ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARREA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

PROYECTO DE GRADO PARA LA OBTENCIÓN DEL TÍTULO EN INGENIERIA

DISEÑO, IMPLEMENTACIÓN Y EVALUACIÓN DE UN SISTEMA DE MONITORIZACIÓN PARA CÁMARAS DE REFRIGERACIÓN EN TRANSPORTE DE ALIMENTOS PERECEDEROS UTILIZANDO LA RED GPRS

Gabriel Alejandro Guzmán Calderón

Paulina Elizabeth Proaño Sotomayor

SANGOLQUÍ – ECUADOR

2013

CERTIFICACIÓN

Por medio de la presente se certifica que el Proyecto de Tesis de Grado: “DISEÑO, IMPLEMENTACIÓN Y EVALUACIÓN DE UN SISTEMA DE MONITORIZACIÓN PARA CÁMARAS DE REFRIGERACIÓN EN TRANSPORTE DE ALIMENTOS PERECEDEROS UTILIZANDO LA RED GPRS”ha sido elaborado en su totalidad por los señores Gabriel Guzmán Calderón con CI: 171593834-4 y Paulina Proaño Sotomayor con CI: 171793257-6, ha sido guiado y revisado periódicamente y cumple normas estatutarias establecidas por la Universidad de las Fuerzas Armadas - ESPE, en el Reglamento de Estudiantes de la Universidad de las Fuerzas Armadas - ESPE.

Ing. Freddy Acosta	Ing. Rodolfo Gordillo
DIRECTOR	CODIRECTOR

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE
DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA
CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

DECLARACIÓN DE RESPONSABILIDAD

Gabriel Alejandro Guzmán Calderón y Paulina Elizabeth Proaño Sotomayor

DECLARAN QUE:

El proyecto de grado denominado "DISEÑO, IMPLEMENTACIÓN Y EVALUACIÓN DE UN SISTEMA DE MONITORIZACIÓN PARA CÁMARAS DE REFRIGERACIÓN EN TRANSPORTE DE ALIMENTOS PERECEDEROS UTILIZANDO LA RED GPRS", ha sido desarrollado con base a una investigación exhaustiva, respetando derechos intelectuales de terceros, conforme las citas que constan al pie, de las páginas correspondientes, cuyas fuentes se incorporan en la bibliografía.

Consecuentemente este trabajo es de nuestra autoría.

En virtud de esta declaración, nos responsabilizamos del contenido, veracidad y alcance científico del proyecto de grado en mención.

Sangolquí, 01 de Octubre de 2013

_____ _____

Gabriel Guzmán Calderón Paulina Proaño Sotomayor

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE
DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA
CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

AUTORIZACIÓN

Gabriel Alejandro Guzmán Calderón
Paulina Elizabeth Proaño Sotomayor

Autorizo a la Universidad de las Fuerzas Armadas - Escuela Politécnica del Ejército la publicación, en la biblioteca virtual de la Institución del trabajo "DISEÑO, IMPLEMENTACIÓN Y EVALUACIÓN DE UN SISTEMA DE MONITORIZACIÓN PARA CÁMARAS DE REFRIGERACIÓN EN TRANSPORTE DE ALIMENTOS PERECEDEROS UTILIZANDO LA RED GPRS", cuyo contenido, ideas y criterios son de mi exclusiva responsabilidad y autoría.

Sangolquí, 01 de Octubre de 2013

Gabriel Guzmán Calderón

Paulina Proaño Sotomayor

DEDICATORIA

Esta tesis dedico a mi abuelita, tía, padres, y hermanas, quienes en cierto momento me brindaron su apoyo moral y psicológico para poder llegar a esta instancia de mis estudios, especialmente a mi madre que nunca me negó su apoyo durante estos años universitarios.

También la dedico a mis amigos con quienes pase algunas noches de desvelo y entre cortocircuitos, risas y estudios siempre fue agradable su compañía a lo largo de toda la carrera, y a pesar de problemas presentados en mi vida, nunca faltaron ellos para sacarme una sonrisa.

Paulina Proaño Sotomayor

Este trabajo lo dedico a mis padres por el gran esfuerzo que significó para ellos darme la educación para culminar mi carrera, a mi hermano por estar a mi lado en todo momento, a mis tíos por el gran apoyo que supieron brindarme, a mis amigos por los momentos especiales que pasamos en nuestra vida universitaria.

Gabriel Guzmán Calderón

AGRADECIMIENTOS

El agradecimiento de mi tesis es principalmente a Dios quien me mantiene aún con vida y me ha dado fortaleza durante estos años para seguir adelante.

A los catedráticos de mi carrera por quienes he llegado a obtener los conocimientos necesarios para poder desarrollar este proyecto de grado, de manera especial a las siguientes personas Ing. Freddy Acosta, Ing. Rodolfo Gordillo quienes nos guiaron en este proyecto de grado.

Agradezco también a la empresa Erictel Telefónica quienes nos abrieron sus puertas para realizar este proyecto de grado y nos brindaron su apoyo incondicional.

Y no podía faltar agradecer a mi compañero de tesis que más que compañero desde años atrás es un gran amigo, al igual que a los demás grandiosos seres humanos que conocí a lo largo de mi carrera y que ahora forman parte de mi vida.

Paulina Proaño Sotomayor

Agradezco a mi familia por su comprensión y por todo el apoyo que supieron brindarme para haberculminado esta maravillosa etapa de mi vida, a mis amigos por estar ahí dándome ánimos en momentos difíciles para enfrentar cualquier adversidad, a mis compañeros por el apoyo moral con el cual supieron incentivarme.

Gabriel Guzmán Calderón

PRÓLOGO

En la actualidad, el transporte de alimentos perecederos no cuenta con un sistema para control de la cámara de refrigeración móvil, por lo tanto es necesario implementar un sistema para conocer en todo momento el estado de temperatura de productos alimenticios que son distribuidos a los diferentes centros de acopio.

Uno de los desafíos es diseñar e implementar un sistema M2M para monitorización de cadena de frío en tiempo real, en transportes de alimentos, utilizando la red GPRS de Movistar con equipos OWASYS, y que sea capaz de transmitir datos contenidos en los dispositivos de forma autónoma; el equipo (OWASYS) integrará funciones de conectividad dedicada desde la red fija a la red GPRS, que proporcionará medidas en tiempo real de la monitorización remota de temperatura, aceleración, apertura de puertas y ubicación del camión por medio del sistema GPS que tiene integrado. Para este proyecto se utiliza el dispositivoowa33A, para realizar el sistema M2M, trabajando con una SIM card con APN específico para soluciones corporativas de Movistar; se realiza un estudio de sensores, para implementar los adecuados para el tipo de medición que se requiere en el proyecto.

Otro de los desafíos de este proyecto es realizar el levantamiento de un servidor, una base de datos y una página web en la que se presentará un historial en donde se detalla los datos de temperatura, apertura de puertas y aceleración con su respectivafecha y hora; logrando con esto supervisar el estado de los alimentos al ser transportados, sin que estos se desperdicien y ocasione pérdidas económicas para la empresa.La central de comunicación tendrá como funcionalidad almacenar la información del estado de las cámaras de refrigeración de los camiones y permitir visualizar esta información desde un celular o una PC, con acceso a la web diseñada para este proyecto; en caso de presentar alguna novedad de los datos adquiridos en la plataforma, el usuario recibirá un mensaje de texto a su celular.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El estado del arte de este proyecto contempla el diseño implementación y evaluación de un sistema de mando y monitorización remota con comunicación M2M, propuesta por la empresa telefónica Movistar. Con el desarrollo de este sistema de comunicación se puede realizar la monitorización de cámaras de refrigeración móvil para transporte de alimentos perecederos a fin de preservar la inocuidad y la aptitud del producto alimentario, conocer la posición del container, apertura de puertas y presentar información de algún movimiento brusco en su trayecto hacia el destino final. Se realiza un análisis de una arquitectura M2M, estudio del funcionamiento del equipo owa33A, selección del sensor de temperatura y movimiento, levantamiento de un servidor para desarrollo de una plataforma web, además de pruebas de funcionamiento y tiempo de respuesta de las señales, comprobando el correcto funcionamiento del equipo a través de resultados reales.Se realizó unanálisis financiero para determinar beneficios o perdidas que pueda tener la implementación de este proyecto, un estudio de inversión y costos para determinar si el proyecto es rentable o no.El proyecto está basado en cuatro puntos reales para lograr la comunicación M2M, los cuales se indican a continuación:

1° Etapa de indagación
2° Etapa de acondicionamiento de señales
3° Etapa de desarrollo de software
4° Etapa de análisis de resultados

CAPÍTULO 1

1.1 ANTECEDENTES

En la actualidad vivimos en una era tecnológica, en la que la información es transmitida a través de las redes celulares mediante el uso de mensajes cortos que son una de las maneras de comunicación de forma rápida y confiable hacia los usuarios que requieren un determinado servicio. Es fundamental contar con un sistema de comunicación veraz y efectivo de transporte de alimentos perecederos para el monitoreo de temperatura, humedad, movimiento en cámaras de refrigeración, túneles de frío, hornos, salas de proceso en empresas de exportación de mariscos, industrias lácteas o de flores, sin estar necesariamente presente en el lugar y que permite tener una eficiencia energética adecuada.

En los procesos de monitoreo tradicionales se obtenía el estado de la cámara de frío en el mismo sitio de trabajo, lo que provocaba que si los usuarios se encontraban fuera del lugar y existía algún problema, los alimentos dentro de la cámara se echaban a perder y por lo tanto provocaba pérdidas en la economía de la empresa. Sin embargo es importante considerar la posibilidad de desarrollar un sistema de comunicación M2M (Machine to Machine) en la que se pueda explotar la red móvil para transmisión y recepción de la información en tiempo real en cualquier momento y desde cualquier lugar mediante la utilización de nuevos equipos

que evitan una menor sobrecarga de trabajo y una mayor tranquilidad a los usuarios.

Tomando en cuenta lo antes mencionado es propicio innovar nuevas tecnologías que estén enfocadas a la telemetría, con la finalidad de facilitar y brindar una mejora en la monitorización de una cámara de refrigeración móvil con los equipos gestionados por las personas interesadas en desarrollar el proyecto y que han sido adquiridos por Telefónica Movistar después de haber realizado el estudio tecnológico correspondiente para la implementación de la solución.

1.2 JUSTIFICACIÓN E IMPORTANCIA

1.2.1 Justificación

El propósito de este proyecto de monitoreo utilizando la red GPRS, es brindar una solución óptima y de bajo costo para el seguimiento de los camiones o container con cámaras de refrigeración asegurando que el producto transportado esté de acuerdo a las normas fitosanitarias, de seguridad alimentaria y física para que personas responsables o jefes de una empresa de alimentos puedan verificar que los bodegones de refrigeración o contenedores fríos estén siendo usados de la manera correcta, llegando las alarmas a dispositivos de uso personal como celulares o computadores.

Para el desarrollo del proyecto se trabajará con equipos robustos y de alta interoperabilidad, que funcionan con un sistema operativo Linux con un Kernelcustomizado, es decir, desarrollado especialmente para este tipo de dispositivos y que a diferencia de otros equipos se obtendrá una mejor eficiencia energética. Además, permite el monitoreo a distancia del área de trabajo y facilita conocer en cualquier momento el estado en que se encuentra el producto.

Teniendo como referencia las necesidades esenciales e indispensables, se requiere emprender un cambio hacia la modernización de los sistemas de telemetría, mediante un operador móvil haciendo uso de mensajería SMS para así tener mayor cobertura en el servicio y sobretodo conllevar a que se establezcan mejorías en la atención consiguiendo así una respuesta rápida y eficiente.

1.2.2 Importancia

El sistema está encargado específicamente de proporcionar máximo control sobre las cámaras de refrigeración móvil, ya que permite conocer la temperatura, tolerancias térmicas, apertura de puertas en forma remota, online y en tiempo real, junto a un sistema de posicionamiento global GPS.Una de las ventajas es tener la tranquilidad, integridad, confidencialidad, disponibilidad, ya que se respalda la información en los servidores para ser usada cuando sea necesario, sin riesgo de pérdida de datos por tener sistemas centralizados. Otra de las ventajas,es el uso de recursos de índole tecnológico como portales web corporativos, desarrollados especialmente para que estos equipos puedan conectarse a servidores para conocer con mayor profundidad lo que ocurre en el sitio de trabajo.

1.3 ALCANCE

El estado del arte de este proyecto contempla el diseño, implementación y evaluación de un sistema de mando y monitorización remota con comunicación M2M (Machine to Machine), propuesto por la empresa Telefónica Movistar. Con el desarrollo de este sistema de comunicación se puede realizar la monitorización de cámaras de refrigeración móvil para transporte de alimentos perecederos a fin de preservar la inocuidad y la aptitud del producto alimentario, conocer la posición del container, apertura de puertas y si tiene movimientos bruscos en su trayecto hasta llegar el destino final.

El proyecto está basado en cuatro puntos importantes para lograr la comunicación M2M, los cuales son: etapa de indagación, etapa de acondicionamiento de señales, etapa de desarrollo de software y etapa de análisis de resultados.

En la etapa de indagación se realizará una descripción referente a la tecnología M2M y su arquitectura para una comunicación en tiempo real, la tecnología con la que se trabajará tal como GPRS de acuerdo a los estándares existentes, además se estudiará acerca de los tipos de sensores que existen y se determinará los apropiados para esta implementación (sensores de temperatura, sensores de humedad, sensores de movimiento). Para implementar el equipo owa33A en el camión se estudiarán los inconvenientes que se puedan presentar al colocarlos en el interior o exterior del transporte y así decidir si se requiere o no algún tipo de protección para el equipo, de igual manera seinvestigará las normas que se deben seguir para el transporte de los diferentes tipos de alimentos perecederos.

La siguiente etapa es el acondicionamiento de señales, se denotará las características técnicas o eléctricas que posee la tarjeta con la que se va a trabajar, es decir, se tomará en cuenta el rango de voltaje máximo de señales que proporcionarán los sensores a implementar. Los datos deberán ser direccionados a la memoria del dispositivo OWASYS y se analizará el diseño de señales tomando en cuenta cuantas entradas y salidas posee el equipo, tanto digitales como analógicas. Se investigará la forma de almacenar en el dispositivo dichas señales y a través de programación se hará un control de las mismas.

La tercera etapa se refiere al software, es decir, como se realizará la transmisión de la información almacenada en el equipo OWASYS a través de un servidor a una PC y a un dispositivo móvil, para realizar así una monitorización utilizando sensores de temperatura, sensores de movimiento. Se trabajará sobre el sistema operativo Linux que es el sistema operativo del

equipo owa33A. Cada proyecto M2M requiere algún grado de conectividad inalámbrica, en algún segmento del camino de la información, al menos para la gran mayoría de los casos. Existen numerosos estándares inalámbricos que pueden ser aplicados a un proyecto de telemetría, por lo tanto se investigará cuál tecnología es la apropiada, revisando algunos aspectos como: tasa de transmisión de datos, alcance de la red móvil, número de usuarios, movilidad, consumo de energía, dirección de la transmisión de señales y calidad de servicio.

El equipo permite aplicaciones SCADA de conectividad M2M, lo que quiere decir que realizará las conexiones por medio de protocolos entre los cuales estaría el SMPP para mensajería y protocolo IPsec para levantar VPN y así mantener comunicación con el servidor, ofreciendo así una seguridad a los usuarios. Los protocolos trabajarán en modo transparente ya que no realizará una mayor modificación en ellos, sino se trabajará de manera que la información del equipo como IP sea reconocida normalmente y sea capaz de conectarse a la red. La SIM card a utilizar es también proporcionada por Telefónica Movistar, es una SIM card con APN específico para soluciones corporativas, que dispone de características de Calidad de Servicio que mantiene un QoS número 10 dentro de la red Movistar por la alta demanda de datos móviles.

El equipo OWASYS integrará funciones de conectividad dedicada desde la Red Fija a la red 2G/GPRS que proporcionará medidas en tiempo real de la monitorización remota del valor de temperatura, aceleración, estado de la puerta, ubicación del camión con cámara de frío por medio del sistema GPS que tiene integrado y visualizar estos datos en la plataforma con acceso a la web diseñada para el equipo OWASYS. Se realizará la programación necesaria para levantar en el ambiente web toda la información sincronizada con el equipo, presentarla en la plataforma y que el usuario pueda acceder a esta en cualquier momento. A su vez se desea programar al equipo para que las alarmas más relevantes sean recibidas mediantes mensajes de texto y también en la plataforma web.

Finalmente, se ejecutará las pruebas de comunicación, envío/recepción de señales y monitorización vía web para asegurar la operatividad adecuada del sistema. Esta gestión de red es la última etapa en donde se hará un análisis de desempeño del sistema para determinar los retardos existentes en la comunicación M2M en un caso real, verificando que se cumpla en tiempo real.

El trabajo servirá de base para el desarrollo de futuras aplicaciones para la integración de una plataforma horizontal M2M que junto a las herramientas de OWASYS permitirán trabajar en proyectos agrícolas, monitoreo del clima, entre otros.

1.4 OBJETIVOS

1.4.1 General

Diseñar e implementar un sistema M2M (Machine to Machine)para la monitorización de cadena de frío en tiempo real en transporte de alimentos utilizando la red GPRS de movistar, con equipos OWASYS y sensores para la transmisión de datos y que el usuario pueda realizar un seguimiento con dispositivos personales.

1.4.2 Específicos

- Presentar el proyecto a ser desarrollado considerando las necesidades de empresas de alimentos con el equipo adquirido en Telefónica.

- Determinar el marco teórico para conocer los conceptos de las redes de telecomunicaciones que se utilizarán en éste proyecto con un sistema M2M (Machine to Machine).

- Conocer y analizar el funcionamiento del equipo con el que se va a trabajar para obtener los datos que se transmitan por medio de la red GSM y que así puedan ser familiarizados con el protocolo de comunicación que maneja el equipo.

- Realizar las instalaciones y pruebas de evaluación requeridas del prototipo de sistema de control de cadena de frío documentando los resultados.

- Mostrar el análisis de costos al utilizar el equipo OWASYS.

1.5 ESTRUCTURA

El capítulo II de este proyecto presenta los fundamentos teóricos de telemetría, su aplicación y equipos de los cuales se encuentra conformado; la definición de GSM y su arquitectura, así como la mensajería corta SMS que cursa por la red. Se hablará también de lo que es GPS, arquitectura y características técnicas; el tipo de programación que se necesita y en qué sistema operativo funciona. Se describirá además de transportes perecederos, acuerdos internacionales que se rigen para el transporte de los diferentes alimentos así como las temperaturas y los tipos de vehículos que los transportan.

En el capítulo III se estudia el tipo de software y hardware que se utilizará para el desarrollo del proyecto. En la parte del software se hablará del tipo de sistema operativo que usa el equipo y del tipo de archivo que necesita para ejecutar la aplicación y que a su vez pueda conectar con la plataforma web gestionada para la aplicación. En la parte del hardware se hablará del equipo, su kit de desarrollo y el esquema general que presentará en la implementación de la aplicación.

En el capítulo IV se encuentra las pruebas y resultados obtenidos del funcionamiento del equipo con respecto a la disponibilidad de servicio, ubicación de sensores y principalmente con el monitoreo y su presentación de datos.

El capítulo V presenta un pequeño análisis financiero del sistema completo implementado y una comparación de la inversión con respecto a la satisfacción de clientes.

Finalmente en el capítulo VI se presenta las conclusiones obtenidas una vez finalizado el trabajo y las recomendaciones para trabajos futuros.

CAPÍTULO 2

FUNDAMENTOS TEÓRICOS

2.1 TELEMETRÍA

2.1.1 DEFINICÓN DE TELEMETRÍA

Telemetría procede de las palabras griegas tele ("lejos") y metron ("medida"), es una medición de magnitudes físicas a distancia. La telemetría es una tecnología que consiste en la adquisición de datos de cualquier índole a distancia, permite la medición remota de magnitudes físicas y el posterior envío de la información hacia el operador del sistema a través de un sistema de telecomunicaciones donde estos datos son administrados, procesados y visualizados; el envío de información también se puede realizar por medio como teléfono, redes de ordenadores, enlace de fibra óptica, etcétera.

2.1.2 APLICACIÓN DE TELEMETRÍA

Las aplicaciones de telemetría se las puede relacionar a varios tipos de sistemas como en la industria espacial, aviones de reconocimiento, plantas químicas, redes de suministro eléctrico, investigación submarina, en agencias espaciales como la NASA para operación de naves espaciales, satélites.

Aplicaciones de telemetría comoen laperforación de pozos petrolíferos para el pulso de lodo y en otros campos ya que facilita la monitorización automática, en lugares donde el ser humano no puede estar presente.

Es bastante útil para el monitoreo de procesos, permitiendo así una visualización en tiempo real, monitoreo de seguridad, alarmas y reportes. Además sirve para obtención de diagnósticos en estudio de tendencias, análisis estadísticos, comparativo de históricos, auditoría de ahorro energético, auditoría en calidad de la iluminación de edificaciones, vigilancia de temperatura de cuartos cerrados e invernaderos de cultivos, medición y detección de velocidad, ubicación y alarmas en vehículos de servicio público, contadores de energía eléctrica inalámbricos, sistemas de detección de desastres naturales, monitoreo de calidad de aire en ciudades, monitoreo de contaminación por ruido, alarmas de seguridad comunitaria, seguridad perimetral, etc.

2.1.3 SENSORES

El sensor es un instrumento de medición, transductor que consta de algún elemento sensible a una magnitud física (Intensidad, Temperatura, Presión, Magnetismo, Humedad) y deben ser capaces por sus propias características o por medio de dispositivos intermedios de transformar esa magnitud de variable física en variables eléctricas. Al diseñar sistemas de adquisición de datos con computadora, se debe tomar en cuenta algunos aspectos de los sensores:

- La influencia de señales de ruido, así como efectos de carga del hardware de adquisición de datos sobre el sensor.

- La calibración del sensor con respecto a la variable física, ya que puede ser lineal o no y una mala calibración provocará mediciones erróneas.

- La precisión del sensor medida de la reproducibilidad del error en las medicionesindica una tolerancia en mediciones sucesivas en idénticas condiciones.

- El tiempo de respuesta del sensor, tiempo necesario para responder a un cambio brusco de la variable sensada.

- El coeficiente de temperatura del sensor, el cual se produce por un cambio en la respuesta del sensor debido al cambio en la temperatura en el que se encuentra.

- La fiabilidad que es la medida de la probabilidad de que el sensor continúe comportándose dentro de los parámetros especificados por el fabricante, referente a la histéresis de funcionamiento, que define la dependencia de la salida del sensor de la anterior respuesta.

2.1.4 Tipos de Sensores

Existen diferentes tipos de sensores dependiendo de las necesidades de las aplicaciones, por ello se pueden escoger:

- Según el tipo de señal de entrada: mecánica, térmica, eléctrica, magnética, radiación, química.

