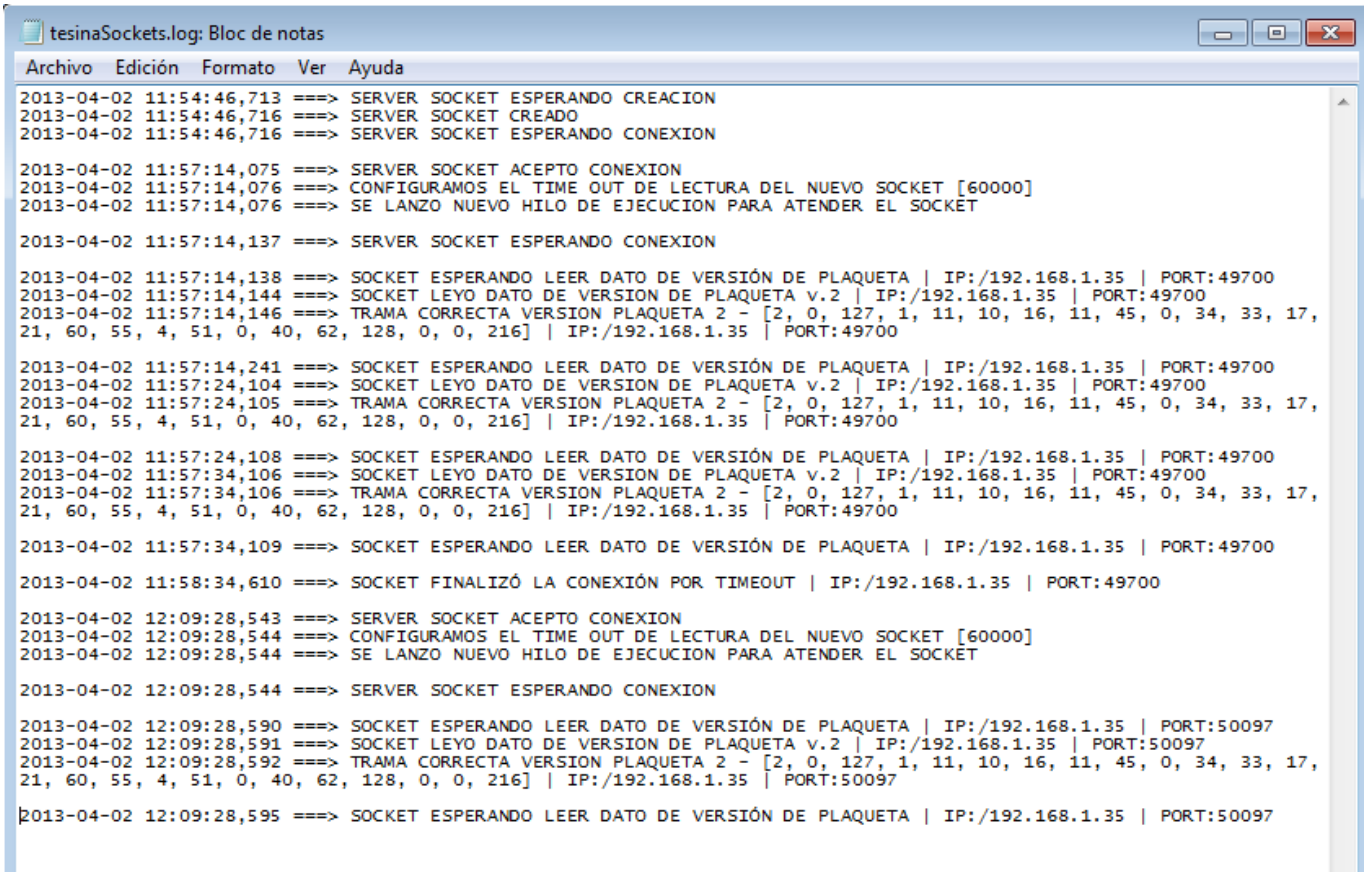
```
1.      public void run() {
2.      String trama;
3.      Socket socketCliente;
4.      int[] tramaByte;
5.      int i;
6.      String strByte;
7.      String strCantReg;
8.      int cantReg;
9.
10.     try{
11.         //CÓDIGO ANTERIOR… VER SECCIÓN "EMULADOR DE EQUIPOS”
12.
13.         //MIENTRAS NO SE DESHABILITE EL HILO MANDA TEMPORALMENTE
            TRAMAS AL SERVIDOR
14.         while (Cliente.estado == 1){
15.             for (i = 0; i<tramaByte.length; i++){
16.                 salida.write(tramaByte[i]);
17.                 salida.flush();
18.             }
19.             Cliente.sleep(delay);
20.         }
21.
22.         //DORMIMOS EL HILO 1 MINUTO
23.         Cliente.sleep(60000);
24.
25.         //CIERRA LA COMUNICACIÓN CON EL SOCKET DEL SERVIDOR
26.         socketCliente.close();
27.
28.     }catch(Throwable t){
29.             new LogExcepcion(t);
30.         }
31.     }
```

Log del Server Socket.

Como se puede observar en el log, el Hilo Receptor que está conectado con el equipo emulado recibe las tramas por el puerto 49700. Cuando detenemos el envío de tramas y el Cliente se duerme un minuto

antes de finalizar la conexión, provoca que el Hilo Receptor finalice por timeout (esto se debe a que el Server Socket le había asignado un minuto de tiempo de timeout).

Luego desde la página principal del Emulador de Equipos se volvió a instanciar un Cliente que ante la solicitud de conexión el Server Socket le asignó un nuevo Hilo Receptor que recibe las tramas en el puerto 50097.



Manejo de excepciones en el Hilo Receptor.