- Según el tipo de señal entregada por el sensor: Analógicos y Digitales. Ejemplos: codificadores de posición, codificadores incrementales,codificadores absolutos, resonadores de cuarzo, galgas acústicas, caudalímetros de vórtices digitales.
- Según la naturaleza de la señal eléctrica generada:Pasivos (de resistencia, capacitancia e inductancia variable) y Activos o

generadores de señal (sensores piezoeléctricos, fotovoltaicos, termoeléctricos, electroquímicos, magnetoeléctricos).

2.1.4.1 Sensores de Temperatura

Existen varias formas de medir la temperatura con sensores de diversas naturalezas, en la tabla 2.1 se muestra los dispositivos capaces de medir temperatura:

Tabla 2.1. Dispositivos de medición de Temperatura

DISPOSITIVOS DE MEDICIÓN DE TEMPERATURA			
Eléctricos	**Mecánicos**	**Radiación Térmica**	**Varios**
Termocuplas Termorresistencias Termistores Diodos Sensores de silicio con efecto resistivo	Sistemas de dilatación Termómetros de vidrio con líquidos Termómetros bimetálicos	Pirómetros de radiación Total (banda ancha) Óptico Pasabanda Relación Termómetros infrarrojos	Indicadores de color (lápices, pinturas) Sondas neumáticas Sensores ultrasónicos Indicadores pirométricos Termómetros acústicos Cristales líquidos Sensores Fluídicos

- **Termocuplas**

Las termocuplas son los sensores de temperatura eléctricos más utilizados en la industria. Una termocupla se hace con 2 alambres de distinto material unidos en un extremo, al aplicar temperatura en la unión de los metales se genera un voltaje muy pequeño en el orden de los milivoltios el cual aumenta con la temperatura. La figura 2.1 es un esquema de ejemplo de una termocupla:

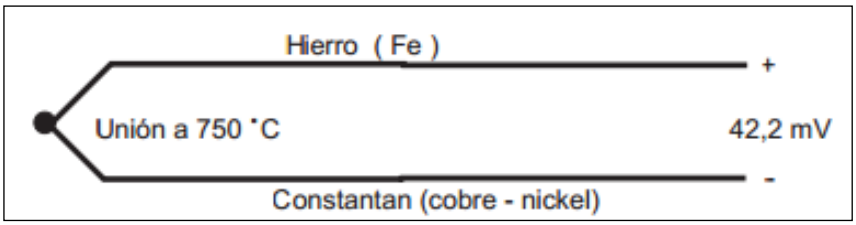


Figura 2.1. Esquema de una termocupla

Normalmente las termocuplas industriales se consiguen encapsuladas dentro de un tubo de acero inoxidable u otro material, en un extremo está la unión y en el otro el terminal eléctrico de los cables protegidos dentro de una caja redonda de aluminio conocida como cabezal.

- **Termistor**

El término termistor proviene de *ThermallySensitive Resistor* y es un sensor resistivo de temperatura que su funcionamiento se basa en la variación de la resistividad que presenta un semiconductor con la temperatura. Son mucho más económicos que las RTD, aunque no son lineales son mucho más sensibles, compuestos de una mezcla sintetizada de óxidos metálicos, el termistor es esencialmente un semiconductor que se comporta como un resistor térmico.

Existen dos tipos de termistor:

- **NTC** (*NegativeTemperatureCoefficient*): coeficiente de temperatura negativo

- **PTC** (*Positive TemperatureCoefficient*): coeficiente de temperatura positivo

Cuando la temperatura aumenta, los tipo PTC aumentan su resistencia y los NTC la disminuyen. En algunos casos la resistencia de un termistor a la

temperatura ambiente puede disminuir en hasta 6% por cada 1 °C de aumento de temperatura. Ésta elevada sensibilidad a variaciones de temperatura hace que el termistor resulte muy adecuado para mediciones precisas de temperatura, es utilizado para aplicaciones de control y compensación en el rango de 150 °C a 450 °C.

- **Termorresistencia (RTD)**

La termorresistencia trabaja según el principio de que en la medida que varía la temperatura, su resistencia se modifica, y la magnitud de esta modificación puede relacionarse con la variación de temperatura. Tienen elementos sensitivos basados en conductores metálicos, que cambian su resistencia eléctrica en función de la temperatura. Este cambio en resistencia se puede medir con un circuito eléctrico, que consiste de un elemento sensitivo, una fuente de tensión auxiliar y un instrumento de medida.

Los dispositivos RTD más comunes están construidos con una resistencia de platino, llamadas también PRTD, es el material más estable y exacto. La relación resistencia vs temperatura correspondiente al alambre de platino es tan reproducible que la termorresistencia de platino se utiliza como estándar internacional de temperatura desde -260 °C hasta 630 °C. También se utilizan otros materiales fundamentalmente como níquel, níquel-hierro, cobre y tungsteno como se muestra en la tabla 2.2 y tienen una resistencia entre 20Ω y 20 kΩ. La ventaja más importante es que son lineales dentro del rango de temperatura entre -200 °C y 850 °C.

Tabla 2.2. Materiales de construcción de RTD

Material	Rango De Temperatura (°C)	Variación coeficiente (%/°C a 25 °C)
Platino	-200 a 850	0.39
Níquel	-80 a 320	0.67
Cobre	-200 a 260	0.38
Níquel-acero	-200 a 260	0.46

Ventajas y Desventajas de los Sensores de temperatura

En la tabla 2.3 presenta una comparación de los sensores de temperatura:

Tabla 2.3. Comparación de los sensores de temperatura

TABLA COMPARATIVA

Sensores	Termorresistencias	Termopares	Termistores
Rango de trabajo	Mide la temperatura de 0 a 850°C con un rango de -50 a + 600°C	-270 a + 3000°C	De -100 a +300 °C
Ventajas	Alto coeficiente de resistencia. Alta resistividad. Relación lineal resistencia-temperatura. Rigidez y ductilidad. Estabilidad de las características durante la vida útil del material. Son más exactos con referencia a los otros sensores de temperatura	Determinación puntual de la temperatura. Respuesta rápida a la variación de temperatura. No necesita alimentación. Rangos de temperaturas grandes. Estabilidad a largo plazo aceptable y fiabilidad elevada.	Son muy pequeños y el tiempo de repuesta depende de la capacidad térmica y de la masa del termistor. La distancia del termistor y el instrumento de medida puedes ser considerable siempre que el elemento posea una alta resistencia comparada con la de los cables de unión.

Continúa ⟶

Desventajas	La medida es indirecta ya que no se mide directamente. Se requiere un circuito de medida para inferir la temperatura partiendo de la resistencia; el circuito habitualmente utilizado es el puente de Wheatstone.	Mantener la unión de referencia a una temperatura constante y conocida. Respuesta no ideal. La temperatura máxima de alcance debe de ser menor al punto de fusión. El medio donde se va a medir no ataca a los metales de la unión. La corriente de alimentación debe ser muy pequeña para despreciar el efecto joule.	La salida de los Termistores se conectan a un circuito de puente de Wheatstone convencional. La corriente que circula a través del circuito debe de ser baja para garantizar que la variación de la resistencia del elemento sea exclusivamente a los cambios de temperaturas del proceso.

2.1.4.2 Acelerómetro

Se denomina acelerómetro a cualquier instrumento destinado a medir aceleraciones; es el tipo de aceleración asociada con el fenómeno de peso, experimentada por una masa de prueba que se encuentra en el marco de referencia del dispositivo. La elección de los sensores para medir aceleración depende de las siguientes características: los márgenes de valores de la aceleración que admite, la capacidad para medir en corriente continua o sólo en alterna, la máxima frecuencia a la que puede trabajar, los parámetros característicos del sensor.

En la siguiente tabla 2.4 se resume las principales características de los distintos tipos de acelerómetros y sus aplicaciones:

Tabla 2.4. Características de tipos de acelerómetros y sus aplicaciones

Tipo	Margen de Medida	Ancho de Banda (Hz)	Ventajas e Inconvenientes	Aplicaciones
MEMS	1.5 a 250g	0.1 a 1500	-Alta sensibilidad -Coste medio -Uso sencillo -Bajas temperaturas	-Impacto -ABS -Airbag -Uso en automoción
Piezoeléctricos	0 a 2000g	10 a 20000	-Sensibilidad media -Uso complejo -Bajas temperaturas -No funcionan en continua	-Vibración -Impacto -Uso industrial
Piezoresistivos	0 a 2000g	0 a 10000	-Respuesta en continua y alterna -Prestaciones medias -Bajo coste -Tamaño y peso Mínimo -Alta sensibilidad	-Vibración -Impacto -Automoción -Biodinámica -Ensayos en vuelo -Test en túneles de Viento
Capacitivos	0 a 1000g	0 a 2000	-Funciona en continua -Bajo ruido -Baja potencia -Excelentes características -Bajo coste	-Uso general -Uso industrial -Sistemas de alarma y seguridad -Mediciones Sísmicas
Mecánicos	0 a 200g	0 a 1000	-Alta precisión en continua -Lentos -Alto coste	-Navegación inercial -Guía de misiles -Herramientas -Nivelación

2.1.5 Cadenas de Frío

La Cadena de Frío se refiere al manejo controlado de las temperaturas y humedad de los productos perecederos para mantener su calidad e inocuidad desde el momento en que sale del campo (cosecha) o punto de

origen a través de toda la cadena de distribución hasta llegar al consumidor final.

Los alimentos perecederos, además de la normativa general relativa al transporte de mercancías, están regulados de forma especial por un acuerdo de transportes internacionales y de vehículos especiales adaptados a este fin. Una reglamentación técnico-sanitaria determina la forma en la que debe realizarse el transporte de alimentos, y otra, las especificaciones que deben cumplir los vehículos especiales para el transporte terrestre a temperatura regulada y los procedimientos de control necesarios para garantizar su seguridad.

Los operadores económicos están igualmente sometidos a normativas y controles sanitarios que pueden ser específicos para el transporte de alimentos o productos concretos; De esta forma, el consumidor tiene la garantía de que los alimentos que llegan a los puntos de venta cumplen con las condiciones higiénicas adecuadas para su consumo, independientemente del origen de los mismos.

2.2 GSM

2.2.1 DEFINICIÓN DE GSM

El Sistema Global para las Comunicaciones Móviles (GSM, proviene de "*GroupSpecial Mobile*") es un sistema estándar, completamente definido, para la comunicación mediante teléfonos móviles que incorporan tecnología digital. Por ser digital cualquier cliente GSM puede conectarse a través de su teléfono con su ordenador y puede enviar y recibir mensajes con e-mail, faxes, navegar por internet, acceso seguro a la red informática de una compañía (LAN/Intranet), así como utilizar otras funciones digitales de transmisión de datos, incluyendo el Servicio de Mensajes Cortos (SMS) o mensajes de texto.

Este estándar es el más extendido en el mundo, con un 82% de los terminales mundiales en uso, con más de 3000 millones de usuarios en 212 países distintos, siendo el estándar predominante en Europa, América del Sur, Asia y Oceanía y con gran extensión en América del Norte.

La ubicación del estándar GSM ha sido una ventaja tanto para consumidores (beneficiados por la capacidad de itinerancia y la facilidad de cambio de operador sin cambiar de terminal, simplemente cambiando la tarjeta SIM) como para operadores de red (que pueden elegir entre múltiples proveedores de sistemas GSM, al ser un estándar abierto que no necesita pago de licencias).

2.2.2 ARQUITECTURA DE LA RED GSM

La demanda por parte de los usuarios de comunicaciones móviles que les admitan a éstos moverse a través de edificios, ciudades o países, ha llevado al desarrollo de nuevas redes de comunicaciones móviles que cumplan ciertas características que se acoplen ciertamente a ser robustas, escalables, fiables entre otras.

El estándar de telefonía celular GSM es totalmente responsable de proporcionar cobertura a través de un territorio particular, llamado región de cobertura o mercado y se ajusta a tener una interconexión la cual se define en una red inalámbrica capaz de proporcionar servicios a los usuarios móviles a través de un país o continente determinando así una arquitectura de la red.

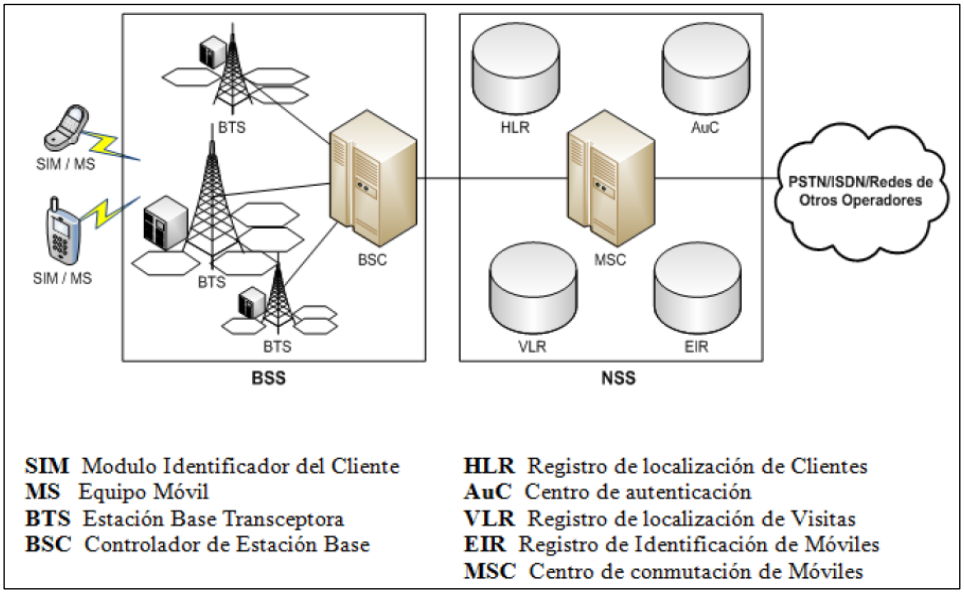


Figura 2.2. Arquitectura de la Red GSM

En la Figura 2.2 se muestra de forma resumida la arquitectura de la red GSM, ésta arquitectura es más compleja y dispone de más elementos que los presentados. El objetivo del proyecto es describir el servicio de mensajes de texto cortos SMS a nivel de aplicación, sin entrar en demasiados detalles de la red subyacente. La arquitectura está constituida por:

1) **Estación Móvil (MS,** *Mobile Station*): Son terminales digitales los cuales pueden ser portables e incluso portátiles incorporan un dispositivo SIM (*SubscriberIdentify Module*) que es básicamente la típica tarjeta que proporciona la información de servicios e identificación en la Red.

2) **Subsistema de Estaciones Base (BSS,** *Base StationSubsystem*): Es una colección de dispositivos que soportan la interface de radio de redes de conmutación. Los principales componentes del BSS son:

   a) ***Estación Transceptora de Base*** **(BTS,***Base TransceiverStation*)
   Consta de módems de radio y el equipo de antenas.

b) ***Controlador de Estación Base*** **(BSC,***Base StationController*) - Gestiona las operaciones de radio de varias BTS y conecta a un único NSS.

3) **Subsistema de red y conmutación (NSS,***Network and SwitchingSubsystem*) Suministra la conmutación entre el subsistema GSM y las redes externas junto con las bases de datos utilizadas para la gestión adicional de la movilidad y de los abonados. Los componentes son:

   a) ***Centro de Conmutación de Servicios Móviles*** **(MSC,** *Mobile ServicesSwitching Center*) Describe al centro de conmutación entre otras funciones.
   b) ***Bases de Datos*****(HLR***Home Location Register,***VLR***Visitor Location Register***, EIR** *Equipment Identity Registry*, **AuC***Authentication Center*).

4) **Centro de mantenimiento y operaciones (OCC,** *Operations and Maintenance Center*) Controla la operación del sistema e inicialización de la red, además de la gestión de los equipos móviles y cobro de cuota.

2.2.3 CARACTERÍSTICAS TÉCNICAS

Las bandas asignadas por la UIT (*International TelecommunicationUnion*) para GSM en la mayoría del mundo son 900 MHz y 1800 MHz. Algunos países en América, incluyendo Ecuador, usan las bandas de 850 MHz y 1900 MHz, partiendo de estos datos las principales características de GSM son:

- **Acceso al medio:** TDMA/FDMA (*Time Division Multiple Access/FrequencyDivisionMultiple Access*).

- **Canales:** 124 canales, cada canal da un servicio de 8 a 16 usuarios a la vez.
- **Bandas:** Usa dos bandas de 25MHz para transmisión y recepción de información.
- **Subida:** Usa la banda de 824-849MHz.
- **Bajada:** Usa la banda de 869-894MHz.
- **Ancho de banda:** 200KHz
- **Modulación:** GMSK (*Gaussian Minimum Shift Keying*).
- **Velocidad máxima del canal de radio:** 270.833Kbps
- **Duración de bit:** 3,692msec.
- **Longitud de trama:** 4,615msec
- **Longitud de slot de tiempo:** 577usec.
- **Codificación de voz:** RPE-LPT 13Kbps (*Regular Pulse Excitation-Long TermPrediction*).
- **Potencia de salida:** de 20mW a 20W.

2.2.4 SERVICIOS

2.2.4.1 Servicios Básicos

Hay dos tipos de servicios que se pueden ofrecer a través de GSM los cuales son la telefonía y los datos.

- Los servicios de telefonía son principalmente servicios de voz que proveen a los suscriptores la capacidad total para comunicarse con otros, incluyendo el equipo terminal necesario y otros servicios además de la telefonía normal, llamadas de emergencia los cuales son DTMF (*Dual-ToneMultifrequency*), Fax, Buzón de Voz y el más importante el cual está estipulado en el proyecto; SMS (*Short MessageService*).

- Los servicios de datos tienen la capacidad necesaria para transmitir datos entre puntos de acceso, creando una interfaz de red.

2.2.4.2 Servicios Suplementarios

GSM soporta un amplio conjunto de servicios suplementarios los cuales complementan a los servicios básicos mencionados anteriormente, a continuación se listan algunos de ellos:

- Marcación abreviada
- Identificación de llamadas.
- Conferencias
- Bloqueo de llamadas entrantes y salientes
- Facturación de servicios y llamadas.
- Transferencia de llamadas.
- Claves de seguridad
- Mantenimiento de llamadas
- Llamadas en espera
- Terminación de llamadas de usuarios ocupados
- Grupos cerrados de usuarios

2.3 MENSAJES CORTOS (SMS)

2.3.1 SERVICIO SMS

El servicio SMS permite trasladar un mensaje de texto entre una Estación Móvil (MS) y otra entidad denominada SME (*Short MessagingEntity*) a través de un centro de servicio SMSC (*Short MessageService Center*) como se muestra en la Figura 2.2. La entidad que recibe el mensaje de texto puede ser otro terminal MS o puede estar situada en una red fija. Cuando se envía un mensaje para solicitar algún tipo de servicio, un extremo es una MS y la otra es un servidor que atiende las

peticiones, el proyecto en marcha se basa en el servicio de SMS mencionado finalmente como muestra la Figura 2.3.

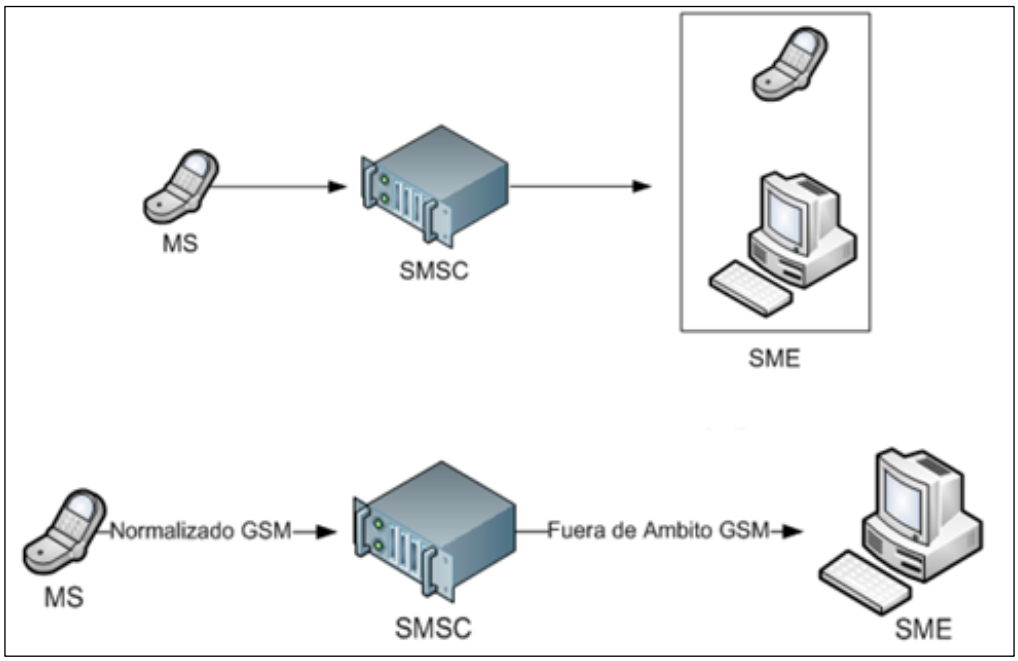


Figura 2.3. Envío de un SMS entre un MS y una entidad Fija/Servidor

El servicio de mensajes de texto SMS se divide en dos servicios básicos los cuales se detallan a continuación y se muestran en la Figura 2.4.

a) **SM-MT** (*Short Message Mobile Terminated Point-to-Point*).Es un servicio de entrega de mensaje que transita desde el SMSC hasta un MS, obteniendo de dicha transacción un reporte de lo ocurrido.

b) **SM-MO** (*Short Message Mobile Originated Point-to-Point*) Es un servicio de envío de mensaje que transita desde el MS hasta el SMSC, obteniendo de dicha transacción un reporte de lo ocurrido.

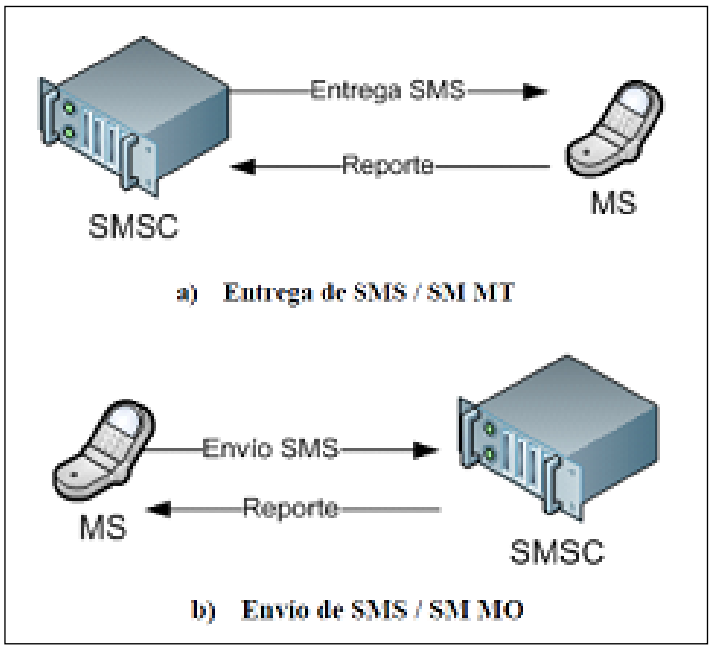


Figura 2.4. Servicios básicos: SM-MT / SM-MO

2.3.2 FORMATOS DEL SMS

Dentro de las especificaciones de mensajes cortos SMS existe la posibilidad de hacer el envío de mensajes de dos maneras:

a) **Modo PDU.-** Es aquella estructura de mensaje que lleva consigo bits de información específica además de funciones de control para la presentación del mensaje.

b) **Modo Texto.-** Es aquella estructura de mensaje que está conformada por caracteres de texto, números y símbolos, cabe recalcar que es un modo de gama media o alta codificación que no se encuentra en todos los terminales.

2.3.3 ARQUITECTURA DE LA RED SMS

Para la estructura básica de la red SMS se presentan varias entidades las cuales se explican de manera vertiginosa y se hacen presentes en la Figura 2.5.

- **MS:** Estación Móvil.
- **MSC:** Centro de Conmutación
- **SMS-GMSC** (*Gateway MSC forShort MessageService*): Es una puerta de enlace para que el MSC resuelva el servicio del SMS.
- **SMS-IWMSC** (*Internetworking MSC for Short MessageService*): Puerta para hacer la interconexión entre el MSC para que resuelva el servicio SMS.
- **SMSC:** Centro de servicios
- **HLR:** Base de datos para registro de localización de clientes.
- **VLR:** Base de datos para registro de localización de visitas.

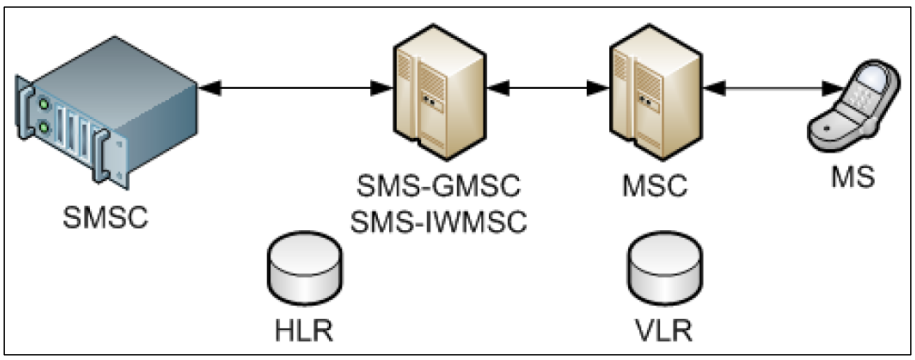


Figura 2.5. Arquitectura de la Red SMS

2.3.3.1 LA TARJETA SIM

Las siglas SIM *SubscriberIdentity Module* o Módulo de Identidad del Subscriptor, la ventaja es que proporcionan movilidad al usuario ya que puede cambiar de teléfono y conservar el mismo número, sin la tarjeta SIM el terminal no funciona al no acceder a la red.

Para evitar inferencia se toman medidas de seguridad, como el encriptado digital del enlace radio para asegurar la privacidad de las conversaciones, y la comprobación, autentificación de las llamadas, mediante el empleo de una tarjeta inteligente de identificación de usuario SIM.

Los terminales móviles sólo funcionan sí disponen de una tarjeta SIM que proporciona una clave de autentificación secreta y un algoritmo codificado, para realizar una función de seguridad que identifique al usuario que vaya a efectuar la llamada. Para esta función, la tarjeta SIM almacena tres tipos diferentes de información relacionada con el cliente:

- Los datos obtenidos durante la fase de tramitación del alta en el servicio, como la clave de autentificación del usuario o la de acceso.

- Los datos temporales de la red: la identidad temporal del usuario, el de área de localización, las redes a las que no se tiene acceso, etc.

- Los datos relacionados con el servicio, como la función de notificación del precio de la llamada.

Existen dos tipos de tarjetas SIM: ID-1, con un formato que cumple las normas ISO y que son de igual tamaño que las tarjetas de crédito, y la conectable, más pequeña que la anterior. Las dos tienen igual constitución electrónica y lógica.En cuanto a seguridad, la tarjeta SIM contiene los siguientes algoritmos y claves:

- Algoritmo de autentificación.
- Algoritmo de generación de la clave de cifrado.
- Clave de cifrado.
- Clave de autentificación de usuario.

La tarjeta SIM en la figura 2.6posee un número de identificación personal PIN (en el caso de GSM) que el usuario deberá introducir cada vez que conecte su teléfono móvil. También cuenta con una clave de desbloqueo PUK para desbloquear la tarjeta en caso de que esté bloqueada, como consecuencia de haber introducido varios errores consecutivos.

Figura 2.6. Tarjeta SIM

2.4 SISTEMA DE POSICIONAMIENTO GLOBAL GPS

2.4.1 DEFINICÓN DE GPS

El GPS (*Global PositioningSystem*; Sistema de Posicionamiento Global) es un sistema global de navegación por satélite que permite determinar en todo el mundo la posición de un objeto, una persona, un vehículo o una nave, con una precisión hasta de centímetros. Aunque su invención se atribuye al gobierno francés y belga, el sistema fue desarrollado, instalado y actualmente operado por el Departamento de Defensa de los Estados Unidos.

2.4.2 ARQUITECTURA DEL SISTEMA GPS

El sistema se descompone en tres segmentos básicos, los dos primeros de responsabilidad militar: segmento espacio, formado por 24 satélites GPS con una órbita de 26560 Km de radio y un periodo de 12 horas; segmento control, que consta de cinco estaciones monitoras encargadas de mantener en órbita los satélites y supervisar su correcto funcionamiento, tres antenas terrestres que envían a los satélites las señales

que deben transmitir y una estación experta de supervisión de todas las operaciones; y segmento usuario, formado por las antenas y los receptores pasivos situados en tierra.

Los receptores, a partir de los mensajes que provienen de cada satélite visible, calculan distancias y proporcionan una estimación de posición y tiempo.

2.4.3 CARACTERÍSTICAS TÉCNICAS DE GPS

Las características establecidas para GPS, están dadas dependiendo de los parámetros que lo componen, los cuales son:

1. Satélites en la constelación: 24 (4 × 6 órbitas)
    - Altitud: 20200 km
    - Período: 11 h 58 min (12 horas sidéreas)
    - Inclinación: 55 grados (respecto al ecuador terrestre).
    - Vida útil: 7,5 años

2. Segmento de control (estaciones terrestres)
    - Estación principal: 1
    - Antena de tierra: 4
    - Estación monitora (de seguimiento): 5, Colorado Springs, Hawai, Kwajalein, Isla de Ascensión e Isla de Diego García

3. Señal RF
    - Frecuencia portadora.
    - Civil – 1575,42 MHz (L1). Utiliza el Código de Adquisición Aproximativa (C/A).

- Militar – 1227,60 MHz (L2). Utiliza el Código de Precisión (P), cifrado.
- Nivel de potencia de la señal: –160 dBW (en superficie tierra).
- Polarización: circular dextrógira.

4. Exactitud

   - Posición: oficialmente indican aproximadamente 15 m (en el 95% del tiempo). En la realidad un GPS portátil monofrecuencia de 12 canales paralelos ofrece una precisión de 2,5 a 3 metros en más del 95% del tiempo. Con el WAAS / EGNOS / MSAS activado, la precisión asciende de 1 a 2 metros.
   - Hora: 1 ns

5. Cobertura: mundial

6. Capacidad de usuarios: ilimitada

7. Sistema de coordenadas:

   - Sistema Geodésico Mundial 1984 (WGS84).
   - Centrado en la Tierra, fijo.

8. Integridad: tiempo de notificación de 15 minutos o mayor. No es suficiente para la aviación civil.

9. Disponibilidad: 24 satélites y 21 satélites. No es suficiente como medio primario de navegación.

2.4.4 FUNCIONAMIENTO

El GPS funciona mediante una red de 27 satélites (24 operativos y 3 de respaldo) en órbita sobre el globo, a 20200 km, con trayectorias sincronizadas para cubrir toda la superficie de la Tierra. Cuando se desea determinar la posición, el receptor que se utiliza para ello localiza automáticamente como mínimo tres satélites de la red como se muestra en la figura 2.7, de los que recibe las señales indicando la posición y el reloj de cada uno de ellos. Con base en estas señales, el aparato sincroniza el reloj del GPS y calcula el retraso de las señales (es decir, la distancia al satélite). Por "triangulación" calcula la posición en que este se encuentra.

En el caso del GPS, la triangulación se basa en determinar la distancia de cada satélite respecto al punto de medición. Conocidas las distancias, se determina fácilmente la propia posición relativa respecto a los tres satélites. Conociendo además las coordenadas o posición de cada uno de ellos por la señal que emiten, se obtiene la posición absoluta o las coordenadas reales del punto de medición.

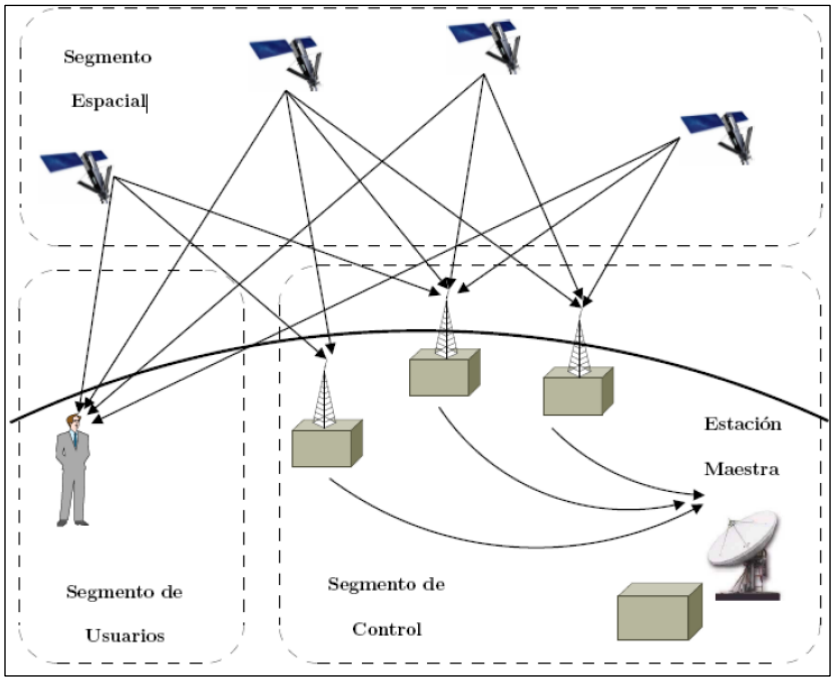


Figura 2.7. Funcionamiento del Sistema GPS

Cada uno de estos satélites da la vuelta al mundo dos veces al día, transmitiendo constantemente su identificación personal, trayectoria, posición respecto a un eje de coordenadas situado en el centro de la Tierra y tiempo.

Por otro lado, se encuentra el denominado segmento de control, esto es, una estación maestra situada en Colorado Springs (EE.UU.) enlazada con una serie de estaciones monitoreadas, cada una de las cuales sigue a los satélites que se encuentran a su vista, transmitiendo la información entre estos satélites y la estación maestra.

El propósito del intercambio de información no es otro que el de permitir al satélite transmitir una señal que es cuidadosamente cronometrada. Finalmente se encuentra el segmento de usuarios, donde un observador mediante un receptor GPS puede determinar su posición y tiempo resolviendo un sistema de cuatro ecuaciones con cuatro incógnitas; observando las señales de cuatro satélites.

La transmisión de la información se realiza mediante la técnica de Spread Spectrum, todos los satélites transmiten en la misma frecuencia, pero cada uno utiliza un código particular que lo identifica. Un receptor extrae la señal de uno en particular mediante un decorrelator. De esta manera, la posición se determina en base a la medición de las distancias desde el observador a los satélites, considerando el tiempo de propagación de la señal.

- **Niveles de Servicio GPS**

El sistema GPS proporciona dos niveles diferentes de servicio que separan el uso civil del militar:

a) **Servicio de Posicionamiento Estándar (SPS**, *Standard PositioningService*). Precisión normal de posicionamiento civil obtenida con la utilización del código C/A (*Coarse/Acquisition*) de frecuencia simple.

b) **Servicio de Posicionamiento Preciso (PPS**, *Precise PositioningService).* Este posicionamiento dinámico es el de mayor precisión.

Este sistema logra acoplarse a la salida de una alarma ya conectada. Es un mecanismo en el que la señal de alarma se remite en forma de mensaje de texto o como datos a un servidor. En estos mensajes se va a encontrar la dirección en la cual se encuentra su automóvil.

Este sistema es tan estricto como los de uso militar (en el peor de los casos, el error puede oscilar entre los 4 y 5 metros), es gratuito, no tiene límite de distancia mientras se encuentre en una región de cobertura GSM y se pueden aplicar a cualquier tipo de vehículo de vía terrestre o marítima.

- **Sistema de Localización de Flotas y Vehículos**

Por medio de la localización GPS representada en la figura 2.8 el cliente puede realizar una comprobación de la ruta de los vehículos así como el tiempo que emplea en dichas rutas, permite saber el número de paradas y tiempo empleado en cada una. El localizador GPS puede activar alarmas por salidas de ruta, por aumento de la temperatura en caso de ser camión frigorífico, etc. Las alarmas de última generación tienen funciones más complejas y son bastante sofisticadas.

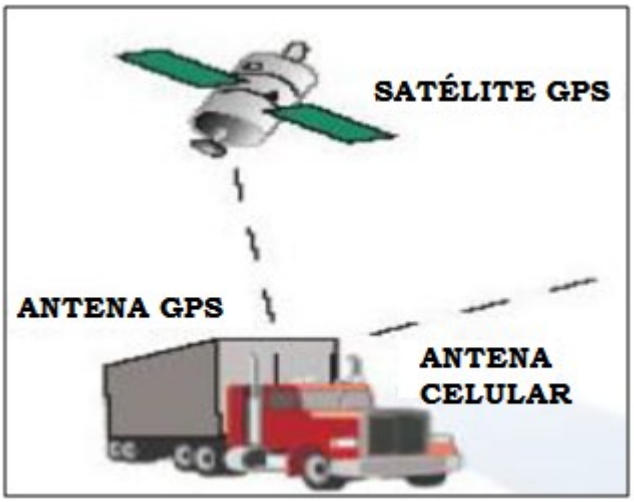


Figura 2.8. Sistema de Rastreo Satelital GPS

2.5 TRANSPORTE DE PRODUCTOS PERECEDEROS

2.5.1 Acuerdo sobre Transporte Internacional de Mercancías Perecederas (ATP)

El acuerdo internacional ATP establece las normas que garantizan el transporte de alimentos en condiciones óptimas para su consumo, la que fue aprobada en septiembre de 1970. El objetivo del ATP es asegurar que las mercancías perecederas sean transportadas en el ámbito internacional de modo que se garanticen las condiciones óptimas para su consumo, asegurando, del mismo modo, que los vehículos que realicen este transporte satisfagan las condiciones técnicas regidas por el propio acuerdo.

Desde su aprobación ha servido de referencia para el desarrollo posterior de la legislación del transporte de mercancías perecederas. Las definiciones y normas que contiene se aplican a todo transporte de mercancías perecederas, tanto destinado a terceros o como mercancía propia, efectuado exclusivamente por ferrocarril, por carretera o por una combinación de ambos métodos, cuando el lugar de carga y de descarga de la mercancía se encuentre en estados diferentes y cuando el lugar de

descarga de la mercancía esté ubicado en el territorio de una de las partes contratantes.

2.5.2 Las mercancías perecederas y sus temperaturas

El ATP establece un listado de las mercancías que han de considerarse perecederas a los efectos de la aplicación del acuerdo. Los operadores económicos que transporten mercancías perecederas deben utilizar vehículos isotermos, refrigerantes, frigoríficos o caloríficos, salvo que las temperaturas previsibles durante el transporte conviertan a esta obligación en no aplicable para el mantenimiento de las condiciones de unas temperaturas fijas que se establecen con relación a los productos listados. Los principales se detallan a continuación:

- Productos ultracongelados y congelados (crema congelada, -20°C; pescados, productos preparados a base de pescado, moluscos y crustáceos congelados o ultracongelados y cualquier producto ultracongelado, -18°C; cualquier producto congelado, -12°C; mantequilla congelada, -10°C).

- Mantequilla: 6 °C.

- Productos de caza: 4 °C.

- Leche en cisterna (cruda o pasteurizada) para al consumo inmediato: 4°C.

- Leche industrial: 6°C.

- Productos lácteos (yogurt, crema, nata y queso fresco): 4°C.

- Pescado, moluscos y crustáceos (con exclusión del pesado ahumado, salado seco o vivo, los moluscos vivos y crustáceos vivos): deberán envasarse siempre en hielo fundente.

- Productos preparados a base de carne (de los que se excluyen los que se han estado estabilizado por salazón, ahumado, secado o esterilización): 6°C.

- Carne (exceptuados los despojos rojos): 7°C.

- Ave de corral y conejos: 4°C.

2.5.3 Tipos de vehículos para cada producto

No todos los vehículos son apropiados para el transporte de mercancías perecederas a fin de mantener la temperatura establecida legalmente para conservar el alimento en condiciones inocuas y aptas para su consumo. La norma define la siguiente tipología de vehículos de transporte:

- *Vehículo isotermo:* vehículo cuya caja está construida con paredes aislantes, incluidos las puertas, el suelo y el techo, que limita el intercambio de calor entre el interior y el exterior.

- *Vehículo refrigerado:* vehículo isotermo que, gracias a una fuente de frío, permite reducir la temperatura del interior de la caja vacía, y de mantenerla después para una temperatura exterior media de 30°C a -20°C como máximo, según la clase de vehículos refrigerados que se establecen.

- *Vehículo frigorífico:* vehículo isotermo que incorpora un dispositivo de producción de frío, y permite, con una temperatura media exterior de 30°C, reducir la temperatura del interior de la caja vacía y de mantenerla de forma permanente entre 12°C y -20°C, dependiendo de la clase de vehículo para esta categoría.

- *Vehículo calorífico:* vehículo isotermo provisto de un dispositivo de producción de calor que permite elevar la temperatura en el interior de la caja vacía y mantenerla después durante doce horas, por lo menos, sin repostado a un valor prácticamente constante y no inferior a 12°C.

2.6 SISTEMAS M2M

2.6.1 DEFINICIÓN

Cuando se habla de tecnología M2M, genera un concepto que engloba la automatización de los procesos de comunicación entre máquinas (*Machine to Machine*), entre dispositivos móviles y máquinas (*Mobile to Machine*), y entre hombres y máquinas (*Manto Machine*).

Estas máquinas pueden ser desde diminutos dispositivos electrónicos como equipos personales para comunicación o entretenimiento, equipos de medición (sensores) o de control (actuadores), etiquetas electrónicas inteligentes, microprocesadores presentes en los artefactos de su hogar, automóvil u oficina, hasta computadores personales o complejos servidores en un centro de procesamiento de datos informático.

Con la tecnología M2M podrá hacer posible la comunicación de sus equipos en terreno con sus centros de información, con otros equipos en distintos lugares, y con personas, a través de dispositivos de comunicación personal, de forma organizada, y en tiempo real.

En este intercambio de información entre máquinas, ya sea mediante redes fijas o móviles, la característica principal es que no necesita intervención humana; su finalidad es el control y supervisión de procesos donde los sistemas han sido configurados para responder automáticamente a las señales que reciben, lo que los hace mucho más rápidos ante

situaciones críticas que cuando existe la intervención humana, lo que evita desastres mayores.

2.6.2 APLICACIONES DE M2M

Las aplicaciones prácticas de M2M son tan numerosas que nadie puede cuestionar su utilidad, tanto en el segmento de consumo como en el corporativo. La telemática en el espacio de la automoción, la seguridad, la automatización de los sistemas del hogar, el ámbito de los servicios y mantenimiento industrial, así como prestaciones de pago y de telemedicina son sus principales campos de aplicación.

Los ejemplos son tan amplios como se puede imaginar: desde aplicaciones para que los vehículos avisen en caso de accidentes, hasta alarmas para el hogar que alertan a los usuarios a su móvil cuando alguien desconocido ha entrado en su casa, pasando por aplicaciones para mantenimiento de los contadores de la luz y el gas, servicios para que las máquinas expendedoras, por ejemplo de bebidas, helados, avisen a sus suministradores cuando sus productos se están acabando o aplicaciones en los parquímetros para que comuniquen a los usuarios cuando se acaba su tiempo de aparcamiento. Las aplicaciones son infinitas y lo serán aún más cuando los sistemas de telecomunicaciones mejoren con la llegada definitiva de UMTS para dispositivos móviles.

2.6.3 TECNOLOGÍA M2M

Las comunicaciones máquina a máquina no son nada nuevo, pues se vienen utilizando desde hace mucho tiempo,por ejemplo para el telecontrol de energía, automatismos diversos, control de paso de personas y vehículos, medida de la contaminación ambiental y acústica, estaciones meteorológicas, paneles de información, etc. Lo novedoso es la aplicación de la tecnología móvil a este campo, pues abre un inmenso mundo de posibilidades, y de negocio, hasta ahora limitado por la falta de movilidad y la

necesidad de disponer de una conexión fija a la red en el punto en el que se quiera ubicar la máquina en cuestión, lo que muchas veces no es viable por su elevado coste o por razones meramente técnicas.

Dejando a un lado las conexiones fijas, dadas sus limitaciones, las conexiones por radio o inalámbricas pueden emplear tecnologías muy diferentes, dependiendo del uso, el alcance, la ubicación y el coste de la conexión. Las más comunes, hoy en día, para la conexión entre máquinas son la tecnología móvil celular (medio y largo alcance), en la que todos los fabricantes están trabajando intensamente; la tecnología WLAN Wi-Fi (medio y corto alcance) y Bluetooth (corto alcance).

- **Tecnología Móvil Celular**

La tecnología móvil celular combinada con la aplicación de la conmutación de paquetes (introducida con GPRS, cdma2000 y UMTS) resulta ideal para su aplicación en la comunicación M2M, sobre todo si las máquinas necesitan intercambiar información a ráfagas, esporádicamente y mucha cantidad.

Las propias redes GSM o TDMA (conmutación de circuitos) también permiten muchas aplicaciones, pero se requiere el establecimiento de la comunicación, lo que conlleva un cierto tiempo, incluso que no se consiga si la red está saturada en ese momento. Además, presentan el inconveniente de su baja velocidad, lo que puede resultar inapropiado para ciertas aplicaciones que requieren el envío masivo de datos. Otra de sus limitaciones es que el coste de la comunicación viene dado por el tiempo, lo que puede resultar demasiado caro para ciertos propósitos.

- **Tecnología Wi-Fi**

La tecnología radio utilizada en las redes locales inalámbricas (WLAN), de las que Wi-Fi es un claro exponente, empieza a encontrar su hueco en el mercado de la comunicación M2M. Como estos sistemas, que utilizan la banda ISM de 2,4 GHz o de 5 GHz, tienen un alcance limitado a unos cuantos cientos de metros, es necesario que el punto de acceso, a su vez, esté conectado a una red más amplia, por cable o por radio (celular), para alcanzar su destino final. Sus principales ventajas, sin embargo, residen en su alta velocidad y en que al ser la banda de uso libre, no precisan licencia.