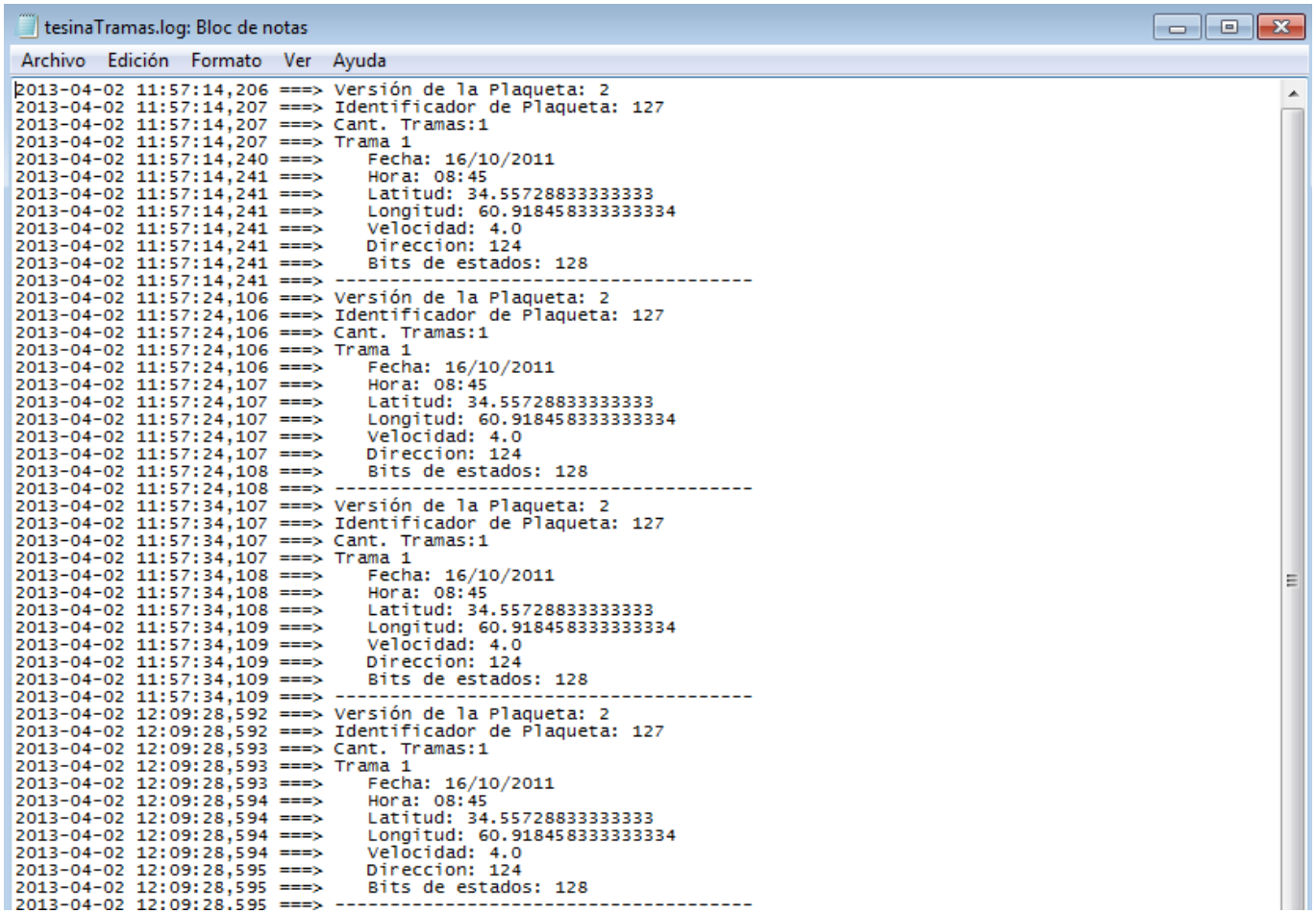
Se muestra los tipos de excepciones que son utilizadas en el Hilo Receptor para capturar los eventos de los sockets y accionar ante lo que está sucediendo.

```
1.   }catch (SocketTimeoutException ste) {
2.           log.info( "SOCKET FINALIZÓ LA CONEXION POR TIME OUT | IP:"
             +this.socket.getInetAddress()+" |
             PORT:"+this.socket.getPort());
3.           //TRATAMIENTO DE ERROR SOCKET TIMEOUT EXCEPCION
4.
5.       }catch (IOException ioe) {
6.           log.info( "SOCKET FINALIZÓ LA CONEXIÓN POR CIERRE DE CONEXIÓN
             | IP:" +this.socket.getInetAddress()+" |
             PORT:"+this.socket.getPort());
7.           //TRATAMIENTO DE ERROR POR CIERRE DE CONEXIÓN. SITUACIÓN NORMAL.
8.
9.       }catch (Throwable t) {
10.          //TRATAMIENTO DE ERROR NO ESPERADO
11.      }finally{
```

```
12.             //VERIFICA SI DEBE CERRAR EL SOCKET
13.         }
```

Manejo de excepciones en el Hilo Receptor.

Se puede observar que las tramas se estaban recibiendo correctamente cada 10 segundos (que es el tiempo que asignamos en la página de Delay), luego se interrumpe un tiempo de 12 minutos aproximadamente y continúa recibiendo nuevamente. Como vimos las primeras se recibieron por un Hilo Receptor y las últimas (luego de los 12 minutos) por otro. Para el resultado final es indistinto el manejo de hilos y sockets.



Modelo de Datos.

El diseño del modelo de datos es una de las claves en la arquitectura de los sistemas telemétricos debido a que, por lo general, estos sistemas poseen las siguientes características:

Tienen una considerable volumetría de datos ya que almacenan todas las lecturas enviadas.

En el ejemplo del Sistema de Monitoreo Satelital es común que un móvil en actividad trasmita una trama entre 30 seg. (móviles ruteros como camiones, colectivos larga distancia, etc.) y 5 seg. (móviles de trabajo en campo como motoniveladoras, tractores, etc.).

Rápido acceso a los últimos datos recibidos (estado actual).

Los sistemas Telemétricos tienen un tablero con información de los últimos estados de los emisores. Por ejemplo en el Sistema de Monitoreo Satelital existe un panel con todos los móviles y por cada uno se visualiza si está activo, fecha/hora de la última lectura y velocidad.

Óptimos tiempos de respuestas.

Información totalizada.

Por lo general estos sistemas totalizan la información para lograr menor tiempo de procesamiento en las consultas y con esto dar mejor tiempo de respuesta. En el Sistema de Monitoreo Satelital se totalizan, por ejemplo, los recorridos diarios de los móviles una vez finalizado el día. Entre los datos totalizados se obtienen cantidad de km. recorridos, velocidad máxima, hora de inicio y finalización de jornada laboral, entre otros.

Acceso a datos verticales.