- **Tecnología Bluetooth**

La tecnología radio Bluetooth, que ya prácticamente incorporan todos los teléfonos móviles, tiene el inconveniente de su poco alcance (lo que es a propósito para evitar interferencias con otros dispositivos y limitar el consumo de energía), que sólo permite alcanzar unos 10 metros. Por esta razón, sus aplicaciones son muy específicas: para conectar periféricos al ordenador, comunicación entre teléfonos móviles y sincronización de PC, PDA y móviles, o como mando a distancia, cascos inalámbricos, etc.

La banda de frecuencias que emplea es la misma que Wi-Fi, la de 2,4 GHz, que no precisa licencia, y la tasa de transmisión de datos queda limitada a 1Mbit/s como máximo.

2.6.4 ESTRUCTURA DE APLICACIÓN M2M

En toda conexión M2M, generalmente bidireccional, la información se intercambia entre un dispositivo remoto y un dispositivo de control como se presenta en la figura 2.9. Dicha comunicación puede realizarse en ambos sentidos:

a) **Dispositivo a centro de control:** Para recopilar toda la información y procesarla de acuerdo a las necesidades.

b) **Centro de control a dispositivo:** Para enviar instrucciones a los dispositivos remotos.

Figura 2.9. Estructura de aplicaciones M2M

La tecnología M2M involucra a varios dispositivos inalámbricos ubicados en lugares remotos con la necesidad de enviar y recibir información de un centro de control ubicado en la infraestructura de red.

El gran inconveniente con las tecnologías M2M es que puede haber una amplia gama de soluciones dependiendo de la tecnología inalámbrica usada, el tipo de dispositivos o la tecnología de red, lo que hace muy difícil su estandarización.

- **Módulos para control inalámbrico**

El resto de tecnologías utilizadas en las comunicaciones M2M también son inalámbricas, pero se llamará inalámbricas a las comunicaciones por radio frecuencia.En el mercado existen módulos de comunicación punto-

punto, con la configuración llamada espejo, donde las salidas del módulo receptor reflejan el estado de las entradas del emisor y viceversa como se presenta en la figura 2.10.

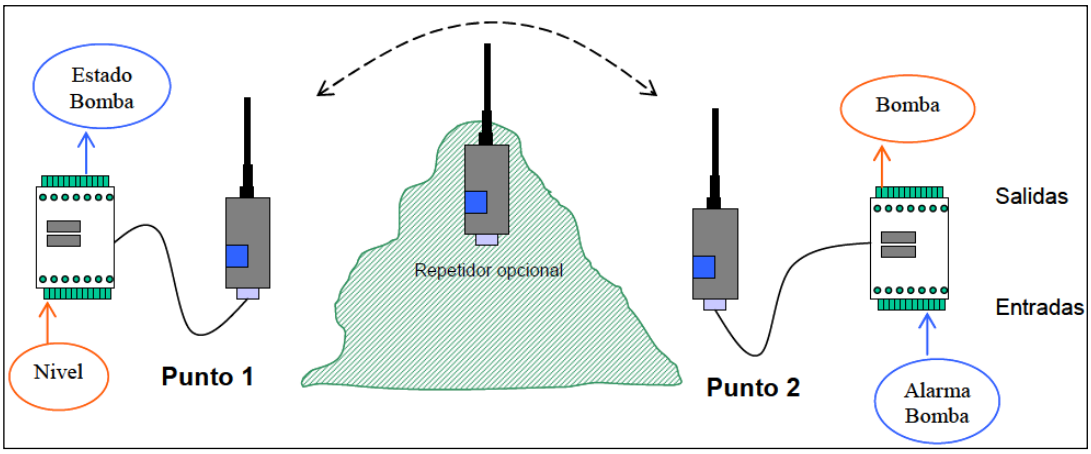


Figura 2.10. Comunicación de tecnología M2M

La gran ventaja de estos módulos es que la comunicación inalámbrica es totalmente transparente para el usuario.Además, estos módulos son capaces de cubrir grandes distancias de hasta 90 km sin necesidad de usar repetidores adicionales.

- **Tecnología utilizada**

El acceso al canal utilizado en GPRS se basa en divisiones de frecuencia sobre un dúplex y TDMA. Durante la conexión, al usuario se le asigna un canal físico, formado por un bloque temporal en una portadora concreta. Ese canal será de subida o bajada dependiendo de si el usuario va a recibir o enviar datos. Esto se combina con la multiplexación estadística en el dominio del tiempo, permitiendo a varios usuarios compartir el mismo canal físico, ya sea de subida o de bajada.

Los paquetes tienen longitud constante, correspondiente a la ranura de tiempo del GSM. El canal de bajada utiliza una cola FIFO para los paquetes

en espera, mientras que el canal de subida utiliza un esquema similar al de ALOHA con reserva.

Que la conmutación sea por paquetes permite fundamentalmente la compartición de los recursos de radio. Un usuario GPRS sólo usará la red cuando envíe o reciba un paquete de información. Todo el tiempo que esté inactivo podrá ser utilizado por otros usuarios para enviar y recibir información. Esto permite a los operadores dotar de más de un canal de comunicación sin miedo a saturar la red, de forma que en GSM sólo se ocupa un canal de recepción de datos del terminal a la red y otro canal de transmisión de datos desde la red al terminal, en GPRS es posible tener terminales que gestionen cuatro canales simultáneos de recepción y dos de transmisión.

- **Servicios ofrecidos**

La tecnología GPRS mejora y actualiza a GSM con los servicios siguientes:

- ✓ Servicio de mensajes multimedia (MMS)
- ✓ Mensajería instantánea
- ✓ Aplicaciones en red para dispositivos a través del protocolo WAP
- ✓ Servicios P2P utilizando el protocolo IP
- ✓ Servicio de mensajes cortos (SMS)
- ✓ Posibilidad de utilizar el dispositivo como módem USB

- **Clases de dispositivos**

Existen tres clases de dispositivos móviles teniendo en cuenta la posibilidad de usar servicios GSM y GPRS simultáneamente:

a) Clase A

Estos dispositivos pueden utilizar simultáneamente servicios GPRS y GSM.

b) Clase B

Sólo pueden estar conectados a uno de los dos servicios en cada momento. Mientras se utiliza un servicio GSM (llamadas de voz o SMS), se suspende el servicio GPRS, que se reinicia automáticamente cuando finaliza el servicio GSM. La mayoría de los teléfonos móviles son de este tipo.

c) Clase C

Se conectan alternativamente a uno u otro servicio. El cambio entre GSM y GPRS debe realizarse de forma manual.

Para que un dispositivo de clase A pueda transmitir en dos frecuencias a la vez, necesitaría dos radios. Para resolver este costoso problema, un móvil con GPRS suele implementar la característica conocida como modo de transferencia dual (*dual transfer mode*, DMT). Un móvil DMT puede usar a la vez el canal de datos y el de voz, puesto que es la red la que coordina y se asegura de que no se requiera transmitir en dos frecuencias diferentes a la vez. Los móviles DMT se consideran de clase A, pero simplificados.

CAPÍTULO 3

DISEÑO E IMPLEMENTACIÓN

3.1 ANÁLISIS Y SELECCIÓN DEL EQUIPO

Para la gestión óptima de camiones de refrigeración móvil, se necesita usar un avanzado Sistema de Tecnología de Punta, para lo cual se debe contar con equipos que permitan cumplir los objetivos de este proyecto, aprovechando de alguna manera las ventajas de la tecnología hoy en día.

Dentro del Sistema de Tecnología de Punta, la parte más importante es el equipo que va a ser usado ya que debe ser el adecuado para el proyecto. El equipo deberá ser un equipo compacto y de un tamaño reducido, esto es por la facilidad de ubicarlo fácilmente en un vehículo con lo que se refiere al hardware.Hablando de la parte del software o la parte lógica, es importante que el equipo sea multifuncional, para evitar de esta manera utilizar diferentes equipos que vendrían a cumplir la misma función en conjunto.

Partiendo de las necesidades se pudo ver la utilización de equipos marca OWASYS, entre los cuales el más adecuado y seleccionado para el proyecto es el owa33A, es un módulo de fabricación Española, el mismo que cuenta con un módulo transceptor de datos GPRS, antena GPS, antena para GSM y GPRS, entradas y salidas analógicas y digitales. Una parte muy

importante del dispositivo es su reducido tamaño y la facilidad de instalación en cualquier ambiente y que es muy factible para la monitorización del vehículo en forma remota.

Se debe aprovechar al máximo los servicios que ofrece el módulo. A través de la antena GPS, el módulo captará datos de posición y velocidad, los mismos que se almacenan en una memoria temporal, para luego ser transmitidos a través de la red de telefonía celular mediante la antena GPRS, estos datos llegarán hacia una estación de monitoreo, donde se lleva el registro de la posición geográfica del móvil y el estado que presenta.

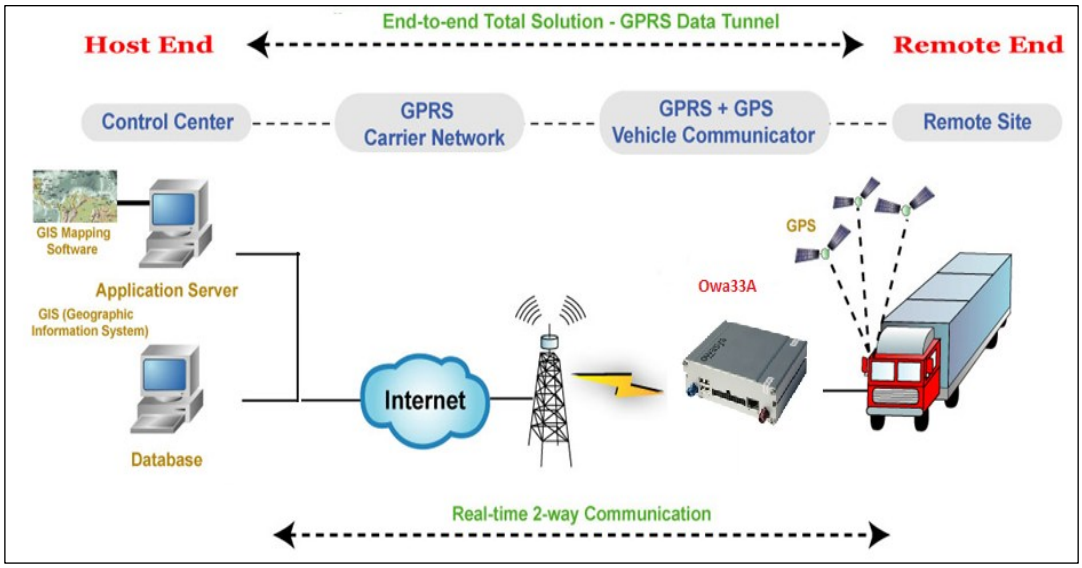


Figura 3.1. Esquema del funcionamiento del módulo elegido

En la figura 3.1 se puede apreciar como el equipo en un vehículo, tiene una conexión punto a punto con el servidor, en el cual existen diferentes módulos pero que son básicos para la formación del servidor.

3.2 CARACTERÍSTICAS DEL EQUIPO owa33A

3.2.1 MÓDULO TRANSEPTOR DE DATOS

El Módulo transceptor de Datos es una de las partes más importantes del equipo, ya que a través de este módulo se realiza el envío y recepción de datos captados por los módulos del equipo y enviados a través de la red GPRS, hacia una Central. Una ventaja es que puede trabajar de modo dúplex por lo cual tiene la capacidad de enviar y recibir datos al mismo tiempo.

La ventaja de trabajar con equipos para trasmisión de datos que se basan en Tecnologías de Segunda Generación como GSM/GPRS, es que a pesar de cambiar de región o de país este puede seguir funcionando debido al módulo transceptor de datos que cuenta con una banda cuádruple para cobertura internacional (850/900/1800/1900) y que funciona en conjunto con Roaming para la conexión.

Los equipos para funciones M2M son especializados principalmente para que la transmisión de datos sea garantizada y esto lo hace mediante un Registro de Almacenamiento de Datos Integrado, cuya función es guardar los datos de forma temporal, hasta el instante que el equipo realice de nuevo la conexión con la red de telefonía celular y transmita los datos, de esta manera no existe pérdida de datos hasta el usuario.

3.2.2 ESPECIFICACIONES DEL EQUIPO owa33A

Las especificaciones técnicas del equipo, se pueden apreciar en la Tabla 3.1

Tabla 3.1. Especificaciones Técnicas del Equipo owa33A

ESPECIFICACIONES TÉCNICAS DEL EQUIPO		
Especificaciones Generales	Procesador	ARM9/ 400 MHz
	Sistema Operativo	Linux OS 2.6.36
	Voltaje	DC 7 V – 48 V
	Flash	32 Mbyte
	Ram	32 Mbyte / 64 Mbyte
	Expansión dememoria	MicroSD card
Especificaciones GSM/GPRS	Bandas de Funcionamiento	GSM850, EGSM900, GSM1800, GSM1900
	Consumo de Potencia	Clase 4 (2W) para GSM850, EGSM900
	Estación móvil	Clase 1 (1W) para GSM1800, GSM1900
	GPRS Multi-Slot	Clase B
	Protocolo	Clase 10 (4+2)
		Punto a Punto (PPP)
Especificaciones GPS	Receptor	Frecuencia GPS L1, 50 canales≤ 4 Hz
	Tasa de actualización	
	Precisión	2.5 metros CEP
	Adquisición de la señal	Inicio en frío : 32 seg
	Readquisición de señal	Inicio intermedio: 32 seg
		Inicio en caliente: < 1 seg
Interfaces	10 Entradas/Salidas Digitales	40 V máximo para entradas
	4 Entradas Analógicas	Las entradas como contadores (odómetro)
	Cobertor removible	Pequeños circuitos de protección para salidas
	3 puertos externos RS232	

Continúa ⟶

	1 Puerto Ethernet 10/100BaseT 4 leds indicadores Audio Puerto CAN Acelerómetro	Precisión de 1% Comparte 4 de los pines de entrada/salida Para batería, SIM y MicroSD Para auriculares y micrófono externo Bus que soporta alta velocidad 1 Mbps CAN 2.0 B Programable a 3 ejes
Especificaciones Técnicas	Temperatura de Operación Dimensiones (L x E x A) (mm) Peso (g) Estructura protectora Protección al medio Conector externo de Antena Batería	-40 °C a 85 °C 110 x 85 x 40 mm 270 g Aluminio IP30 GPRS, GPS Li-Ion recargable 3.7 V, 1800 mA

3.2.3 ACCESORIOS DEL EQUIPO owa33A

El equipo cuenta con los siguientes accesorios:

1. Módulo owa33A
2. Antenas GPS
3. Antena GSM/GPRS
4. Kit de Desarrollo (Desarrollo y programación del equipo)
5. Batería de respaldo

3.3 DESCRIPCIÓN DEL HARDWARE

3.3.1 Características Externas del Hardware

El equipo presenta las siguientes características externas:

3.3.1.1 Panel Frontal

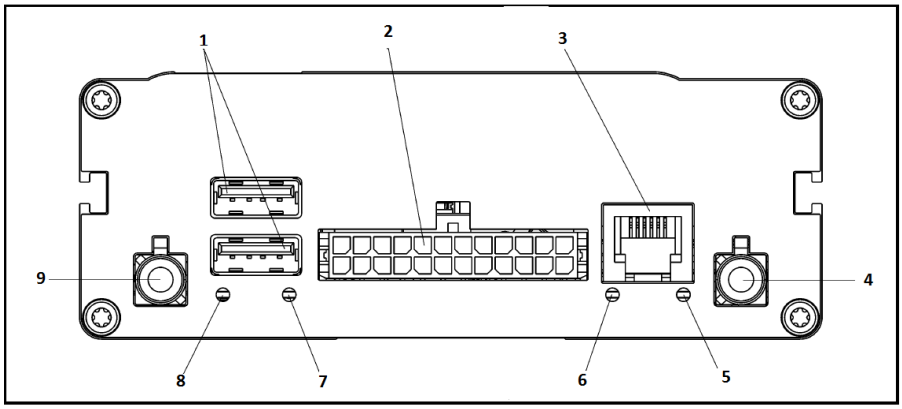


Figura 3.2. Panel Frontal del equipo owa33A

1. Conectores USB.
2. Conector Machine (BUS) para uso de entradas y salidas analógicas y digitales.
3. Conector Rj11 para audio.
4. Conector de la Antena GSM (SMA o FAKRA).
5. Led indicador de radio (color amarillo).
6. Led indicador del estado (color rojo).
7. Led indicador de alimentación (color verde).
8. Led indicador de GPS (color naranja).
9. Conector de Antena GPS (SMA o FAKRA).

3.3.1.2 Panel Posterior

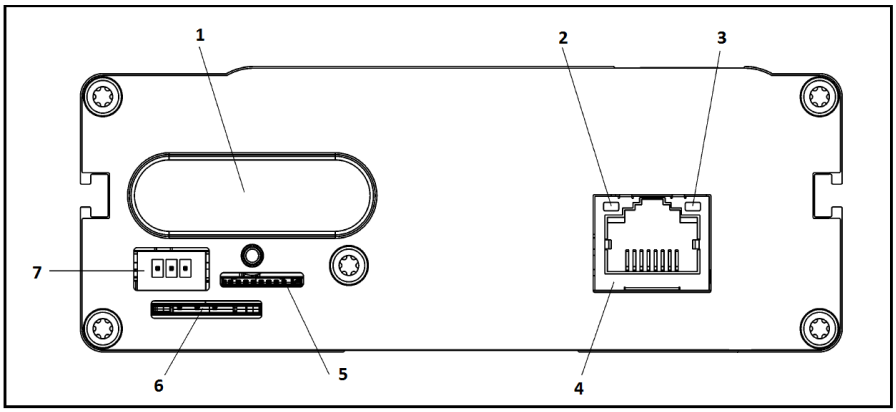


Figura 3.3. Panel Posterior del equipo owa33A

1. Compartimento para batería de respaldo.
2. Led indicador de velocidad 100 Mbps (color amarillo).
3. Led indicador de enlace (color verde).
4. Conector Rj45, Ethernet.
5. Ranura para Micro SD (Extensión para memoria).
6. Ranura de SIM.
7. Conector para batería de respaldo.

3.3.2 Características Internas del Hardware

A continuación se muestra algunos de los módulos que son acoplados para el funcionamiento general del equipo.

3.3.2.1 Módulo GSM/GPRS CINTERION BGS3

El BGS3 es un módulo correspondiente a la familia CINTERION la cual se dedica a la creación de tecnologías para comunicaciones GSM/GPRS, siendo de esta manera capaz de ser integrado en cualquier sistema o producto que necesite la transmisión de información de voz o datos. Por lo

tanto, mejora las capacidades del sistema, transformándolo en un producto independiente.

A pesar de ser un elemento de pequeñas dimensiones, posee características muy avanzadas, diseñadas para facilitar la integración rápida y sencilla para procesos de desarrollo de empresas. Las principales características son:

- Básica funcionalidad M2M con GPRS clase 10.
- Dimensiones: 29.6 x 33.9 x 3.2 mm.
- Peso: 5.5 g.
- Control vía comando AT.
- EDGE y GPRS multi-slot.
- Rango de temperaturade trabajo: -40°C a +85 °C.
- Rango de alimentación: 3.2 -4.5 V.
- Servicios de Internet: TCP, UDP, HTTP, FTP, SMTP, POP3, Ping.
- Protocolo de apilamiento TCP vía comandos AT.

3.3.2.2 Módulo Receptor GPS LEA-6S

El receptor GPS dispone de 50 canales, ofreciendo de esta manera los servicios de navegación de coordenadas longitudinales (Latitud/Longitud), basándose en un sistema de posicionamiento ingenioso llamada u-blox6 que es un extenso arreglo de características con opciones de conectividad flexible.Algunas de sus características son:

- Dimensiones: 22.4 x 17 x 2.4 mm
- Peso 2.1 g
- Voltaje de alimentación: 2.7 a 3.6 V
- Consumo de energía: 115 mW en 3V a máximo rendimiento y 51 mW en 3V en modo ahorro de energía.
- Cuenta con interfaces: UART, USB full speed a 12 Mbps.
- Rango de operación de temperatura: -40 °C a 85 °C.

- Límites de operación: Velocidad a 500 m/s y altitud a 50.000 m.

Para la recepción de los datos del Sistema de Posicionamiento Global (GPS), utiliza un protocolo de datos NMEA (*National Marine ElectronicsAssociation*, Asociación Electrónica Marina Nacional), el cuál inicia con la sentencia $GPRMC (Mínimo de Datos GPS/TRANSIST Específicos Recomendados) con la estructura mostrada en la tabla 3.2.

Tabla 3.2. Sentencia $GPRMC de datos GPS.

Bytes	5	1	10	1	1	1	11	1	12	1	4	1	6	1	6	1	1	1	3
Datos	$GPRMC	Separador	Hora UTC	Separador	V/A	Separador	Latitud	Separador	Longitud	Separador	Velocidad	Separador	Curso	Separador	Fecha	Separador	Separador	Separador	Checksum

Es decir:

$GPRMC, time (hhmmss), (A/V), latitud(ddmm.mmm), (South/North), longitud (dddmm.mmm), (East or West), Velocidad en nudos (kkk.k), dirección (ddd.d), fecha (ddmmyy), CS*.

3.4 SOFTWARE

3.4.1 Descripción General del Firmware

Con el fin de manejar los recursos de la plataforma, librerías completas de las APIs, manejo de GPS, conexión a internet, administración de interfaces, funciones GSM/GPRS y demás funcionalidades válidas, el desarrollador no necesita considerar bajos niveles de lenguajes, tales como: drivers, protocolos. Por lo tanto deben enfocarse en la aplicación por medio de APIs amigables para el usuario. La estructura general del software del owa33A se muestra en la figura 3.4.

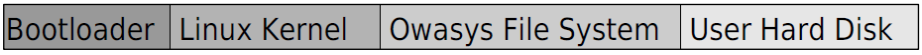


Figura 3.4. Estructura del software del owa33A

3.4.1.1 BootLoader

El BootLoader es el sistema de booteo que administra el kernel y los sistemas de archivos flash. Espera 2 segundos antes de inicializar Linux. Durante este tiempo si el boot recibe un carácter diferente que ENTER a través del puerto serial, inicializará Linux. Si recibe un ENTER, entra al modo de comandos donde recibe caracteres que son interpretados como comandos. Una vez finalizado lo que se desee hacer en el modo de comandos, se ingresa boot e inmediatamente inicializará Linux.

3.4.1.2 Linux Kernel

Es un Kernel estándar de Linux, versión 2.6.25 con opción MMU. Como es un kernel estándar, los aplicativos desarrollados en computadora son fácilmente compatibles con la plataforma del owa33A. El kernel también puede actualizarse y seguir revisiones del kernel estándar.