Como sabemos la instrucción "join" realiza un producto cartesiano en tablas, cuando éstas son voluminosas, la operación es muy costosa en procesamiento y por ende en tiempo de respuesta, aun cuando se hayan creados índices específicos y utilizado la potencia del motor de base de datos donde se está ejecutando. Una característica importante en el diseño es definir tablas totalizadoras o con información del último evento que permitan listar información, estos listados permiten seleccionar un ítem y esa acción desencadena el acceso a una tabla de datos extensa. Un ejemplo es listar las últimas lecturas de los móviles que se encuentran en una tabla de últimas lecturas, y seleccionado un móvil se recuperan todas las lecturas del día actual (existe una tabla por móvil con toda la información).

Contingencia de información.

Cuando se recepciona una trama, se verifica si alguna de las alertas están activas. Ejemplos de alertas son botón de pánico, auxilio mecánico, zonificación, etc. Muchas veces estas alertas generan un evento dentro del sistema, pero en ocasiones hacen que se tengan que comunicar con un sistema externo. Los sistemas telemétricos intentan comunicarse on-line con los sistemas receptores de alarmas, pero si por alguna razón existe una falla en el sistema receptor, se debe almacenar el evento y con una tarea automática periódicamente se debe intentar comunicar para informar el evento. Ejemplos muy comunes se encuentran con sistemas telemétricos que tienen botones de pánico, cuyo sistema recibe el bit de pánico activo y se intenta comunicar con sistemas de empresas de seguridad.

Log de transacciones.

Las principales acciones que se realizan en el sistema son almacenadas con el fin de poder auditar y reconstruir cronológicamente todas las acciones que hicieron los usuarios en el sistema. Algunos de los datos que se recomiendan almacenar son: fecha, usuario, roles del usuario, acción que ejecutó y dirección IP de la máquina de la que estaba operando. El log de transacciones también es utilizado para obtener información estadística respecto a la utilización de cada una de las opciones de la aplicación.

Monitoreo de errores.

Es recomendable implementar aplicaciones telemétricas en lenguajes de programación que brinden un buen manejo de excepciones con el objetivo de guardar el mayor detalle posible del error, línea de código en la que se produjo, trace, usuario o emisor que intentó realizar la acción, etc. Estos sistemas deben almacenar e informar por distintas vías errores con el fin de activar guardias pasivas o advertir ni

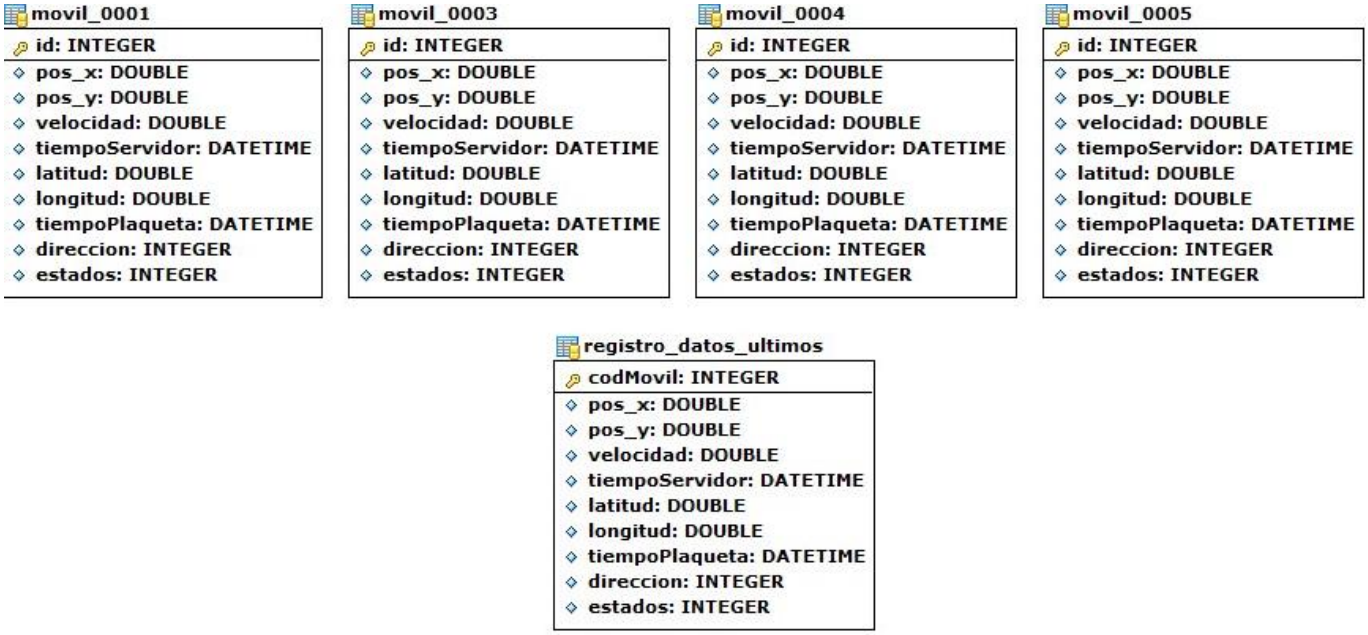
bien ocurrido el error a centros de monitoreo para que puedan comunicarse con los sectores técnicos encargados de resolver los inconvenientes.

Para satisfacer las características anteriores se listan consideraciones del diseño del modelo de datos para sistemas telemétricos:

Cada emisor posee su tabla de datos donde se almacena la información de las tramas enviadas por el equipo del emisor. Se hace un insert por cada trama enviada del equipo del emisor.

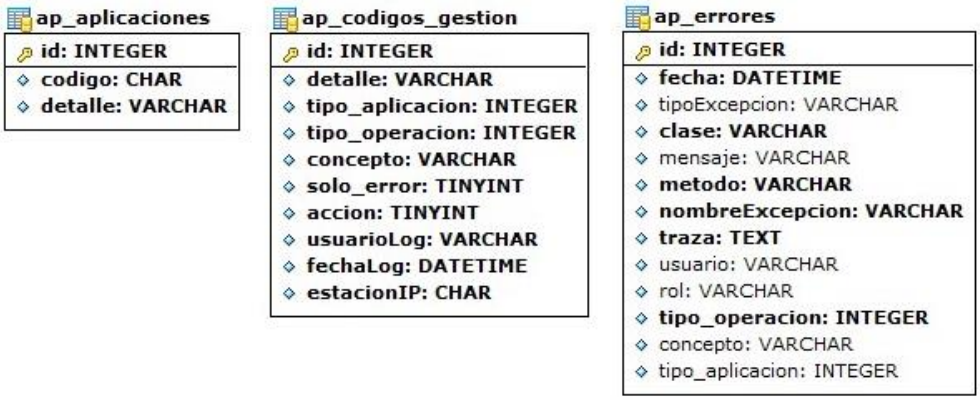
Siempre hay tablas por emisores, no por identificación de los equipos. Esto se debe a que los equipos se reemplazan por ruptura.