3.4.1.3 Sistemas De Archivos OWASYS

La estructura del sistema de archivos es como se muestra en la Figura 3.5.

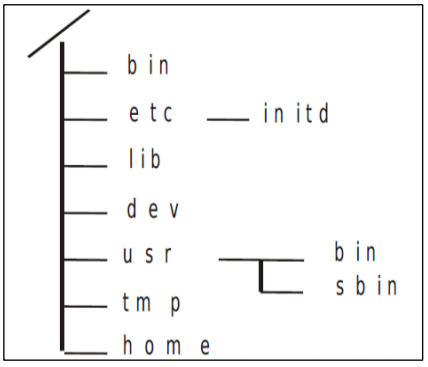


Figura 3.5. Estructura del Sistema de Archivos Linux

- **bin y sbin:** Estos directorios incluyen comandos básicos de Linux para la operación de las utilidades del sistema y de la capa del sistema embebido.

- **etc:** En este directorio se encuentran todos los archivos de configuración y parámetros de booteo que son guardados.

- **dev:** Directorio con todas las especificaciones periféricas.

- **tmp:** En este directorio, el usuario puede almacenar archivos durante la operación del equipo. La información se perderá cada vez que el sistema se reinicie.

- **lib:** Contiene todas las librerías estándares de Linux y librerías del API.

- **usr:** Contiene comandos, librerías, documentación y otros archivos que no cambian durante la operación normal.

3.4.1.4 Disco Duro del Usuario

El disco duro del usuario se encuentra ubicado en el directorio /home, como se muestra en la figura 3.5.

Este directorio puede ser usado como un disco duro de 20 Mbytes, así toda la información almacenada en el directorio es recuperable cuando se reinicia la unidad. En este directorio, el usuario puede crear la estructura de directorio deseada y almacenar todo tipo de archivos dentro de ella. La información almacenada en el directorio /home es escrita en una memoria flash no volátil, mientras que todo lo que se almacena fuera del directorio /home es escrito en la memoria RAM volátil.

3.4.2 Instalación y Uso Del Compilador Cruzado

El compilador cruzado permite al desarrollador fácilmente crear archivos binarios con el contenido del aplicativo deseado que pueden ser directamente ejecutados en el owa33A. Se describe a continuación los pasos para su instalación:

- Instalar una computadora o máquina virtual con Linux.
- Copiar el contenido del directorio /crosscompilera la computadora.
- Ingresar como root.
- Ejecutar el archivo de instalación ./install.sh para instalar automáticamente el compilador cruzado en el directorio /usr/local/arm/[73versión]/.

3.4.3 Conexión de owa33A con la PC

La conexión se puede realizar tanto para sistemas operativos Linux o Windows a través del puerto serial. Por tanto los parámetros de configuración para la conexión son los siguientes:

- Tasa de Bit: 115200 bps
- Bit de Datos: 8
- Paridad: none
- Bit de Parada: 1
- Control de flujo: none

3.4.4 Desarrollo de la aplicación

3.4.4.1 Plataforma y Archivos de desarrollo

El desarrollo de la aplicación, es sobre un sistema operativo Linux Mint 14 edición KDE 64-bit el cual contiene el compilador cruzado para la creación de los archivos binarios. Como se muestra en la figura 3.6, se presentan los archivos creados para el desarrollo, en el que se presentan archivos con extensión **.c** y **.h** que son archivos de programación en Lenguaje C.

Se crea un archivo principal Test_Module.c el cual contiene la programación principal donde se hace los llamados a las funciones de los archivos generados para cada módulo del equipo. Todos los archivos están agregados en el Anexo 1.

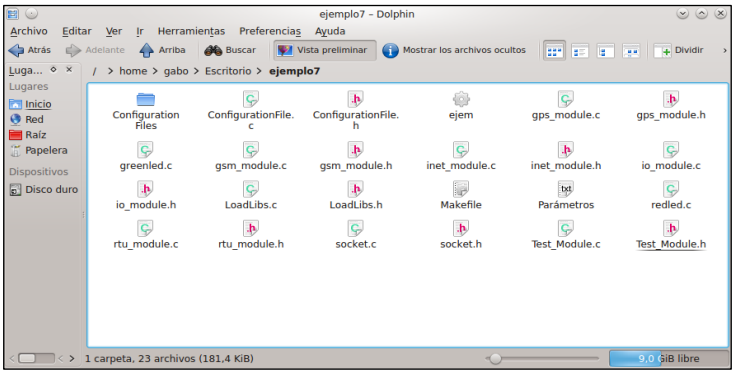


Figura 3.6. Archivos de Programación

El archivo **Makefile** contiene la programación que generará los archivos binarios después de compilar y confirmar que todas las librerías y líneas de código están escritas correctamente. El archivo **ejem** es el archivo binario que será ejecutado en el directorio /home del equipo y que inicializará con la toma de información de los archivos owa.ini y gprs.net.

El archivo **owa.ini** contiene la configuración inicial tal como el nombre del equipo, pin de la SIM, número de la SIM, número del celular al cuál llegarán las alarmas. El archivo **gprs.conf** contiene toda la información correspondiente al operador que le va a permitir la conexión a internet, tales

como: APN, DNS, usuario, contraseña. En esta aplicación se aplicará configuraciones para conectarse a la red GPRS de Telefónica Movistar, a través de la cual se realizará la conexión con la plataforma desarrollada para presentar los datos que el equipo vaya tomando.

3.4.4.2 Acceso a Las Librerías

En la Tabla 3.3 se presenta los archivos y librerías a los que se puede acceder dependiendo de las funcionalidades y módulos que se desea aplicar.

Tabla 3.3. Librerías y Archivos del owa33A

	Archivo Incluido	Archivo cargado	Tipo
RTU	<owa3x/RTUControlDefs.h>	/lib/libRTUControl.so	Librería Dinámica
IO	<owa3x/owa31/IOs_ModuleDefs.h>	/lib/libIOs_Module.so	Librería Dinámica
GSM	<owa3x/GSM_ModuleDefs.h>	/lib/libGSM_Module.so	Librería Dinámica
GPS	<owa3x/GPS_ModuleDefs.h>	/lib/libGPS_Module.so	Librería Dinámica
iNET	<owa3x/INET_ModuleDefs.h>	/lib/libINET_Module.so	Librería Dinámica

Para el proyecto es necesario que se utilicen todas estas librerías en cada uno de los archivos programados en lenguaje C++ con el utilitario específico del módulo de la unidad que se desea. En la figura 3.7 se presenta el código que se utiliza para hacer las llamadas a las librerías dinámicas.

```
VOID* wLibHandle = NULL;

wLibHandle = dlopen("/lib/libGSM_Module.so", RTLD_LAZY);

if (!wLibHandle) {
   printf("No shared library found");
}
```

Figura 3.7. Código de acceso a las librerías dinámicas.

En primer lugar, la aplicación debe tener un controlador para la librería; para ello se usa la llamada del sistema a **dlopen()**. Una vez que la aplicación ha tomado el control de la librería, necesita tener una referencia para funciones posteriores. Para ello es usado la llamada del sistema a **dlsym()**.

```
   INT (*FncDIGIO_SetLED_SW0)( BYTE)
```

Figura 3.8. Código para llamada de un puntero

En la figura 3.8 se presenta el código que se utiliza para realizar una llamada de un puntero a una función. La declaración debe ser realizada para cada una de las funciones que se necesite en la aplicación. La llamada a **dlsym()**, mostrada en la figura 3.9 ayuda a pasar como parámetros el control de la librería y el nombre de la función deseada para arrojar el resultado en el puntero de la función.

```
FncDIGIO_SetLED_SW0= (INT(*)(BYTE))dlsym(wLibHandle, "DIGIO_SetLED_SW0");
if ( dlerror() != NULL) {
   printf("No DIGIO_SetLED_SW0 found..\n");
}
```

Figura 3.9. Llamada a la referencia por dlsym().

La llamada a la función se muestra en la figura 3.10.

```
INT  ReturnCode = NO_ERROR;

if ( (ReturnCode = ( *FncDIGIO_SetLED_SW0)( 1)) != NO_ERROR ) {
   printf("Error %d in DIGIO_SetLED_SW0()\n", ReturnCode);
}
```

Figura 3.10. Llamada a una función.

Todas las funciones retornan **NO_ERROR** en caso de funcionar correctamente; por otro lado si retorna un error, lo hace acorde a un subconjunto de errores definidos para cada API. Para controlar estos errores, la aplicación ejecutará algunas tolerancias para cualquier falla que se produzca.

En caso de que una aplicación haya terminado sin acceder a una librería dinámica, tal vez deje recursos del sistema libre por ser removido de la memoria, y por tal motivo el sistema llama a la función **dlclose()**. La figura 3.11 representa el código para cerrar una librería que no ha sido cargada correctamente.

```
if( (dlclose( wLibHandle) ) != 0) {
   printf( "UnloadExternalLibrary() error\n");
   exit(1);
}
printf( "UnloadExternalLibrary() ok\n");
```

Figura 3.11. Uso de la función dlclose().

3.4.4.3 API Controlador del RTU

En la librería **LibRTUControl** existen cuatro funciones válidas, las cuales se nombran a continuación:

- Un selecto control del sistema: basado en Linux daemon, que ofrece una clara y fácil manera de agrupar puertos síncronos y asíncronos para ser leídos. Es necesario por otros módulos del sistema como GPS y GSM.
- Tiempo del sistema centralizado, válido para aplicaciones y módulos en el sistema.
- Función para introducir retardos. La aplicación de usuario es libre de usar cualquiera de las funciones válidas para funciones sleep en cualquier sistema Linux.
- Seteo de timers válidos para todo tipo de uso.

Inicialización y Finalización del Módulo RTUControl

Internamente la librería LibRTUControl es necesaria para los módulos del sistema de GPS y GSM. Por tal motivo antes de usar los módulos de GPS y GSM, el módulo RTU debe estar inicializada. En la figura 3.12 se muestra el código necesario para inicializar y finalizar el módulo RTU.

```
#define LIBRTU    "/lib/libRTUControl.so"

VOID *LibHandleRTU = NULL;

//Load RTUControl library
LoadExternalLibrary(LIBRTU, &LibHandleRTU);

// load RTU Control functions
LoadRTUControlFunctions(LibHandleRTU);

if( ( ReturnCode = ( *Fnc_RTUControl_Initialize)(NULL) != NO_ERROR)
{
   printf( "Error %d in RTUControl_Initialize()\n", ReturnCode);
   UnLoadExternalLibrary(LibHandleRTU);
   return 1;
}

if( ( ReturnCode = ( *FncRTUControl_Start)()) != NO_ERROR)
{
   printf( "Error %d in RTUControl_Start()\n", ReturnCode);
   (*FncRTUControl_Finalize)();
   UnLoadExternalLibrary(LibHandleRTU);
   return 1;
}
```

Figura 3.12. Código de Inicialización y finalización del módulo RTU.

Tiempo del sistema

El equipo owa33A está provisto con un Reloj de tiempo real (RTC), con opciones de calendario y completamente programable usando las funciones API. El reloj tiene una batería de respaldo dedicada, y así mantiene el tiempo cuando la unidad está apagada.

Una vez que se conoce el tiempo, hay cuatro funciones dentro del API del RTUControl para administrar el tiempo del kernel de Linux. En la figura 3.13, se presenta el código que es necesario para que el programador pueda obtener o establecer el tiempo del sistema. Estas funciones cuentan con un tipo de estructura TSYSTEM_TIME.

```
INT GetSystemTime( TSYSTEM_TIME *wSystemTime);
INT SetSystemTime( TSYSTEM_TIME wSystemTime);
```

Figura 3.13. Funciones para obtención y establecimiento del reloj.

Para administrar el tiempo del RTC, el cual trabaja con un patrón del tiempo del sistema se utilizan las funciones de la figura 3.14 donde hacen el mismo efecto que las funciones explicadas anteriormente pero dentro del RTU.

```
UINT RTUGetHWTime( THW_TIME_DATE *CurrentTime);
UINT RTUSetHWTime( THW_TIME_DATE CurrentTime);
```

Figura 3.14. Función de obtención y establecimiento del reloj - nivelRTU.

Funciones Sleep and usecsleep

Estás funciones permiten al equipo entrar a un tiempo de suspensión por un determinado tiempo especificado en segundos como se muestra en la figura 3.15.

```
usecsleep(INT seconds, INT useconds)
```

Figura 3.15. Funciones sleep y usecsleep

Administración De Alimentación Del Equipo

La librería LibRTUContol provee funcionalidades requeridas para establecimiento de dispositivos en dos diferentes modos de alimentación: **Modo Standby** y **Modo Stop**.

Las señales configuradas que pueden accionar al equipo son:

- Movimiento: Se acciona si el sensor de movimiento detecta algún movimiento.

- RXD2: Se acciona si existe alguna recepción de datos desde el módulo GSM.
- CAN1RD: Se acciona si existe alguna recepción de datos desde el bus CAN.
- DIN0…9: Se acciona si una señal externa en DIN0…9 es activada.
- RXD0: Se acciona si un carácter es recibido en el pin RX0 de la interface RS232.
- PWRFAIL: Se acciona en caso de detectar que la unidad no está siendo alimentada externamente.

Sensor de Movimiento

El dispositivo cuenta con un acelerómetro que muestra si ha existido movimiento dependiendo de la aplicación que se desea ejecutar. La librería LibRTUControl tiene dos funciones para obtención y reseteo del estado por el que este cursando el equipo. Las funciones que determinan el movimiento son presentadas en la figura 3.16.

```
INT RTUGetMoved(UCHAR *MovedValue);
INT RTUResetMoved(void);
```

Figura 3.16. Funciones Sensor de Movimiento

La función **RTUGetMoved** retorna el parámetro en la bandera MOVED que representa el estado. Si este es 0 significa que la unidad no se ha movido; por otro lado si contiene un 1 significa que el equipo ha sido movido. Mientras que la función **RTUResetMoved** resetea el valor del parámetro MOVED. Si después del reseteo el equipo tiene algún movimiento vuelve a establecer el valor correspondiente; por lo tanto después de cada cambio que sufra, es necesario resetearlo para conocer los estados.

3.4.4.4 API Del Módulo de IOs

La librería LibIOs_Moduleadministra un recurso para acceder a la arquitectura del sistema de IOs, ofreciendo algunos servicios:

- Acceso para escritura/lectura de las entradas/salidas digitales.
- Acceso de lectura para las entradas analógicas.
- Administración del encendido de los LEDs.

Establecer El Estado De Un LED

Algunos de los leds pueden ser controlados por el usuario (verde, rojo) excepto el led naranja que corresponde al funcionamiento del GPS y el led amarillo que corresponde a GSM. Por defecto, el LED verde se prende después de inicializar el owa33A, pero dependiendo de la aplicación que se ejecute se puede cambiar el estado del led. Así como el led rojo se encuentra en un estado de apagado.

El código correspondiente para establecer que se encienda o apague uno de los leds, se muestra en la figura 3.17

```
if((ReturnCode = (*FncDIGIO_SetLED_SW1)(1)) != NO_ERROR) //POWERS ON LED
printf("Error %d in DIGIO_SetLED_SW1()\n", ReturnCode);

if((ReturnCode = (*FncDIGIO_SetLED_SW1)(0)) != NO_ERROR) //POWERS OFF
LED
printf("Error %d in DIGIO_SetLED_SW1()\n", ReturnCode);
```

Figura 3.17. Seteo de los LEDS

Inicialización y Finalización Del Módulo de IOs

Primero se carga la librería de IOs, la cual es llamada **LibIOs_Module**, permite ejecutar los servicios especificados anteriormente. Una vez cargada la librería se debe inicializar las entradas/salidas; en caso de no devolver

NO_ERROR se cierran las librerías completamente como se muestra en la figura 3.18.

```
#define LIBIO "/lib/libIOs_Module.so"
VOID *LibGPIOHandle = NULL;
LoadExternalLibrary( LIBIO, &LibIOHandle);
LoadIOsGenericFunctions(LibIOHandle);
LoadIOsFunctions(LibIOHandle);
LoadIOsAnalogicFunctions(LibIOHandle);
printf("All external IOs functions loaded\n");

// Initialize & Start IOs
if( ( ReturnCode = (*FncIO_Initialize)() != NO_ERROR) {
        printf("Error %d in IO_Initialize()\n", ReturnCode);
        UnloadExternalLibrary(LibIOHandle);
        return 1;}
if( ( ReturnCode = (*FncIO_Start)()) != NO_ERROR) {
        printf("Error %d in IO_Start()\n", ReturnCode);
        UnloadExternalLibrary(LibIOHandle);
        return 1;}
```

Figura 3.18. Código de Inicialización Módulo IOs

Elección De Entradas Digitales

Con el fin de obtener el valor de una de las entradas digitales, se toma una función presentada en la figura 3.19 y a la cual se puede acceder por medio de llamadas a la librería **LibIOs_Module.**

```
if( (ReturnCode = (*FncDIGIO_Get_DIN)(1, &din1)) != NO_ERROR)
{
        printf("Error %d in FncDIGIO_Get_DIN()", ReturnCode);
        return 1;}
```

Figura 3.19. Código Obtención de valores en estradas/salidas digitales.

Como se puede notar en el código la llamada de la función se la realiza para obtener el valor que tenga en la entrada digital 1 en la cual **&din1** es un puntero que va guardando el resultado que va teniendo de la lectura; En caso de devolver un valor erróneo en la lectura da un aviso de error y sale de la función.

3.4.4.5 API Del Módulo GSM

La librería de GSM funciona de igual manera que las de RTU y IOs al llamar a dos funciones encargadas de la inicialización del módulo de GSM y que son: **GSM_Initialize()** y **GSM_Start()**. Se debe tener muy claro que se debe cargar e inicializar los módulos de RTU e IOs antes de levantar el módulo de GSM.

En la Figura 3.20 se puede observar el código necesario para que inicie el módulo GSM en el cuál se puede notar que antes de inicializarlo se debe ingresar el PIN para tener acceso a la SIM insertada en el equipo. El número de PIN ingresado se ubica dentro de un puntero el cual guardará todo su contenido dentro de la función **gsm_event_handler**. Para inicializar toda la configuración del módulo, se declara al inicio un semáforo de C/C++, lo que permite que varios procesos o hilos accedan a un recurso en común.

Una vez habilitado el semáforo, se inicializa el módulo de GSM y crea un hilo al final del código para empezar a ejecutar un proceso; el proceso permitirá que se verifique el módulo de GSM constantemente para ejecutar peticiones según los requisitos que se le pida al equipo mediante el archivo programado por el usuario.

```
//GSM events Handler
sem_init(&gsmHandlerSem, 0, 0);
// *************** All of the Lib and Lib Functions Loaded.
// *************** Starting Generic Application Note #1 Program
printf ("Starting GSM Application Note #1 Program\n");
printf ("Insert PIN NUMBER: ");
memset( ( VOID *) &keyEntry, 0, sizeof( keyEntry));
getEntry(keyEntry);
memset(&gsmConfig, 0, sizeof( TGSM_MODULE_CONFIGURATION));
if(keyEntry[ 0] == 0)
        strcpy( ( CHAR*) gsmConfig.wCode, "");
else
        strcpy( ( CHAR*)( gsmConfig.wCode), ( CHAR*) keyEntry);
gsmConfig.gsm_action = gsm_event_handler;
InitGsmEventBuffer();
if( ( ( *FncGSM_Initialize) ( ( VOID*) ( &gsmConfig))) != NO_ERROR)
        exit( 1);
if( ( ( *FncGSM_Start) ( )) != NO_ERROR)
        exit( 1);
( *FncGSM_IsActive) ( &isActive);
if(isActive == 1){
        printf("\n**** GSM IS UP AND RUNNING *****\n");
}else{
        printf("\n****GSM IS NOOOOOT RUNNING *****\n");
}
runHandleEvents = true;
pthread_create(&gsmEvents,NULL,handleEvents,NULL);
```

Figura 3.20. Código para Cargar Módulo GSM.

Gestión de Eventos GSM

Cada vez que un evento sea reportado a la función del gestor, genera rutinas para establecer una bandera la cual indica el evento GSM que está pendiente y da señales al semáforo para acceder al proceso. En caso de que no existieran eventos el hilo se encontrará descansando en el semáforo, mientras que cuando exista un evento la función **gsm_event_handler** despertará al hilo. En la figura 3.21 se muestra el código que se presenta en la función del gestor de eventos.

En la figura 3.22 se muestra el código para determinar el tipo de evento. Se entiende a esta función como un lazo controlado por una bandera de tipo bool y que trabaja de manera global; para aprovechar recursos y usos del procesador, cuando no exista un evento se suspenderá el hilo

llamando a **sem_wait(&gsmHandlerSem)** hasta que llegue un nuevo evento.

```
static VOID gsm_event_handler( gsmEvents_s *pToEvent)

{
        INT auxi = (GsmEventsWr+1);
        GsmEventBuffer[GsmEventsWr] = *pToEvent;
        if(GsmEventsWr == GsmEventsRd) {
                GsmEventsWr = auxi;
        } else {
                if(auxi>= MAX_EVENTS) {
                auxi = 0;
                }

                if(auxi != GsmEventsRd) {
                GsmEventsWr = auxi;
                }
        }

        if(GsmEventsWr>= MAX_EVENTS) {
                GsmEventsWr = 0;
        }
        sem_post(&GsmEventsSem);
}
```

Figura 3.21. Función Gestor de Eventos GSM

```
VOID* GSMHandleEvents( VOID *arg)
{
        gsmEvents_s *owEvents;
        gsmEvents_sLocalEvent;
        //User Vars.
        INT retVal;
        //RING
        BYTE ringTimes = 0;
        //SMS
        INT SMSIndex;
        SMS_s incomingSMS;
        UCHAR SMSSize;
        while(runGSMHandler == TRUE){
                sem_wait(&GsmEventsSem);
                if(GsmEventsRd == GsmEventsWr) {continue;}
                LocalEvent = GsmEventBuffer[GsmEventsRd++];
                owEvents = &LocalEvent;
                if(GsmEventsRd>= MAX_EVENTS) {
                        GsmEventsRd = 0;}
```

```
switch ( owEvents->gsmEventType){
        case GSM_NO_SIGNAL:
        break;
        case GSM_RING_VOICE:
        case GSM_RING_DATA:
        ringTimes ++;
        if(owEvents->gsmEventType == GSM_RING_DATA){
                printf( "OWASYS--> GSM RING DATA signal Phone Number:
                %s \n",owEvents->evBuffer);
        } else {
                printf( "OWASYS--> GSM RING VOICE signal Phone Number:
                %s \n",owEvents->evBuffer);
        }
        break;
        case GSM_NEW_SMS:
                [...]
                break;
                default:
                printf( "Unknown temperature status\n");
                break;
        }
        break;
        default:
                printf( "OWASYS--> Signal Event not found ...%d \n",
owEvents->gsmEventType);
        }}
return NULL;}
```

Figura 3.22. Función Eventos del Gestor

Finalización del Módulo GSM

Una vez que se desee detener la aplicación, se debe finalizar los módulos en sentido contrario a la inicialización; por lo tanto se debe finalizar el módulo de GSM antes de los módulos RTU e IOs. El código presentado en la figura 3.23 muestra la manera de finalizar GSM, en donde se puede notar que primero finaliza el semáforo, luego el hilo y por último descarga las librerías de GSM que habían sido llamadas al comenzar la aplicación.

```
( *FncGSM_Finalize) ( );
runGSMHandler = false;
sem_post(&GsmEventsSem);
pthread_join(gsmEvents, NULL);
( *FncIO_Finalize)( );
( *FncRTUControl_Finalize) ( );
UnLoadExternalLibrary(LibHandleGSM);
UnLoadExternalLibrary(LibHandleIO);
UnLoadExternalLibrary(LibHandleRTU);
printf( " Ending the GSM Application Note #1\n");
exit ( 0);
```

Figura 3.23. Código Finalización GSM

3.4.4.6 API Del Módulo iNet

La librería iNet provee todas las funciones necesarias para conectar el equipo a internet, obteniendo una IP al establecer una sesión de GPRS.