Existe una tabla con los datos de las últimas tramas de cada uno de los emisores. Cuando se recepciona una trama, se inserta en la tabla del emisor y en la tabla de últimos datos.



Existe generalmente un grupo de tablas relacionadas que se utilizan para almacenar el log de transacciones de los usuarios. Una de las tablas posee información de cada una de las opciones de la aplicación y la otra de las transacciones generadas por las aplicaciones.

Un factor importante es la detección, almacenamiento y notificación de errores o anomalías que se produzcan en los sistemas. Los sistemas telemétricos generalmente por contingencia almacenan los errores en más de un repositorio (los más utilizados son archivos de log y bases de datos). Al producirse un error en el sistema, se informa por diferentes vías. Entre las más comunes se hallan: alerta en un tablero de control, envío de email, llamado telefónico y envío de mensaje de texto.



SOFTWARE DE VISUALIZACIÓN.[5]

El otro componente software en los sistemas Telemétricos es la aplicación que gestiona y visualiza los datos enviados por los equipos que fueron almacenados por la aplicación de recepción de tramas.

Las aplicaciones que visualizan los datos varían de acuerdo a preferencias de los usuarios, entidades que se están monitoreando, tipo de monitoreo (centro de monitoreo con guardia activa o pasiva), dispositivo en los que se debe visualizar, etc.

A continuación se muestran las principales pantallas de diferentes aplicaciones telemétricas con el objetivo de exponer consideraciones respecto al diseño de interfaz y optimización con el modelo de datos.

[5] Suministrado por la empresa SkyControl. www.skycontrol.com.ar, consultado el 10/07/2013.

Pantalla principal de un sistema de monitoreo satelital.

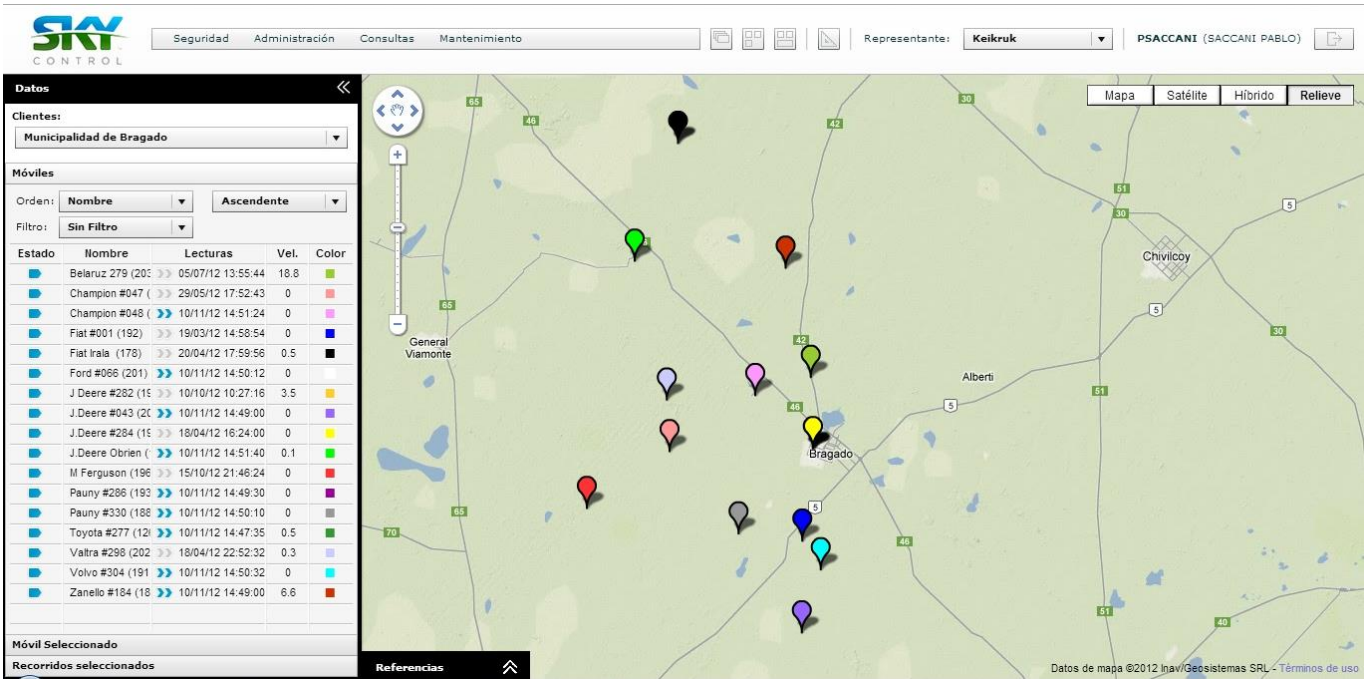


Figura 3.1. Sistema de Monitoreo Satelital SkyControl – Ventana inicial (suministrada por SkyControl SH)

Está compuesta por tres bloques:

Barra superior: contiene la barra de menúes, herramientas rápidas, representante y botón de cierre de sesión.

Panel de datos: está compuesto por una componente acordeón con información de los móviles. Al seleccionar un móvil la segunda sección del acordeón muestra información detallada y la última información de los recorridos. El panel de datos permite ser desplazado a la izquierda con el objetivo de que el mapa cubra la mayor superficie de la pantalla.

Cartografía: muestra las posiciones actuales de los móviles en el mapa. Pueden existir varias vistas, además de las típicas de GMaps, se puede visualizar cartografía propia del usuario. Por ejemplo, si es un municipio, una con caminos rurales que no son provistos por las principales compañías que ofrecen imágenes satelitales.

Panel de datos.

Exhibe los móviles asociados al cliente. De cada móvil se visualiza el estado, nombre, fecha y hora de la última lectura, velocidad y color.

El estado denota la situación en que se encuentra el móvil. Esta opción es modificada por el administrador encargado de los móviles.

Delante de las lecturas se encuentra un indicador “>>” que representa el estado actual de las lecturas del móvil según el tiempo de la última recepción.

Desde el punto de vista del diseño, el panel consume un servicio que accede a la tabla que tiene las últimas lecturas de los móviles.

El panel posee un Timer que cada cierto tiempo parametrizado en la aplicación realiza la invocación y refresca los datos, dando la sensación de un seguimiento totalmente on-line.

Seleccionando un móvil el componente acordeón se desplaza al segundo panel donde se muestra información del móvil seleccionado.

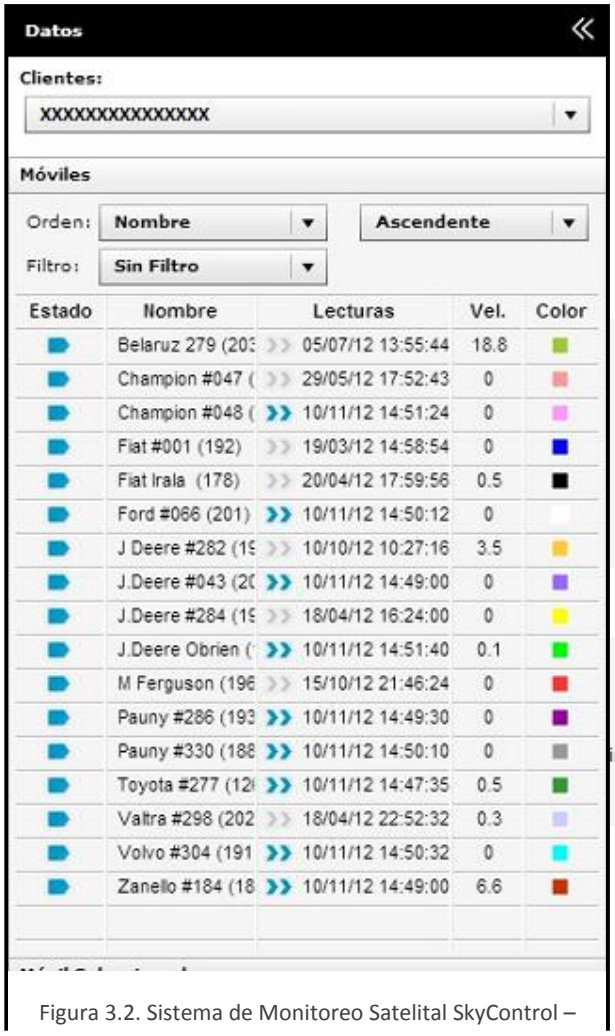


Figura 3.2. Sistema de Monitoreo Satelital SkyControl – Panel de móviles (suministrada por SkyControl SH)

El panel muestra información del móvil, de la última lectura y a través de un servicio de geocoding reverso, dada las coordenadas, se obtiene la descripción de la ubicación.

Existen botones para centrar el móvil en el mapa, zoom positivo, negativo y para mostrar las referencias de velocidades.

La funcionalidad *"Mantener centrado en mapa"* se utiliza para fijar el móvil en pantalla y a medida que avanza se mueve la cartografía en vez del móvil. Es útil para hacer seguimiento de un vehículo sin que éste se desplace de la pantalla.

La funcionalidad *"Mantener recorrido en mapa"* muestra el recorrido de la fecha y a medida que avanza el móvil lo va actualizando. Para mostrar el recorrido, el sistema accede a la tabla donde están todas las lecturas del móvil. Como se accede en forma vertical a una tabla sin hacer cruces con otras, la velocidad de respuesta es buena. Debido a que hay móviles, como por ejemplo sembradoras y cosechadoras, que estando en movimiento envían tramas cada 5 segundos, se tuvo que optimizar lo máximo posible el algoritmo y manejo de la memoria del lado de la máquina del cliente para que la gráfica y cálculos sean eficientes.

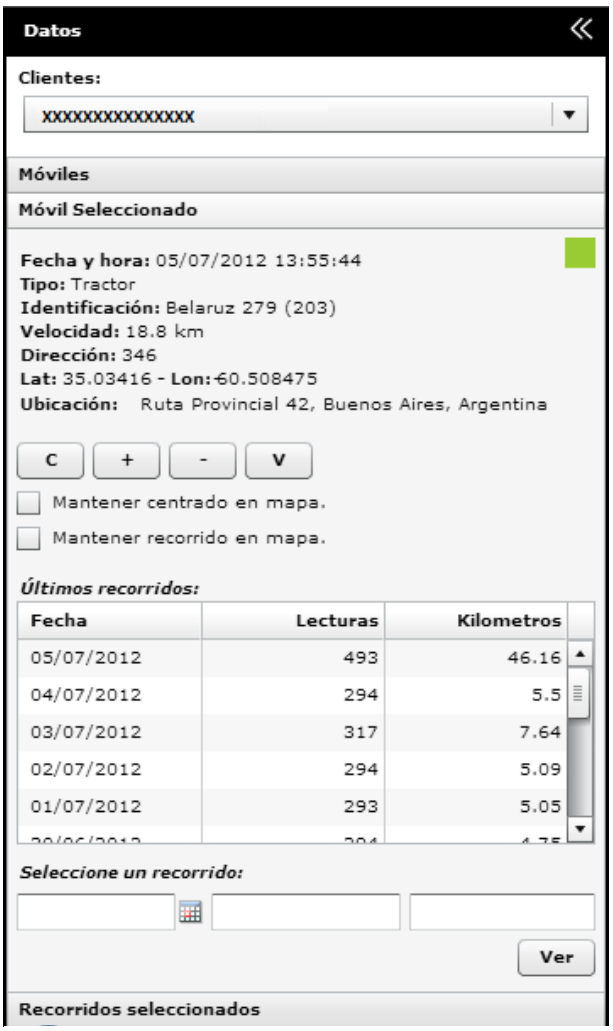


Figura 3.3. Sistema de Monitoreo Satelital SkyControl – Información de un móvil (suministrada por SkyControl SH)

Por último, se muestran los últimos 20 recorridos del móvil con información totalizada (cantidad de lecturas y km.). Si se desea visualizar un recorrido anterior a los últimos 20 se debe buscar la fecha en el calendario y presionar el botón Ver. Los datos son obtenidos de una tabla totalizada que contiene información diaria, la cual se procesa por la noche finalizado el día (después de las 00hs), al acceder a una tabla totalizada el tiempo de respuesta es bueno.

Seleccionando un recorrido, en el caso del ejemplo la fecha 05/07/2012, se visualiza el recorrido centrado en el mapa y en el panel los Detalles obtenidos de las tablas totalizadoras. La gráfica se realiza utilizando componentes de la API de GoogleMaps.