Inicialización Conexión a Internet

Para establecer la conexión a internet, de igual manera se cargará la librería LinINET_Module, que contiene las funciones necesarias para realizar las configuraciones que levantarán el módulo para la transmisión de datos. El código presente en la figura 3.24 muestra el orden en que se deben cargar las librerías hasta llegar al iNet.

```
printf( " Load RTUControl library\n");
LoadExternalLibrary( “/lib/libRTUControl.so”, &LibHandleRTU);
printf( " Load libIOs_Module library\n");
LoadExternalLibrary("/lib/libIOs_Module.so", &LibHandleIO);
printf( " LOADING libGSM_Module library\n");
LoadExternalLibrary("/lib/libGSM_Module.so", &LibHandleGSM);
printf( " LOADING libINET_Module library\n");
LoadExternalLibrary( “/lib/libINET_Module.so”, &LibHandleINET);
LoadRTUControlSignalManagementFunctions(LibHandleRTU);
LoadRTUControlGenericFunctions(LibHandleRTU);
LoadIOsGenericFunctions(LibHandleIO);
LoadGSMGenericFunctions(LibHandleGSM);
```

Figura 3.24. Código Cargar Librería iNet

La inicialización del módulo de iNet es igual a GSM, tiene presente semáforos e hilos para ejecutar subprocesos funcionales con paquete de datos que se debe tener contratado en la tarjeta SIM. El código para la creación del semáforo e hilos se muestra en la figura 3.25.

```
TINET_MODULE_CONFIGURATION    iNetConfiguration;
GPRS_CONFIGURATION                    gprsConfiguration;
sem_init(&iNetHandlerSem, 0, 0);
runiNetHandleEvents = true;
pthread_create(&iNetEvents, NULL, iNetHandleEvents, NULL);
```

Figura 3.25. Código Inicialización iNet.

Antes de iniciar, el módulo necesita de algunos parámetros de configuración para la conexión; por lo tanto se ingresan los datos: Nombre de Usuario, Contraseña, DNS1, DNS2 y APN como se muestra su código en la figura 3.26.

```
printf ("Insert USER: ");
memset( ( VOID *) &strEntry, 0, sizeof( strEntry));
getEntry(strEntry);
strcpy( ( CHAR*) gprsConfiguration.gprsUser, ( CHAR*) strEntry);
printf ("Insert PASSWORD: ");
memset( ( VOID *) &strEntry, 0, sizeof( strEntry));
getEntry(strEntry);
strcpy( ( CHAR*) gprsConfiguration.gprsPass, ( CHAR*) strEntry);
printf ("Insert DNS1: ");
memset( ( VOID *) &strEntry, 0, sizeof( strEntry));
getEntry(strEntry);
strcpy( ( CHAR*) gprsConfiguration.gprsDNS1, ( CHAR*) strEntry);
printf ("Insert DNS2: ");
memset( ( VOID *) &strEntry, 0, sizeof( strEntry));
getEntry(strEntry);
strcpy( ( CHAR*) gprsConfiguration.gprsDNS2, ( CHAR*) strEntry);
printf ("Insert APN: ");
memset( ( VOID *) &strEntry, 0, sizeof( strEntry));
getEntry(strEntry);
strcpy( ( CHAR*) gprsConfiguration.gprsAPN, ( CHAR*) strEntry);
iNetConfiguration.wBearer = INET_BEARER_GPRS;
iNetConfiguration.inet_action = inet_event_handler;
InitInetEventBuffer();
iNetConfiguration.wBearerParameters = (VOID*) &gprsConfiguration;
(*Fnc_iNetInitialize)( ( VOID*) &iNetConfiguration);
ReturnCode = ( *Fnc_iNetStart) ( );
if(ReturnCode != NO_ERROR){
        iNetFinalized = TRUE;
        runiNetHandleEvents = FALSE;
        sem_post(&iNetHandlerSem);
        sem_destroy(&iNetHandlerSem);
        printf("OWASYS--> ERROR Initializing the Internet
        session(%d)\n",ReturnCode);}
else {
        iNetFinalized = FALSE;
        printf("OWASYS--> OK Internet session started\n");}
```

Figura 3.26. Código Programación Parámetros GPRS

Una vez ingresado correctamente los parámetros de configuración GPRS, son guardados dentro de una estructura**iNetConfiguration** que ingresan a una función conocida como **Fnc_iNetInitialize**; En esta función se realizará la verificación de los datos ingresados y de ser correctos, establece la corrección con el proveedor del servicio de telefonía celular y del paquete de datos.

Una vez realizada la conexión GPRS con el proveedor de servicios, el equipo está listo para realizar el envío de datos hacia un servidor; la conexión con el servidor se la realizará mediante TCP especificando la

dirección IP del servidor y el protocolo por el cual está dedicado para recibir la información.

Cierre de Conexión a Internet

Para finalizar la conexión a internet se debe seguir un proceso especificado en la figura 3.27 y que son:

- Llamar a la función iNet_Finalize().
- Detener el controlador de eventos del módulo iNet.
- El semáforo detiene los procesos que hayan estado funcionando.
- Se mata todos los hilos que hayan sido controlados por el controlador de eventos.
- Se mata al proceso del semáforo.
- Se descargan las librerías del iNet.

```
( *FnciNet_Finalize) ( );
runiNetHandleEvents = FALSE;
sem_post(&iNetHandlerSem);
pthread_join(iNetEvents,NULL);
sem_destroy(&iNetHandlerSem);
UnloadExternalLibrary(LibHandleINET);
```

Figura 3.27. Código Finalización Conexión a Internet

3.4.4.7 API Del Módulo GPS

Inicialización Del Módulo GPS

Al ser un módulo que funciona de manera parecida a GSM, por lo tanto necesita que se carguen las librerías de RTU e IOs para poder cargar su librería **LibGPS_Module**. En la figura 3.28 se puede ver el código determinado para cargar las librerías en su determinado orden.

```
VOID *LibGPSHandle = NULL;
VOID *LibControlHandle = NULL;
VOID *LibGPIOHandle = NULL;
//load RTU control library.
LoadExternalLibrary( LIBRTU, &LibControlHandle);
LoadRTUControlFunctions(LibControlHandle );
//load IOs library.Needed for timers use.
LoadExternalLibrary( LIBIO, &LibGPIOHandle);
LoadIOsFunctions(LibGPIOHandle);
//load GPS library.
LoadExternalLibrary( LIBGPS, &LibGPSHandle);
LoadGPSFunctions(LibGPSHandle );
```

Figura 3.28. Código Carga de Librería GPS

Una vez cargadas las librerías, se procede a inicializar el módulo de GPS muy distintamente de GSM. En la figura 3.29 se presenta un código completo de cómo se debe programar la inicialización paso a paso, empezando por la configuración de los módulos necesarios que deben arrancar antes de el de GPS.

```
INT ReturnCode;
TGPS_MODULE_CONFIGURATION GPSConfiguration;

// RTUControl module initialization.
if( ( ReturnCode = (*FncRTUControl_Initialize)( NULL) ) != NO_ERROR) {
        printf( "Error %d in RTUControl_Initialize()\n", ReturnCode);
        UnloadExternalLibrary(LibGPSHandle);
        UnloadExternalLibrary(LibControlHandle);
        UnloadExternalLibrary(LibGPIOHandle);
        Return 1;
}
// RTUControl module startup.
if( ( ReturnCode = (*FncRTUControl_Start)() ) != NO_ERROR) {
        printf( "Error %d in RTUControl_Start()\n", ReturnCode);
        (*FncRTUControl_Finalize)();
        UnloadExternalLibrary(LibGPSHandle);
        UnloadExternalLibrary(LibControlHandle);
        UnloadExternalLibrary(LibGPIOHandle);
        Return 1;
}
// IOs module initialization.
memset(&IOConfiguration, 0, sizeof( TIOS_MODULE_CONFIGURATION));
if( ( ReturnCode = ( *FncIO_Initialize)( ( VOID *)&IOConfiguration)) !=
NO_ERROR) {
        printf( "Error %d in IO_Initialize()\n", ReturnCode);
        (*FncRTUControl_Finalize)();
        UnloadExternalLibrary(LibGPSHandle);
        UnloadExternalLibrary(LibControlHandle);
        UnloadExternalLibrary(LibGPIOHandle);
        Return 1;
}

// IOs module startup.
if( ( ReturnCode = ( *FncIO_Start)()) != NO_ERROR) {
        printf( "Error %d in IO_Start()\n", ReturnCode);
        (*FncRTUControl_Finalize)();
        (*FncIO_Finalize)();
        UnloadExternalLibrary(LibGPSHandle);
        UnloadExternalLibrary(LibControlHandle);
        UnloadExternalLibrary(LibGPIOHandle);
        Return 1;
}
// set GPS configuration parameters.
memset(&GPSConfiguration, 0, sizeof( TGPS_MODULE_CONFIGURATION));
GPSConfiguration.DeviceReceiverName = GPSValidType[1];
GPSConfiguration.ParamBaud = B9600;
GPSConfiguration.ParamParity = IGNPAR;
GPSConfiguration.ParamLength = CS8;
GPSConfiguration.ProtocolName = GPSValidProtocol[0];
GPSConfiguration.GPSPort = COM6;
```

```
// GPS module initialization.
if( ( ReturnCode = ( *FncGPS_Initialize)( ( void *)
        &GPSConfiguration)) != NO_ERROR) {
        printf( "Error %d in GPS_Initialize()\n", ReturnCode);
        (*FncRTUControl_Finalize)();
        (*FncIO_Finalize)();
        UnloadExternalLibrary(LibGPSHandle);
        UnloadExternalLibrary(LibControlHandle);
        UnloadExternalLibrary(LibGPIOHandle);
        Return 1;
}
// GPS receiver startup.
if( ( ReturnCode = (*FncGPS_Start)()) != NO_ERROR ) {
        printf( "Error %d in GPS_Start()\n", ReturnCode);
        (*FncGPS_Finalize)();
        (*FncIO_Finalize)();
        (*FncRTUControl_Finalize)();
        UnloadExternalLibrary(LibControlHandle);
        UnloadExternalLibrary(LibGPIOHandle);
        UnloadExternalLibrary(LibGPSHandle);
        Return 1;
}
```

Figura 3.29. Código Inicialización Módulo GPS

En el código presentado se determina una variable tipo TGPS_MODULE_CONFIGURATION, y no es más que una estructura que contiene los parámetros que configurarán el módulo de GPS para su funcionamiento. Luego de cargar estos parámetros se procede a iniciar el módulo con la función ***FncGPS_Initialize()**; en caso de ocurrir un error al ejecutar la función se finaliza los módulos cargados de RTU e IOs y se descarga las librerías para salir de la aplicación.

De igual manera después de inicializar el módulo, se procede a activar el receptor de los datos de GPS con la función **FncGPS_Start()**; si no tiene ningún error prosigue con el código programado en el programa central, pero en caso de que se tenga un error se finaliza los módulos de GPS, IOs y RTU, al igual que cerrar la carga de las librerías que fueron llamadas al inicio de la programación.

Finalización Del Módulo GPS

Para la finalización del módulo se debe seguir los pasos contrarios a los de inicialización como se muestra en la figura 3.30 empezando con GPS, RTU y por último con el IOs.

```
(*FncGPS_Finalize)();
(*FncRTUControl_Finalize)();
//IOs must be the last to finalize because it closes the IOs driver.
(*FncIO_Finalize)();
UnloadExternalLibrary(LibGPSHandle );
UnloadExternalLibrary(LibControlHandle );
UnloadExternalLibrary(LibGPIOHandle );
```

Figura 3.30. Código Finalización Módulo GPS

Creación del Ejecutable

El ejecutable se lo realiza en un Sistema Operativo Linux debido al compilador cruzado que se encuentra instalado dentro del owa33A; por lo tanto para generar el ejecutable se debe tener el mismo kernel e instalado el compilador arm.

La manera de escribir en un terminal es:

- **arm-linux-gnueabi-gcc -c -o ejecutable.o (archivos).c**
- **arm-linux-gnueabi-gcc -Wall ejecutable.o -o ejecutable**

En la primera línea en el lugar de archivos, se procede a colocar el nombre de todos los archivos .c que se haya creado y que se encuentren como librerías ya que entre todos los archivos, especialmente el archivo principal hará un llamado de las funciones establecidas en el momento que se lo necesite. En la segunda línea se procede a convertir el archivo generado ejecutable o en el ejecutable, que se crea como un archivo binario y por lo tanto no presenta ninguna extensión.

Para proceder a ejecutarlo, primero se copia este archivo dentro del directorio /home del owa33A; en caso de estar conectado por consola mediante el hyperterminal.

Funcionamiento General del Programa

En la figura 3.31 se presenta un diagrama de flujo del funcionamiento general del programa. Una vez que se inicia la aplicación de carga las librerías de los módulos necesarios, que luego son inicializados y en caso de no poder levantarse descargan las librerías y finaliza la aplicación; en caso de levantarse sin problemas, pasa a ejecutar las líneas de comando para un proceso y a su vez se hace una comprobación de los módulos para determinar si están funcionando adecuadamente.

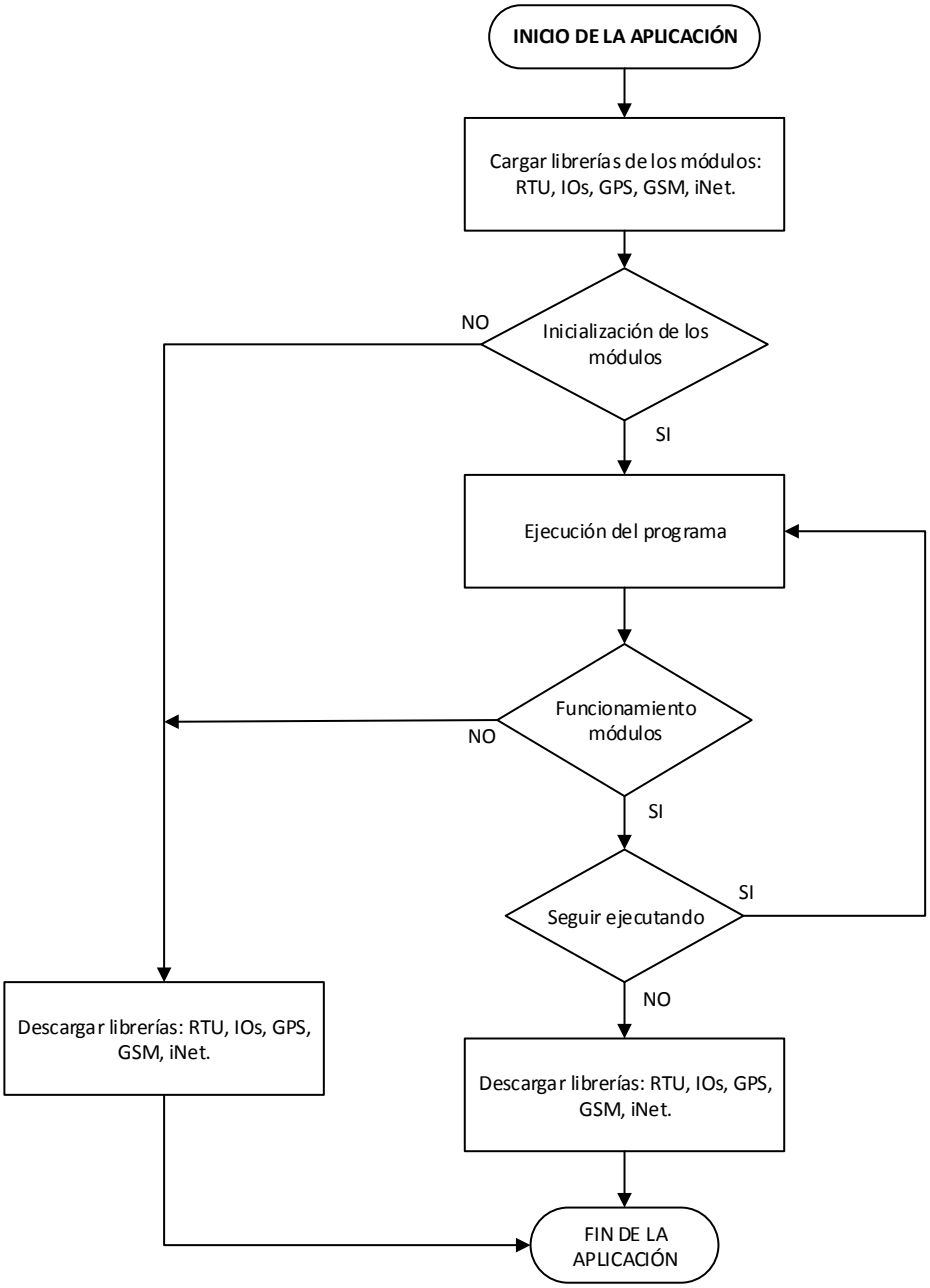


Figura 3.31. Diagrama de Flujo Programa

Una vez que se ha determinado que los módulos funcionan de manera adecuada, se verifica si se desea seguir ejecutando la aplicación y si es el caso de seguir, retorna a la ejecución de programa; en caso de que el programa ya no desee seguir o necesita parar, descarga las librerías de los módulos y finaliza la aplicación.

3.4.5 Desarrollo de la Plataforma Web

3.4.5.1 Servidor Web

El Servidor Web está montado con un Sistema Operativo Linux Ubuntu 11.04 con un kernel 3.0.5, cuya instalación se presenta en el Anexo 2. Las características que debe tener presente el servidor para poder realizar el correcto monitoreo de los datos enviados por el equipo son:

- Guardar los datos enviados desde el equipo en una base de datos.
- Establecer seguridad al ingresar en la plataforma.
- Presentación de datos por tablas.
- Mostar ubicación del equipo en un mapa.
- Todos los datos deben presentar la hora y fecha a la que se presentó la alarma.

Para el levantamiento de un Servidor Web deben ser instalados tres elementos importantes, que para el caso de la plataforma que se va a realizar son esenciales:

1. Apache 2
2. Phpmyadmin
3. PHP

Apache 2

Es el servidor web más común utilizado en sistemas operativos Linux y el cual funciona bajo peticiones de un cliente para mostrar el contenido de la página web; Un cliente ingresa mediante una URL que es la dirección que especifica al servidor web, el cual reconoce que es una petición basada en el protocolo *Hyper Text Transfer Protocol* (HTTP), o a su vez si se desea

mayor seguridad con el protocolo *Hyper Text Transfer ProtocolSecure* (HTTPS).

PhpMyadmin

Es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web, utilizando internet. Se puede crear y eliminar Base de Datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier secuencia SQL, administrar claves en campos, administrar privilegios, exportar datos en varios formatos.

PHP

Lenguaje de programación especialmente adecuado para el lado del servidor de desarrollo web en PHP y que generalmente es ejecutado en un servidor web. Los archivos de PHP pueden ser programados de tal manera que presenten contenido dinámico y pueda ser más amigable el ambiente hacia el usuario.

Para mayor seguridad del servidor con el fin de evitar un ataque, se procedió a cambiar el puerto de escucha; es decir, que por defecto utiliza el puerto 80 para HTTP, mientras que para el proyecto se cambió al puerto 9090 y se lo hace dentro de los archivos instalados de apache 2 como se muestra en el Anexo4.

3.4.5.2 Plataforma

La plataforma es desarrollada en lenguaje HTML y PHP con conexiones CSS para dar una interacción mucho más gráfica con el usuario y a su vez tiene que ser de fácil entendimiento para acceder a todas sus funciones. El código de las platillas se presenta en el Anexo 4.

Para acceder a la plataforma se debe contar primero con una dirección IP pública, que debe ser establecida por el proveedor de internet que esté presente en el lugar donde se ubicará el servidor. Con la IP provista se procede a realizar un nateo en el *Access Point* en el lugar de trabajo para poder direccionar las peticiones que se realicen a la IP pública hacia una IP privada, la cual será establecida en el equipo con el servidor.

Para poder acceder a la plataforma desarrollada que se muestra en la figura 3.32, se procede a ingresar la URL 186.4.224.249:9090; de esta manera se presenta el inicio de la página web, donde se debe ingresar el nombre de usuario y la clave para poder acceder a la visualización de los datos. En caso de poner un usuario o clave erróneos no podrá acceder.

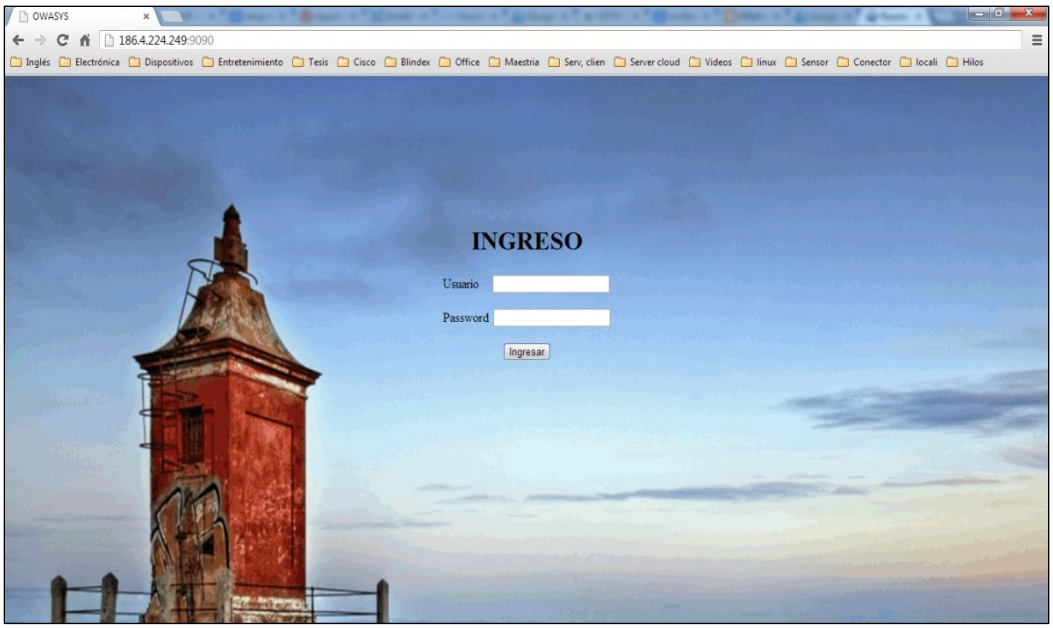


Figura 3.32. Ingreso a la plataforma

Una vez que el usuario se haya registrado correctamente, se direccionará hacia la interacción de datos como se muestra en la figura 3.33. Dentro se esta se muestran 5 pestañas:

- **Home:**seindica una descripción general de las comunicaciones M2M.