Seleccionando el check *"Ver referencias entre lecturas"* se visualizan en el mapa puntos con información de la lectura. Dado que es costoso tener en la máquina cliente (Flash Player) una colección grande de puntos con información, el componente *"Cantidad de lecturas"* conformado por una barra deslizante, permite acotar la cantidad especificando, el intervalo de lecturas entre los que se visualizaran los puntos de información.

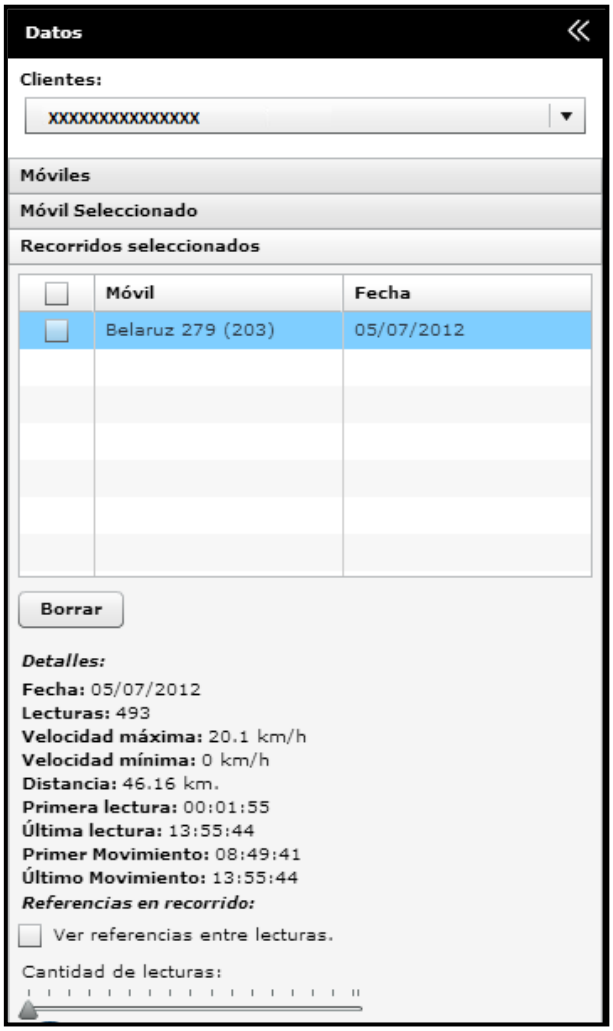


Figura 3.3. Sistema de Monitoreo Satelital SkyControl – Panel de recorridos (suministrada por SkyControl SH)

Figura 3.4. Sistema de Monitoreo Satelital SkyControl – Recorrido entre dos ciudades (suministrada por SkyControl SH)

Pantalla principal de un sistema telemétrico de monitoreo de cisterna y digestor.

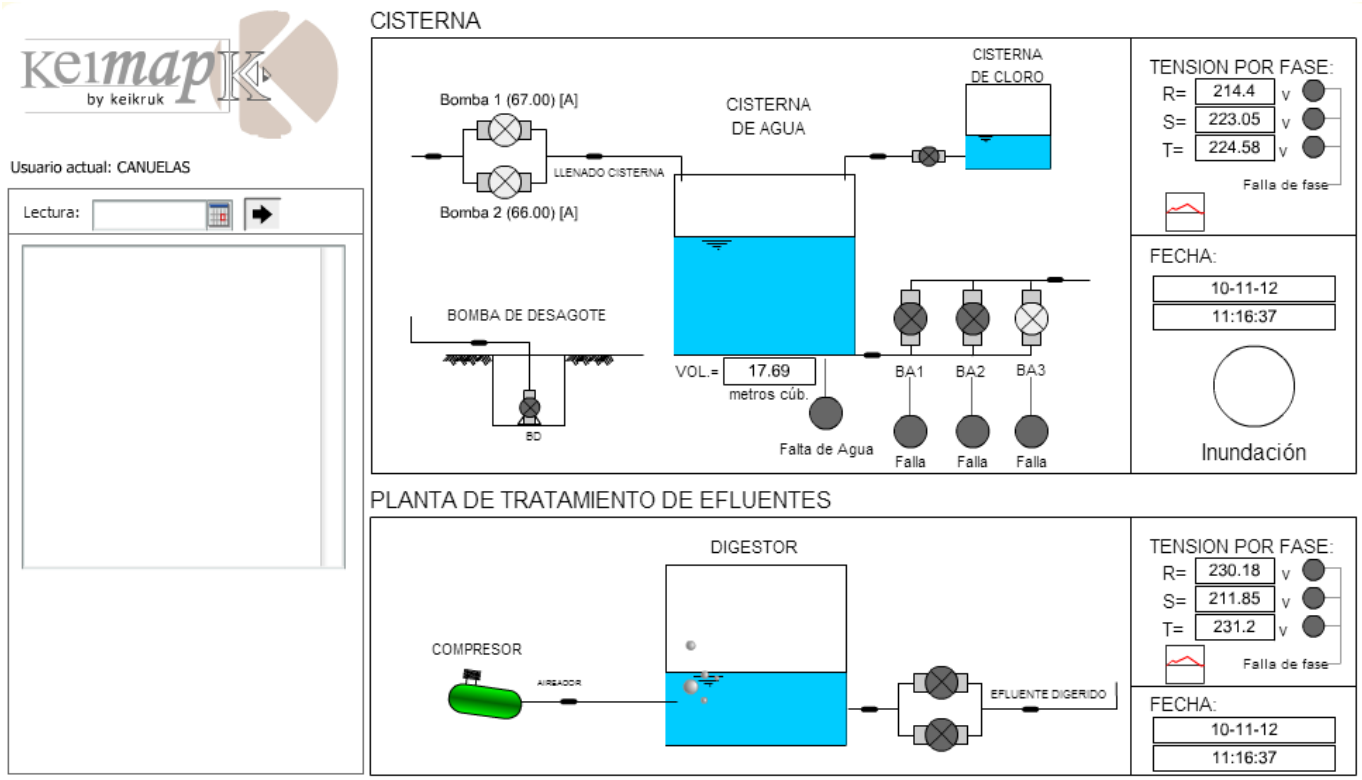


Figura 3.3. Sistema de Telemetría de Cisterna y Planta de tratamiento de efluentes (suministrada por Keikruk SH)

Conclusiones y Trabajos futuros

Los Sistemas Telemétricos son utilizados en un amplio rango de aplicaciones. Algunas soluciones se diseñan de forma específica por tener que cumplir un objetivo determinado, requiriendo generalmente acceder a lugares inalcanzables por el ser humano u obtener mediciones con mucha precisión. Estos tipos de soluciones tienen elevados costos ya que, además del software, se deben fabricar transmisores y receptores especiales, y muchos de ellos también deben implementar un medio de transmisión propio. La mayoría de estos sistemas suelen ser fabricados a pedido de organismos gubernamentales y misiones militares, solo un subconjunto de ellos se producen con el fin de ser comercializados masivamente. Existe otro grupo de aplicaciones telemétricas que crecieron masivamente en las últimas dos décadas, han crecido tanto como lo ha hecho Internet y la red de comunicación GSM. Estas componentes, junto con sus protocolos de comunicación, permitieron que los ingenieros reutilicen canales de comunicación para implementar soluciones telemétricas, abaratando de manera notable los costos. Muchos de estos sistemas son utilizados para monitorear plantas metalúrgicas, estaciones meteorológicas, flotas de vehículos, sistemas de seguridad en la vía pública, entre otras. El rango de aplicaciones y componentes a ser monitoreados y/o controlados remotamente es tan amplio como elementos a los que se les puede leer y/o enviar información. Este grupo de aplicaciones se desarrollan mayoritariamente con fines comerciales, permitiendo además disparar distintos tipos de alarmas y alertas. El trabajo de esta tesina se basó principalmente en este segundo grupo de soluciones.

Las primeras secciones del trabajo presentan y describen las componentes intervinientes en los sistemas telemétricos. El objetivo de éstas es presentar los conceptos que, directa o indirectamente, son utilizados en la sección donde se implementa una solución genérica de comunicación que puede ser utilizada para cualquier tipo de sistema telemétrico de similares características.

La solución implementada consiste en la creación de un socket administrador que escucha en un determinado puerto de un servidor. Por cada requerimiento de conexión externa, el socket administrador crea un socket cliente que será el encargado de dialogar con el transmisor mientras dure la comunicación. La comunicación puede ser unidireccional o bidireccional. Existen dos características sobresalientes, una de ellas es que presenta un mecanismo de recuperación ante reinicios de servidores en la solución: el sistema siempre estará activo mientras el servidor donde se está ejecutando se encuentre activo. La otra es la autoadministración de memoria, ésto se debe a que cuando finaliza una comunicación, ya sea de forma acordada por las partes o por problemas del canal de comunicación, el socket cliente se autodestruye. De esta forma el sistema mantiene en ejecución sockets clientes activos.

Un uso muy común cuando se desarrollan estas soluciones, es disponer de un simulador que permite representar diferentes escenarios de pruebas, sin la necesidad de contar con transmisores físicos. Este software emulador tiene dos grandes ventajas: una de ellas radica en que permite el desarrollo en paralelo del software receptor (o receptor / transmisor si la comunicación es bidireccional) mientras se fabrica, ensambla y programa el equipo (emisor / receptor que estará en las componentes remotas). La otra reside en que se pueden realizar pruebas de stress sin la necesidad de contar con equipos físicos. Por ejemplo, en un sistema de monitoreo satelital de vehículos podría realizarse una prueba de más de mil equipos transmitiendo en simultáneo sin la necesidad de contar con más de mil equipos.

El software implementado básicamente es una administración de sockets TCP, donde si se reemplaza el módulo que interpreta la trama enviada por la interpretación de otra trama, puede ser utilizado para cualquier solución telemétrica, desde monitoreo satelital de vehículos, como es el caso que se ejemplificó en este trabajo, hasta monitoreo y control de plantas metalúrgicas, silos, etc. Lo interesante es que puede ser utilizado como un canal de comunicación para enviar y recibir información prácticamente desde y hacia cualquier lugar del mundo. Esto es, cualquier elemento que se monitoree localmente, ya sea por un medio cableado, WIFI, Bluetooth, etc., puede ser monitoreado o controlado desde cualquier lugar del mundo con un costo muy bajo, solo se requiere Internet y red GSM.

En cuanto a trabajos futuros son varias las líneas de acción que se pueden realizar:

En lo que respecta a componente software de recepción / envío de información, se puede implementar una solución análoga bajo el protocolo UDP, permitiendo abaratar costos en las transmisiones, dado que al ser éste un protocolo no orientado a la conexión, los datos de control son considerablemente inferiores al utilizado por el protocolo TCP. También se puede ampliar la solución para contar con un panel de monitoreo, donde se informe cantidad de sockets activos, cantidad de sockets que se autodestruyeron, lo que permitiría conocer el estado del canal, ya que si hay muchos sockets que se autodestruyen significa que hay muchas reconexiones, cantidad de información transferida, etc. También se puede contabilizar volumen de información transferida y hacer un chequeo estimativo con el costo facturado por la empresa proveedora del canal de comunicación.

En lo que respecta a sistemas telemétricos, la solución lograda puede aplicarse y adaptarse para ser utilizada en cualquier sistema que actualmente está implementada localmente, algunos de ellos son: visualización a distancia de variables de monitores de siembra de maquinarias agrícolas, reparación a distancia de maquinarias agrícolas, control remoto de robots.

Apéndice 1: Medición de errores del Sistema GPS

Resumen de las fuentes de error del sistema GPS Errores típicos, en Metros (Por cada satélite)			
Fuentes de Error	**GPS Actual Desde 2/5/2000**	**GPS Standard Hasta 2/5/2000**	**GPS Diferencial**
Reloj del Satélite	1.5	1.5	0
Errores Orbitales	2.5	2.5	0
Ionosfera	5.0	5.0	0.4
Troposfera	0.5	0.5	0.2
Ruido en el Receptor	0.3	0.3	0.3
Disponibilidad Selectiva	0	30	0
Exactitud Promedio de la Posición			
Horizontal	15	50	1.3
Vertical	24	78	2.0
3-D	28	93	2.8

Conclusiones.

La ionosfera y la troposfera causan demoras en la señal de GPS que se traducen en errores de posicionamiento.

El GPS Diferencial puede eliminar casi todos los errores.