- **Localización:** muestra una tabla con el nombre del equipo, fecha y hora que se tomaron los datos en el quipo y parámetros de latitud y longitud tomadas por el equipo desde el módulo GPS.

- **Mapa:**permite visualizar un mapa de *OpenStreetMap* en la cual se presenta un ícono de un automóvil dando la ubicación del equipo.

- **Historial:**presenta una tabla con la fecha y hora en que se tomaron los datos de temperatura y la temperatura que el sensor adquirió desde el equipo.

- **Alarmas:** Muestra una tabla con la fecha, hora y el tipo de alarma que se produzca en el equipo. Se puede mostrar alarmas por movimiento del equipo o por apertura de puertas.

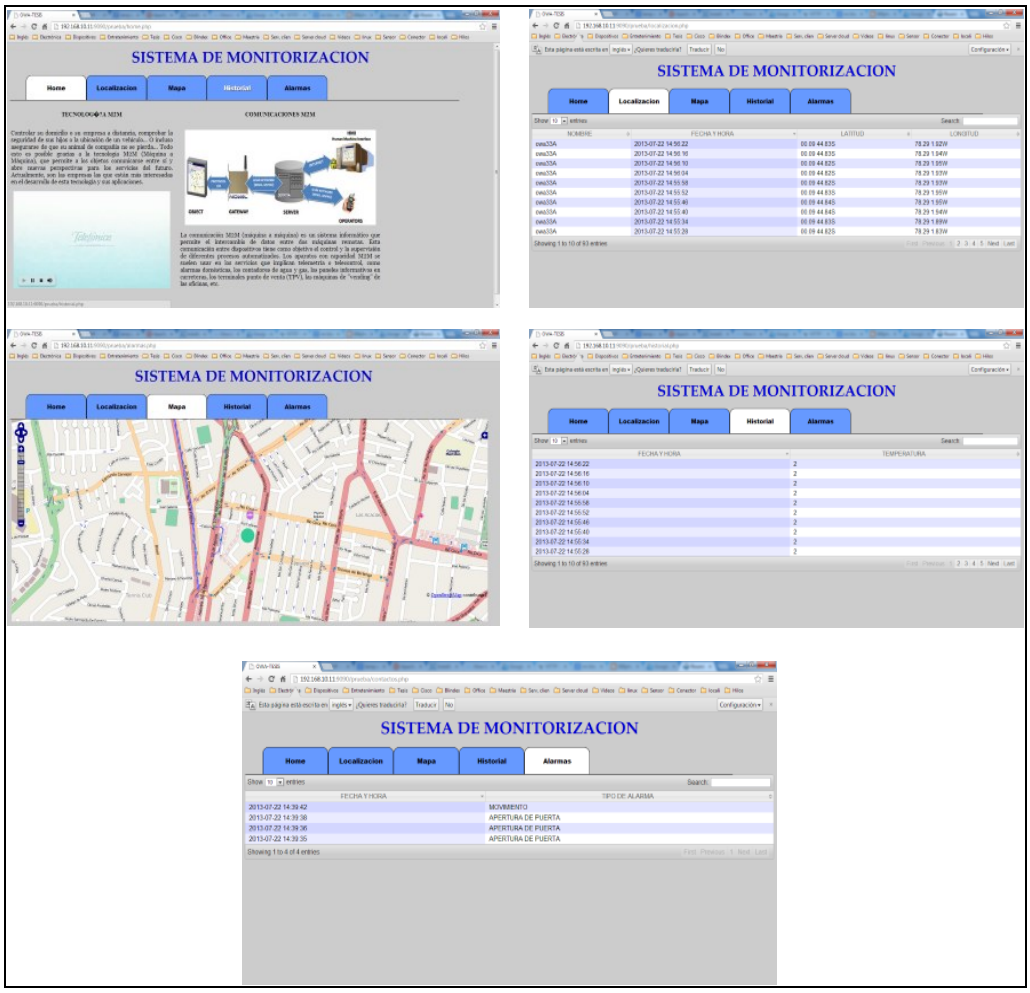


Figura 3.33. Plataforma

Comunicación y Base de Datos

La comunicación que establece el equipo con la plataforma la hace mediante sockets, es decir establece un socket con la IP del servidor y el puerto de escucha en la que se va a encontrar para la recepción de los datos. El puerto establecido para el envío de datos es el 9595. El código con la programación del socket se muestra en el Anexo 5.

La Base de Datos se la realizó en MySQL mediante PhpMyAdmin para solventar un trabajo más gráfico de las tablas como se muestra en la figura 3.34, en la cual creamos dos bases de datos:

- **Usuarios:** contiene los nombres de usuarios y contraseñas con los que se rivalidará para el acceso a la plataforma.

- **Tablas:**contiene las tablas para guardar la información enviada desde el equipo como se muestra detalladamente en la tabla 3.4.

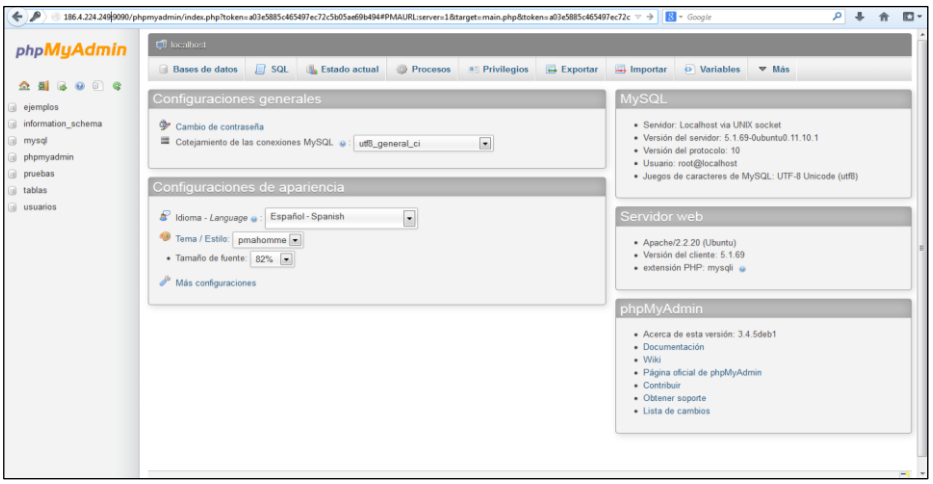


Figura 3.34. PhpMyAdmin

Tabla 3.4. Tablas de la base de datos tablas

TABLA	CONTENIDOS
Alarmas	Fecha(DATETIME) Tipo de alarma(VARCHAR)
Localización	Nombre del equipo(VARCHAR) Fecha y hora(DATETIME), Latitud(VARCHAR), Longitud(VARCHAR)
Temperatura	Fecha(DATETIME), Temperatura(DOUBLE)

El código escrito en C/C++ para guardar datos desde en MySQL, se lo presenta en el Anexo 6, en el que se describe como cargar las librerías de MySQL y como se debe escribir a manera de que MySQL interprete lo que el usuario desea hacer con la base de datos y la tabla correspondiente.

La información que es enviada por el equipo y recibida por el servidor, la hace en un solo string; el string se lo debe separar por cadenas de caracteres para ir guardando en cada cadena la parte correspondiente del parámetro que se determina, tal como fecha y hora, nombre, latitud, longitud y temperatura del sensor conectado.

Funcionamiento General de la plataforma

La comunicación de la plataforma con los diferentes dispositivos se puede ver en la figura 3.35 representando el envío de datos desde el equipo hacia el servidor web mediante sockets por el puerto 9595; mientras desde un Smartphone o un computador con internet se puede tener acceso a la plataforma en la IP 186.4.224.249 con el puerto 9090.

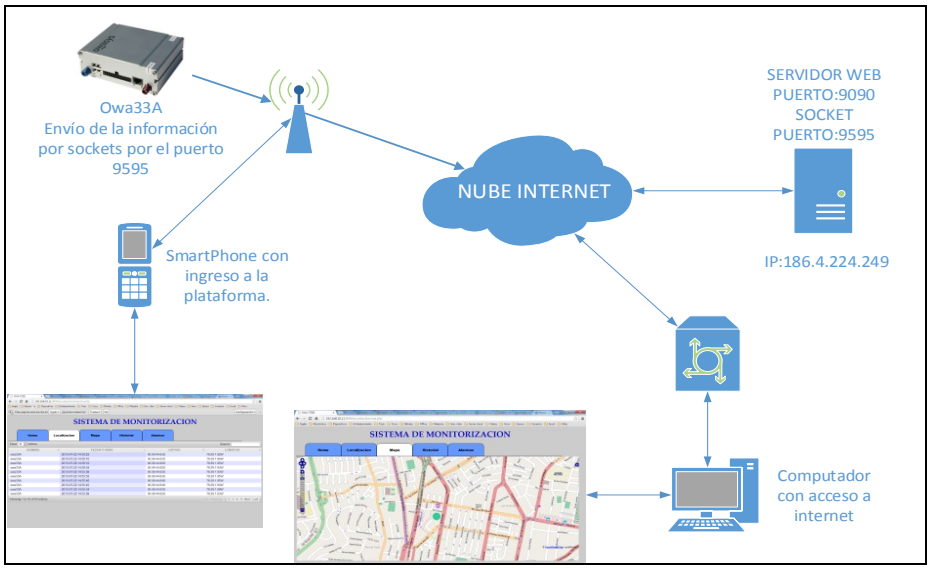


Figura 3.35. Diagrama Comunicación Plataforma

CAPÍTULO 4

PRUEBAS Y RESULTADOS EXPERIMENTALES

4.1 Funcionamiento del Equipo owa33A

El equipo owa33A es un dispositivo robusto y de tamaño pequeño capaz de permitir su ubicación en cualquier punto de una cabina de un camión o tráiler. Se debe considerar que no debe producir ningún tipo de molestia al conductor o a las personas que se encuentre en el frente del vehículo como en la figura 4.1 y pasar lo más desapercibido posible ya que en ciertos casos los conductores no deben estar al tanto del monitoreo que se realizará al vehículo.

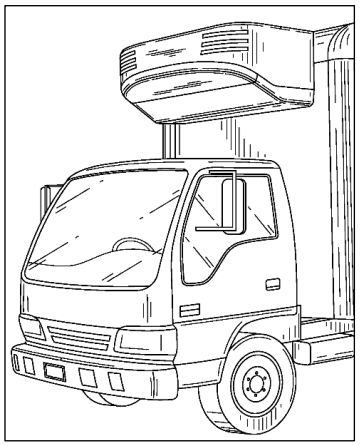


Figura 4.1 Frente del Camión de Refrigeración

El owa33A no está diseñado para ser ubicado en exteriores a diferencia de otro tipo de dispositivos OWASYS que si lo permiten; para el proyecto no se vio necesario ya que lo que se desea analizar es lo que se ocurra dentro del conteiner o camión que se montará para el monitoreo. Se debe ser muy prudente con la ubicación de las antenas de GPS y GSM magnéticas que se muestran en la figura 4.2 (a) y (b).

(a) (b)

Figura 4.2 Antenas GPS y GSM

Como se puede observar las antenas presentadas en la figura 4.2 cuentan con conectores FAKRA, los cuales son fácil de empatar con el equipo y seguros impidiendo mucha pérdida en las antenas para una excelente recepción y envío de datos. Las antenas son parte fundamental para un funcionamiento correcto del equipo y por lo tanto su ubicación es de vital importancia, ya que si el equipo no logra conectarse a la red GSM, GPRS o GPS, no logrará levantar el equipo para la aplicación deseada. Las dos antenas tienen que estar lo más descubiertas posibles, es decir, deben estar ubicadas en los exteriores del vehículo y en la parte más alta del mismo y que no sufran ningún tipo de impacto con alguna estructura.

4.2 Ubicación de Sensores

La ubicación de los sensores dentro del contenedor de alimentos es importante ya que de la manera adecuada este tomará una medida correcta de los alimentos que necesitan una temperatura determinada. El tipo de sensor que se utilizará es Pointer AR0202 como se muestra en la figura 4.3,

el cual da una respuesta analógica de la temperatura que se esté midiendo y a su vez a su cable largo de 20 metros para alcanzar a todos los productos que se encuentren en un contenedor.

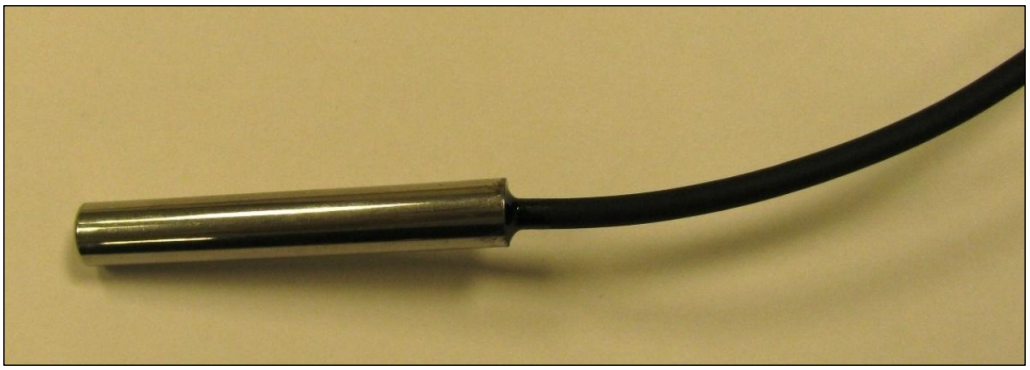


Figura 4.3 Sensor de Temperatura AR0202

Como se puede observar en la figura 4.3, es un sensor que cuenta con una parte metálica que contiene el lector y una manguera que contiene los cables para voltajes y datos, recubiertos de tal manera que no pueda pasar agua ni polvo. Por lo tanto, es un sensor muy resistente y al tener una larga longitud del cable permite monitorear el producto de una manera más exacta.

Se debe tomar en cuenta que los camiones son diseñados con paredes especiales, ya que deben mantener una temperatura interna para un determinado producto, sin ser afectada por la temperatura en el medio exterior. Además cuenta con un sistema de refrigeración, que va a producir corrientes de aire frío dentro del camión como se muestra en la figura 4.4.

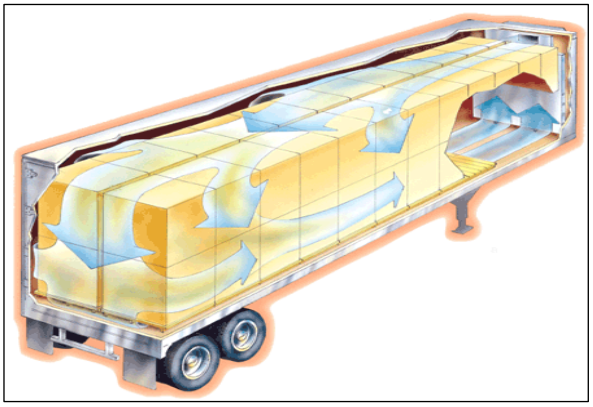


Figura 4.4 Flujo de Aire en un Camión de Refrigeración

Los camiones por lo general tienen ubicado el congelador en la parte delantera y desde allí parte el flujo de aire frío, por lo tanto el producto se ubicaría como se muestra en la figura 4.5 para dejar que el aire pase tanto por el medio y lados del camión. A su vez al colocar los alimentos sobre las plataformas de madera, permitirá que el flujo de aire frío pase por debajo.

En base a los argumentos anteriores, para determinar si el proceso de congelamiento está funcionando correctamente, tanto en la parte trasera como delantera del contenedor, se escogió colocar los sensores como se indica en la figura 4.6, obteniendo así la señal de temperatura exacta a la que deben estar los productos.

Figura 4.5 Colocación de los productos

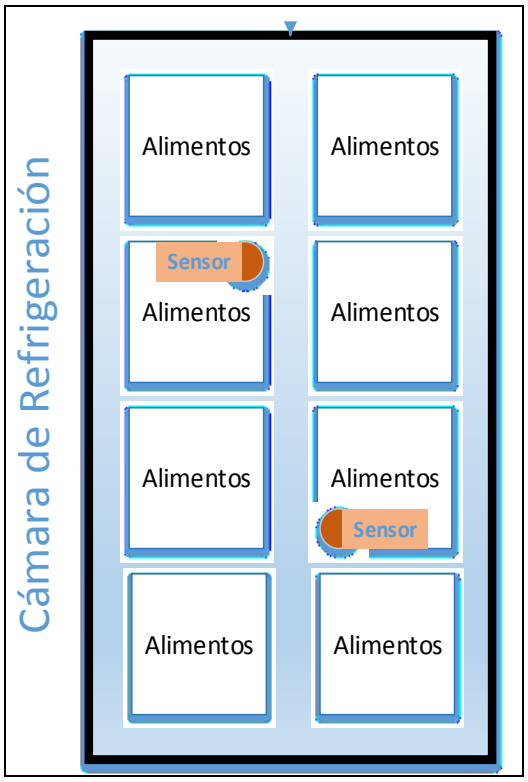


Figura 4.6 Ubicación de Sensores en el Camión

4.3 Pruebas de Laboratorio

Las pruebas de laboratorio fueron realizadas en las instalaciones de telefónica con el equipo owa33A y el simulador de Red GSM/GPRS Aligent 8960 que se muestra en la figura 4.7. Con el simulador se conecta al equipo y genera virtualmente una red de telefonía para el rango de frecuencias específico para una operadora y que para telefónica son las de 800 MHz y 1900 MHz.

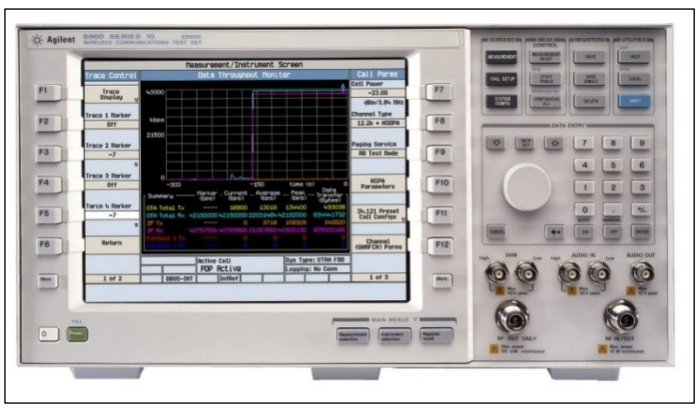


Figura 4.7 Aligent 8960

Con el simulador se realizaban 4 tipos de pruebas:

- GSM a 850 MHz
- GSM a 1900 MHz
- GPRS a 850 MHz
- GPRS a 1900 MHz

El equipo owa33A se lo debe colocar dentro de una cabina pequeña como se muestra en la figura 4.8, en la cual se realizan las conexiones internamente con el equipo para llevar por fuera de la cabina las conexiones de prueba hacia el simulador y a su vez realizar las conexiones con la computadora para almacenar en esta los datos de la simulación. El motivo de colocar el equipo dentro de la cabina es para simular que el equipo no tenga ningún tipo de pérdida y su aplicación con la señal de GSM/GPRS sea limpia.

Figura 4.8 Cabina sin Pérdidas

Al equipo owa33A se debe colocar una SIM card especial del equipo Aligent con el fin de que el equipo simule estar dentro de la red generada. Una vez listo el equipo y el simulador se procede a realizar una llamada entre los dos equipos para las pruebas de GSM a 850 MHz, 1900 MHz, obteniendo los resultados presentados en la tabla 4.1. Los resultados de GPRS a 850 y 1900 MHz son presentados en la tabla 4.2.

Tabla 4.1 Pruebas GSM

Frecuencia (MHz)	Tiempo simulación (seg)	Canal	Potencia de la celda (dBm)	Nivel de transmisión
850	178.47	224	-75	5
1900	111.10	650	-85	5

Tabla 4.2 Pruebas GPRS

Frecuencia (MHz)	Tiempo simulación (seg)	Canal	Potencia de la celda (dBm)	Nivel de transmisión
850	563.12	224 y 231	-50 a -85	5
1900	582.82	650 y 710	-85 a -99	5

En las tablas anteriores se presenta las frecuencias con el tiempo de simulación y la potencia que abarca el owa33A al estar dentro de una zona en la cual no tenga pérdidas de ningún tipo. Los canales en los cuales hace las conexiones y a su vez se puede decir que los resultados son presentados a un nivel de transmisión 5, mientras el simulador realiza una prueba desde un nivel de potencia de transmisión 1 hasta el 14.

La parte más importante es llegar a determinar el funcionamiento correcto de la red mediante un gráfico relacionando la Potencia vs el Tiempo como se muestra en la figura 4.9. En este tipo de gráfico tiene un rango establecido por el simulador para que un equipo que funcione dentro del rango, es un equipo con buen nivel de respuesta en funciones con conexión a una red de telefonía celular. En caso de que se suba el canal se puede notar que la señal va cambiando y se sobrepone en cierto momento sobre algún límite del gráfico por lo cual debe tratar de mantenerse como se muestra en la figura 4.10.