APÉNDICE 2: TRAMA DE DATOS GSM[6]

* La trama está compuesta por: (**5 + 20 x CANT_REGISTROS)** bytes

  4 bytes encabezado

  20 bytes por cada registro/dato enviado

  1 byte con suma de comprobacion

* Cada byte puede tomar valores entre 0 y 255

DECODIFICACION de la trama con Función = 2
1) Se lee el byte 4 y se corrobora la cantidad de bytes recibidos: m = 5 + 20 x b4
2) Se verifica la suma de comprobación:
   suma = b1 + b2 + b3 + … + bm-1
   suma = suma MOD 256 (resto de dividir por 256)
   SI suma = bm → datos OK
3) Se desarma la trama y leen los respectivos registros

CAMPO			RANGO	BYTES	
Función = **2**			0..255	b1	Este campo distingue la trama respecto de la implementación anterior
Nro Serie			0..65535	b2	Hi
				b3	Lo
Cant. Registros			0..255	b4	
Registro 1	Fecha	Año	0..255	b5	Desde año 2000 en adelante
		Mes	1..12	b6	
		Dia	1..31	b7	
	Hora	Hora	0..23	b8	
		Min	0..59	b9	
		Seg	0..59	b10	
	Latitud	Grados	0..89	b11	
		Minutos	0..59 100..159	b12	Min = 0..59 Min = 100..159
		Decimales de Minutos	0..9999	b13	Hi
				b14	Lo Decimales =
	Longitud	Grados	0..179	b15	

[6] Suministrado por la empresa SkyControl. www.skycontrol.com.ar, consultado el 10/07/2013.

		Minutos	0..59 100..159	b16	Min = 0..59 Min = 100..159
		Decimales de Minutos	0..9999	b17	Hi
				b18	Lo Decimales =
	Velocidad	Unidades de 0.1 Km/h	0..65535	b19	Hi
				b20	Lo Vel [Km/h] =
	Course Over Ground	Unidades de 2 grados	0..179	b21	Dirección (0..358) = 2 * b21
	Status		0..255	b22	FLAGS de STATUS (Operación de la máquina) LSBit = Flag1, ….., MSBit = Flag8
	Libre		0..255	b23	
	Libre		0..255	b24	
Registro 2				b25 .. b34	
Tantos registros como se defina en **b4**					
Suma (en 8 bits) de **b1** a **bm-1**				bm	

Referencias Bibliográficas y Entrevistas

Bibliografía

BRICEÑO MÁRQUEZ, José E. (2005): "Transmisión de Datos", Tercera edición (Edición digital). Universidad de los Andes, Facultad de Ingeniería, Departamento de Publicaciones, Mérida, Venezuela.

HOFMANN-WELLENHOF, B., LICHTENEGGER, H., COLLINS, J. (1997). "GPS theory and practice".

MARCANO, Florencia; MARCANO Pedro (2012): "Sistemas de Radiolocalización y Telemetría". Ingeniería en Telecomunicaciones, UNEFA, Venezuela. En http://es.scribd.com. Consultado en noviembre de 2012.

PACHÓN DE LA CRUZ, Álvaro (2004): "Evolución de los sistemas móviles celulares GSM". Universidad ICESI, Revista Sistemas y Telemática, Vol.2 Nro.4.

TANENBAUM, ANDREW S. (1997): "Redes de Computadoras", Tercera edición. Prentice Hall Hispanoamericana, ISBN 968-880-958-6.

DAVID MUÑUZ RODRIGUEZ (2002): "Sistemas Inalámbricos de Comunicación Personal". Alfaomega Grupo Editor, ISBN 970-15-0516-6.

Referencias

[1] Para una profundización sobre Sistemas SCADA, se sugiere: BRICEÑO MÁRQUEZ, José E. "Transmisión de Datos", pág. 227 a 230.

[2] "Durante más de un siglo, la principal infraestructura de telecomunicaciones internacional ha sido el sistema telefónico público de conmutación de circuitos. Este sistema se diseñó para la transmisión analógica de voz y es inadecuado para las necesidades de las comunicaciones modernas. Anticipando una demanda considerable por parte de los usuarios de un servicio digital de extremo a extremo, las compañías de teléfonos y las PTT se unieron en 1984 bajo los auspicios de la CCITT y estuvieron de acuerdo en construir un sistema de teléfonos de conmutación de circuitos nuevo, completamente digital, para principios del siglo XXI. Este nuevo sistema llamado ISDN, tiene como meta principal la integración de servicios de voz y sin vos." TANENBAUM, ANDREW S. (1997), pág. 139.

[3] Para una profundización sobre GSM, se sugiere: TANENBAUM, Andrew "Redes de Computadoras", pág. 266 a 275.

[4] "Identificación de componentes", "Conexión del emisor con el medio y el receptor", y "Conceptos" en base a entrevista Ing. Electrónico Baldoma, Germán.

[5] Características técnicas prototipo suministrado por Ing. Electrónico Baldoma, Germán.

[6] Gutovnik, Pedro en http://gutovnik.com/como_func_sist_gps.htm extraído el 03/04/2013

[7] Para una profundización sobre Trilateración, se sugiere: Gende, M., admin. (2011, June 06). Trilateración. Retrieved November 25, 2013, from Facultad de Ciencias Astronómicas y Geofísicas Web site: http://catedras.fcaglp.unlp.edu.ar/geofisica/referenciacion-en-geofisica/teoria/instrumental-y-tecnicas-topograficas/trilateracion.

[8] International Telecommunication Union (ITU) - GSM Case Study en http://www.itu.int/osg/spu/ni/3G/casestudies/GSM-FINAL.pdf

[9] Dirección General de Protección Civil y Emergencia, Mrio. del Interior, Gobierno de España. Publicado en (www.proteccioncivil.org/catalogo/carpeta02/carpeta24/vademecum12/vdm036.htm), extraído el 01/05/2014.
También se ha consultado Publicación Estándar GSM (Sistema global de comunicaciones móviles) en http://es.kioskea.net/

[10] Para una profundización sobre el modelo de referencia OSI, se sugiere: TANENBAUM, Andrew "Redes de Computadoras", pág. 28 y 34.

[11] Para una profundización sobre UDP, se sugiere: TANENBAUM, Andrew "Redes de Computadoras", pág. 542 a 545.

[12] Para una profundización sobre TCP, se sugiere: TANENBAUM, Andrew "Redes de Computadoras", pág. 521 a 542.

Entrevistas

BALDOMA, Germán (Ing. Electrónico): Empresa TECMES. Entrevista realizada el 28/12/2012.

BARISICH, Nicolás (Ing. Electrónico): Empresa BARISICH ELECTRÓNICA. Entrevista realizada el 02/01/2013.