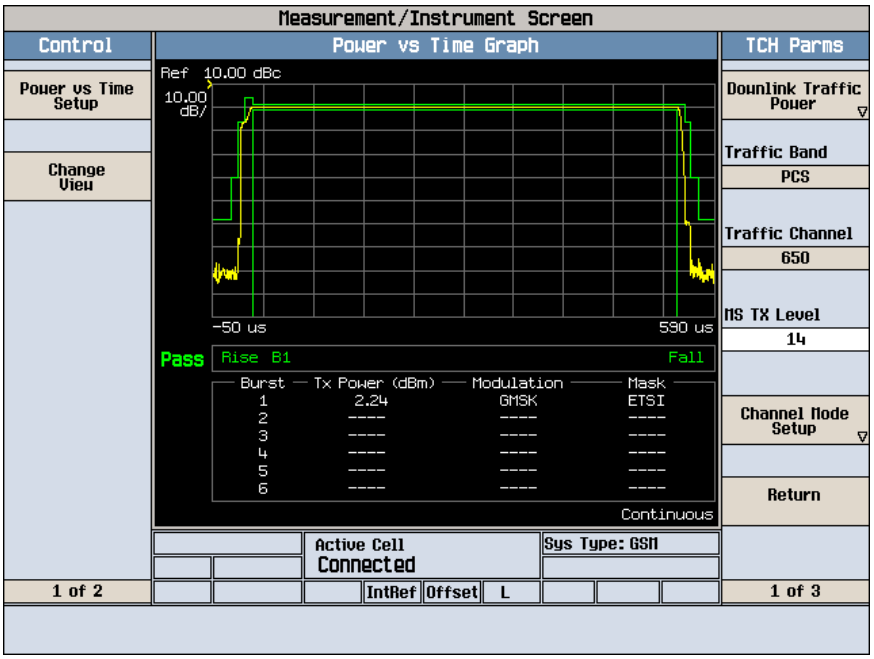


Figura 4.9 Relación Potencia vs Tiempo

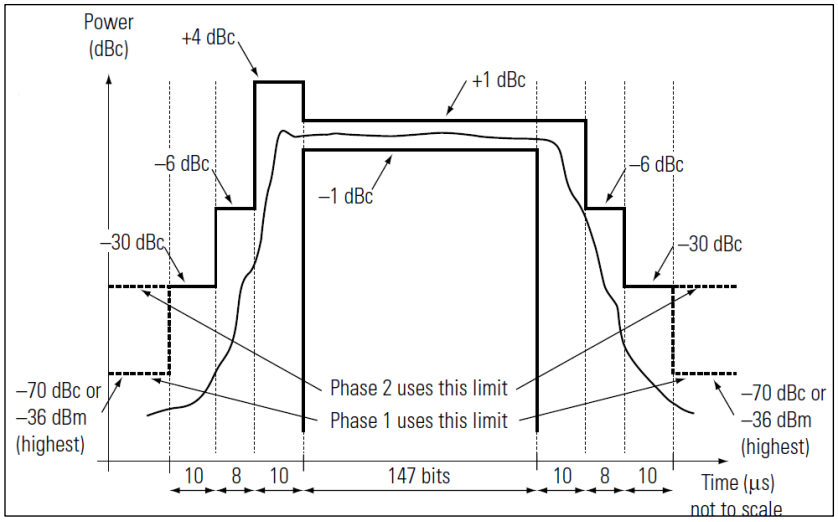


Figura 4.10 Limitaciones Potencia vs Tiempo

4.4 Pruebas de Campo

Se realizó pruebas de campo estableciendo:

- En el sector de Nayón, al norte de Quito.
- Temperatura ambiental de 19 °C.
- Cielo no despejado.

- Velocidad del vehículo de 50 a 80 Km/H.

Se estableció este Sector, ya que es el sitio idóneo escogido por el equipo de Homologación de Movistar para realizar pruebas de cobertura con respecto a cualquier dispositivo GSM/GPRS. Esta zona presenta bajo nivel de cobertura y entre más uno se aleja, más se pierde señal hasta un punto donde los teléfonos inteligentes pierden conexión tanto para GPRS como para GSM.

A diferencia de los celulares el equipo owa33A puede ser conectado con una antena de cualquier ganancia y para el proyecto se lo realizó con una antena de 3 dBi debido a tu tamaño y flexibilidad. El equipo mide los niveles de señal en RSSI (Indicador de Potencia de Señal Recibida) los cuales deben ser transformados a niveles de potencia en dBm como se presenta la tabla de conversiones para GSM en la tabla 4.3.

Tabla 4.3 Conversión RSSI a dBm

RSSI	dBm	RSSI	dBm
0	-113	14	-85
1	-111	15	-83
2	-109	16	-81
3	-107	17	-79
4	-105	18	-77
5	-103	19	-75
6	-101	20	-73
7	-99	21	-71
8	-97	22	-69
9	-95	23	-67
10	-93	24	-65
11	-91	25	-63
12	-89	26	-61
13	-87	27	-59

El equipo como mínimo presentó 21 RSSI, es decir -71 dBm y para este nivel de potencia los Smartphones Samsung Galaxy S4 y Sony Xperia P, perdieron la señal completamente sin poder recibir o realizar llamadas.

Mientras el owa33A permanecía enviando datos a través de GPRS en 2G, y en la plataforma se verificó que la recepción de datos seguía funcionando con normalidad de tal manera que se podía ubicar el equipo en el mapa.

CAPÍTULO 5

ANÁLISIS FINANCIERO

5.1 ANÁLISIS FINANCIEROACUMULATIVO

El análisis financiero del proyecto constituye la técnica matemático-financiera y analítica, cuyo objetivo es analizar los beneficios o pérdidas que se pueda tener con esta implementación, es un estudio de inversión y costos, para determinar si el proyecto es rentable o no. A continuación se presentan los indicadores utilizados para evaluar un proyecto:

- **Valor Actual Neto (VAN).-** es un procedimiento que permite calcular el valor presente de un determinado número de flujos de caja futuros, originados por una inversión dentro de un proyecto. Es decir, es igual a la diferencia entre el valor actualizado de ingresos esperados menos el valor actualizado de los costos previstos. El criterio para ver si una inversión es aconsejable o no, es que el VAN sea mayor, igual o menor que cero:

  VAN > 0, La inversión es aconsejable.
  VAN = 0, La inversión es indiferente.
  VAN < 0, La inversión no es aconsejable.

$$VAN = I_0 + \sum_{i=t} \frac{U_B}{(1+i)^t}$$

- **Periodo de Recuperación (TR).-** se define como el tiempo que se tarda en recuperar la inversión inicial. Para el cálculo del período de recuperación hay que dividir el monto invertido entre el beneficio medio anual.

- **Valor Actual de Beneficios (B/C).-** es el valor actualizado, de la diferencia entre los beneficios brutos y costos brutos futuros de un proyecto, utilizando como tasa de descuento el costo de capital invertido. O sea, mide los beneficios netos futuros, al día de hoy que se espera obtener de una inversión. Si el índice es igual o superior a la unidad, el proyecto es aceptable, este índice es el índice neto, en lugar del índice bruto:

  B/C > 1-> inversión aconsejable
  B/C < 1-> inversión desaconsejable
  B/C = 1-> inversión indiferente.

- **Tasa de Descuento.-** tasa de interés a la cual son puestas en valor presente las sumas futuras.

- **Tasa Interna de Retorno (TIR).-** se utiliza para decidir sobre la aceptación o rechazo del proyecto. Se calcula cuando el VAN=0.

  TIR > r aconsejable
  TIR = r indiferente
  TIR < r no aconsejable

5.2 ANÁLISIS DE INVERSIÓN VS SATISFACCIÓN DE CLIENTES

5.2.1 Análisis de Inversión

La inversión que se realizó para este proyecto en gran parte estuvo a cargo de la empresa Telefónica Movistar del Ecuador.

Para obtener una correcta inversión se realizó una selección minuciosa del equipo entre algunas versiones de OWASYS y también se determinó los sensores apropiados, en la siguiente tabla 5.1 se indica el costo de equipo y de accesorios utilizados para este proyecto.

Tabla 5.1 Costos de Equipo y Accesorios

ITEM	DESCRIPCIÓN	CANTIDAD	PRECIO UNITARIO	PRECIO TOTAL
1	Equipo owa33A	1	$ 420	$ 420
2	Cables	15	$ 12	$ 180
3	Conector Fakra	2	$ 2	$ 16
4	Sensor de Temperatura	1	$ 70	$ 70
5	Sensor de Magnético	1	$ 4	$ 4
6	Access Point	1	$ 32	$ 32
7	SIM card	1	$ 5	$ 5
COSTO TOTAL				$ 727

El precio total del proyecto es de $ 727 más gastos de internet para utilización de ip privada y pago de plan de datos de la SIM card.

5.2.2 Satisfacción del Cliente

Los clientes buscan brindar calidad en sus productos, al tener control en todo momento de alimentos, este proyecto será de gran ayuda para evitar pérdidas en la empresa, ya que el factor de temperatura en varios productos

de transportación es de gran importancia para mantener en buen estado la mercadería.

El equipo owa33A al ser implementado en carros de transportación y presentará una amplia funcionalidad para mantener un control en los productos a transportar, el equipo cuenta con tecnología 2G que es básicamente lo que se necesita para realizar el proyecto M2M. Al trabajar con un equipo OWASYS de tecnología 3G, sería una mala decisión ya que el precio es mayor, y realmente no es necesario para este tipo de implementación.

5.3 CÁLCULO DE VIABILIDAD DEL PROYECTO

Tabla 5.2 Estudio Financiero

ESTUDIO FINANCIERO		
SUPUESTOS		
Venta Equipo:	$	3.200,00
Cobro Mensual Servicios:	$	22,00
Cobro Anual Servicios:	$	264,00
Costo de Equipo:	$	727,00
Costo Anual Internet:	$	504,00
Costo Anual Plan Datos:	$	384,00
Años	**# Equipos**	
1er año	30	
2do año	15	
3er año	40	
4to año	**80**	
5to año	100	

Continúa ⟶

	1	2	3	4	5
INVERSIÓN INICIAL: -10000					
INGRESO:					
Cobro Equipo	$ 96.000,00	$ 48.000,00	$ 128.000,00	$ 256.000,00	$ 320.000,00
Cobro Servicio	$ 7.920,00	$ 3.960,00	$ 10.560,00	$ 21.120,00	$ 26.400,00
Tot. Ingresos	$ 103.920,00	$ 51.960,00	$ 138.560,00	$ 277.120,00	$ 346.400,00
EGRESO:					
Gasto Equipo	$ 21.810,00	$ 10.905,00	$ 29.080,00	$ 58.160,00	$ 72.700,00
Gasto Internet IP Pública	$ 15.120,00	$ 7.560,00	$ 20.160,00	$ 40.320,00	$ 50.400,00
Gasto Plan de Datos	$ 11.520,00	$ 5.760,00	$ 15.360,00	$ 30.720,00	$ 38.400,00
Tot. Egresos	$ 48.450,00	$ 24.225,00	$ 64.600,00	$ 129.200,00	$ 161.500,00
INGRESO/EGRESO BRUTO:	**$ 55.470,00**	**$ 27.735,00**	**$ 73.960,00**	**$ 147.920,00**	**$ 184.900,00**
Tasa de Descuento:	15%				
VAN	$ 284.338,05				
BENEFICIO MEDIO ACTUAL:	$ 97.997,00				
TR	$ (0,10)				

Luego de analizar las proyecciones a 5 años, para este proyecto el valor actual neto es VAN>0, por lo que se puede decir que la inversión es aconsejable.

CAPÍTULO 6

CONCLUSIONES Y RECOMENDACIONES

6.1 CONCLUSIONES

- Se realizó el diseño e implementación de un sistema de monitorización para cámaras de refrigeración en transporte de alimentos perecederos utilizando la red GPRS, el cual después de ser probado cumplió con los objetivos planteados y permitió obtener un conocimiento actualizado de las comunicaciones M2M.

- Se desarrolló una aplicación conformada por módulos de entradas y salidas digitales, rtu, GPS, GSM y GPRS, que trabajan en conjunto y son compiladas en Linux mediante un compilador cruzado, el cual está instalado en el owa33A y permite que se ejecuten correctamente para el desarrollo del proyecto.

- Se configuró una base de datos en un servidor web para almacenar la información de temperatura, ubicación, movimiento obtenida de la gestión de comunicación entre los sensores y el equipo.

- Se comprobó con pruebas realizadas en laboratorio que la conexión que se establece entre el owa33A y la plataforma es confiable y está dentro de los rangos establecidos en la normativa de Potencia vs Tiempo para los 14 canales en los que trabaja el operador móvil.

- Mediante pruebas realizadas en campo se logró determinar que el proyecto es fiable ya que realiza una sincronización inmediata entre el equipo y la plataforma permitiendo una transmisión de datos continua y que se garantiza por la antena de 3dBi para una mejor conexión con la estación base del operador.

- Se concluyó que la comunicación de datos vía GPRS en 2G es muy útil y eficiente para nuestro entorno a diferencia de 3G, puesto que en 2G se puede abarcar una mayor cobertura que para el proyecto es necesario por la necesidad de localizar los vehículos y debido a que la transmisión de datos es mínima.

6.2 RECOMENDACIONES

- Se recomienda trabajar con plataformas en la nube para evitar la programación en su totalidad.

- Para elegir el sensor de temperatura, es recomendable basarse en la sensibilidad de respuesta que posea, para este proyecto se trabajó con el ar0204 ya que se estudió y es suficiente para implementarlo en este proyecto.

- En la instalación de los cables en el camión, se recomienda que tengan una cobertura de material especial, para que soporten cualquier temperatura a la que se encuentre la cabina del camión.

- El owa33A en la instalación, es preferible que se lo instale en el cabezal del camión por motivo de cobertura de las antenas GSM y GPS.

- Empezar realizando pruebas de las antenas GPS y GSM con las que se trabaja en el equipo owa33A, para asegurarse del correcto estado de funcionamiento de estas.

REFERENCIAS BIBLIOGRÁFICAS

Arian. (1989). *Arian*. Obtenido de http://www.arian.cl/downloads/nt-002.pdf

Ávila, P. E. (2011). *ESPE*. Obtenido de http://repositorio.espe.edu.ec/bitstream/21000/2639/1/T-ESPE-030150.PDF

Bastidas, L. F. (2006). *ESPE*. Obtenido de http://repositorio.espe.edu.ec/bitstream/21000/660/1/T-ESPE-014263.pdf

Carletti, E. J. (2012). *Robots*. Obtenido de Robots: http://robots-argentina.com.ar/Sensores_general.htm

Carpross. (2009). *Carpross*. Obtenido de Carpross: http://www.carpross.com.mx/gps/tecnologiagsm.html

Centeno, D. M. (2010). *TAV*. Obtenido de TAV: http://www.tav.net/transductores/acelerometros-sensores-piezoelectricos.pdf

Cristian Gonzalo Karolis, D. F. (2009). *ESPE*. Obtenido de ESPE: http://repositorio.espe.edu.ec/bitstream/21000/470/1/T-ESPE-024461.pdf

David Alejandro Cerda Sánchez, I. P. (2011). *ESPE*. Obtenido de http://repositorio.espe.edu.ec/bitstream/21000/3784/1/T-ESPEL-0800.pdf

FCEFyN. (2003). *FCEFyN*. Obtenido de FCEFyN: http://www.efn.unc.edu.ar/departamentos/electro/cat/eye_archivos/apuntes/a_practico/Cap%206%20Pco.pdf

Hidalgo, J. R. (2003). *EROSKY Consumer*. Obtenido de EROSKY Consumer: http://www.consumer.es/seguridad-alimentaria/normativa-legal/2003/12/22/10013.php

Infraestructura Digital. (2012). *Infraestructura Digital*. Obtenido de Infraestructura Digital.

Iocom Ltda. (1999). *IOCOM*. Obtenido de IOCOM: http://iocom.com.co/docs/Telemetria_Iocom.pdf

Jesús Bausà Aragonés, C. G. (2003). *UNIVERSIDAD POLITÉCNICA DE VALENCIA*. Obtenido de UNIVERSIDAD POLITÉCNICA DE

VALENCIA: http://server-die.alc.upv.es/asignaturas/LSED/2003-04/0.Sens_Temp/ARCHIVOS/SensoresTemperatura.pdf

Jhony Polibio Pozo Chaves, W. P. (2010). *ESPE*. Obtenido de http://repositorio.espe.edu.ec/bitstream/21000/2940/1/T-ESPEL-0705.pdf

López, E. P. (2007). *Wikidot*. Obtenido de http://linuxemb.wikidot.com/tesis-c2

Madrid Salud. (2012). *Madrid Salud*. Obtenido de Madrid Salud: http://www.madridsalud.es/temas/transporte_alimentos.php

Rosero, L. D. (2011). *ESPE*. Obtenido de http://repositorio.espe.edu.ec/bitstream/21000/3316/1/T-ESPE-031145.pdf

SAPIENSMAN. (2006). *SAPIENSMAN*. Obtenido de SAPIENSMAN: http://www.sapiensman.com/medicion_de_temperatura/

Universidad de Valencia. (2003). *Universidad de Valencia*. Obtenido de http://server-die.alc.upv.es/asignaturas/LSED/2003-04/0.Sens_Temp/Clasify/Termorresistencias.htm

Universidad de Valencia. (2003). *Universidad de Valencia*. Obtenido de Universidad de Valencia: http://server-die.alc.upv.es/asignaturas/LSED/2003-04/0.Sens_Temp/Clasify/Termocuplas.htm

Universidad de Valencia. (2003). *Universidad de Valencia*. Obtenido de Universidad de Valencia: http://server-die.alc.upv.es/asignaturas/LSED/2003-04/0.Sens_Temp/Clasify/Termorresistencias.htm

Universidad de Valencia. (2003). *Universidad de Valencia*. Obtenido de Universidad de Valencia: http://server-die.alc.upv.es/asignaturas/LSED/2003-04/0.Sens_Temp/Clasify/Termistores.htm

ANEXOS

ANEXO 1

A. Archivos de programa de C

Test_module.c

```
#define __TEST_Module_C
//-----------------------------------------------------------------//
//System Includes
//-----------------------------------------------------------------//
#include <stdio.h>
#include <signal.h>
#include <errno.h>
#include <ctype.h>
#include <unistd.h>
//-----------------------------------------------------------------//
//User Includes
//-----------------------------------------------------------------//
#include "LoadLibs.h"
#include "Test_Module.h"
#include "rtu_module.h"
#include "io_module.h"
#include "gps_module.h"
#include "gsm_module.h"
#include "inet_module.h"
#include "owa3x/owerrors.h"
#include "owa3x/owcomdefs.h"
#include "owa3x/GPS_ModuleDefs.h"
#include "ConfigurationFile.h"
//-----------------------------------------------------------------//
//Private variables
//-----------------------------------------------------------------//
// File descriptor for logging.
FILE *logfd = NULL;
/* functions definition */
//-----------------------------------------------------------------//
// Function: main()
// Input Params:
```

```
// Output Params:
// Description:
//---------------------------------------------------------------//
int main(int argc, char *argv[])
{
  INT                   ReturnCode = 0;
  BOOL                        terminate=FALSE;
  //CHAR                      keyEntry[255];
  //CHAR                      MessageToSend[ BUFFER_SIZE];
  TGPS_POS          CurCoords;
  TGPS_SPEED                  CurSpeed;
  CHAR  PLMN[ 7];
  CHAR GsmImsi[ 17];
  CHAR buffer_ip[MAX_BUFFER_SIZE];
  CHAR  Buffer[ EVENTS_BUFFER_SIZE];
  CHAR  MessageToSend[ BUFFER_SIZE];
  INT  bandera=0;
  CHAR SMSON[]="ON";
  CHAR SMSOFF[]="OFF";
  CHAR PRI[7];
  INT COMPA;
  INT SMSIndex;
  SMS_s IncomingSMS;

 ReadConfigurationFile(APP_CONF, Settings);

  printf( "\n OWASYS -> Starting Test_Module %s (%s, %s)\n", APP_VERSION,
__DATE__ , __TIME__);
  if (InitRTUModule() != NO_ERROR)
    exit(EXIT_FAILURE);
  if (InitIOModule() != NO_ERROR)
    exit(EXIT_FAILURE);
  if (InitGPSModule() != NO_ERROR)
    exit(EXIT_FAILURE);
  if (InitGSMModule() != NO_ERROR)
    exit(EXIT_FAILURE);
  GSMImsi( GsmImsi, 16);
  memset( PLMN, 0, 7);
  strncpy( PLMN, ( CHAR*) GsmImsi, 5);
  ReadGPRSConfigurationFile( GPRS_NET, GPRSSettings, PLMN);
  printf("\n \n");
  if (InitModINET( gprsUser, gprsPass, gprsDNS1, gprsDNS2, gprsAPN) !=
NO_ERROR)
    exit(EXIT_FAILURE);
  greenled(LEDOFF,0);
  redled(LEDOFF, 0);
  sem_init( &Events, 0, 0);
  printf("\n");
  GetSystemTime();
  printf("\n");
```

```
TomaMovimiento();
  printf("\n");
  Config_Interrupts();
printf("\n");
  while(!terminate) {
    Wait(ONE_SECOND * 4,0);
printf("Los Datos son: \n");
GetGPSPosition(&CurCoords , &CurSpeed);
    //printf("%s y %s.\n",CurCoords, CurSpeed); %02d
                        printf( "\n-----------------------------------------------------\n");
                        printf ("LATITUDE --> %02d degrees %02d minutes
%04.04f seconds %c\n",

        CurCoords.Latitude.Degrees, CurCoords.Latitude.Minutes,

        CurCoords.Latitude.Seconds, CurCoords.Latitude.Dir);
                        printf ("LONGITUDE --> %03d degrees %02d minutes
%04.04f seconds %c\n",

        CurCoords.Longitude.Degrees, CurCoords.Longitude.Minutes,

        CurCoords.Longitude.Seconds, CurCoords.Longitude.Dir);
                        //printf( "-----------------------------------------------------\n");
                        //printf( "\n-------------------------------------------\n");
                        printf ("speed in knots = %04.04f\n",
CurSpeed.speed_in_knots);
                        printf ("speed in kmh = %04.04f\n", CurSpeed.speed_in_kmh);
                        printf( "-------------------------------------------\n");
        printf("\n");
        GetVin();
        GetVbat();
        GetTemp();
        printf("\n");
        GetIP( buffer_ip);
        printf( "-------------------------------------------\n");
        printf ("IP address -> %s\n", buffer_ip);
        printf( "-------------------------------------------\n");
        if( IsThereGSMEvent( Buffer) == SMS_EVENT){
            SMSIndex = atoi( Buffer);
            printf( "SMSIndex %d\n", SMSIndex);
            if( ReadSMS( SMSIndex, &IncomingSMS) == NO_ERROR){
              //ParseSMS( SMSCommands, IncomingSMS);
              //redled(LEDON, 1);
                    printf("Mensaje: %s\n",(&IncomingSMS)->owBody);
                    strcpy((CHAR*)PRI, (CHAR*)((&IncomingSMS)->owBody));
                    COMPA=strncmp(PRI,SMSON,2);
                    //printf("COMPA:%d,                                    PRI:%s,
SMSON:%s\n",COMPA,PRI,SMSON);
                    if(COMPA==0){
                      redled(LEDON, 15);
```

```
                    Wait(ONE_SECOND * 4,0);
                printf("\nLlego mensajes con: ON\n");
                }
                    COMPA=strncmp(PRI,SMSOFF,2);
                    //printf("COMPA:%d,                                    PRI:%s,
SMSON:%s\n",COMPA,PRI,SMSON);
                    if(COMPA== 0){
                      redled(LEDON, 2);
                      Wait(ONE_SECOND * 4,0);
                printf("\nLlego mensajes con: OFF\n");
                }
                  }
        }else{
          redled(LEDOFF, 0);
        }
        //Blank Line..
      memset( MessageToSend, ' ', 80);
      MessageToSend[ 0] = '\r';
      write( fileno( stderr), MessageToSend, 80);
        ComposeMessage(MessageToSend);
        printf( "\n%s\n", MessageToSend);
        if(bandera > 2){
          CreateTCPServerSocket((CHAR *)IPaddr, dest_port);
          SendMessage( MessageToSend);
        }
        /*if(bandera == 10){
          printf( "\n%s\n", MessageToSend);
        SendMessage( MessageToSend);
        }*/
        bandera++;
  }
  //-------------------------------------------------------------------------
  sem_post    ( &Events);
  sem_destroy ( &Events);
 greenled(LEDOFF,0);
  redled(LEDOFF, 0);
 FinalizeApplication = TRUE;
  if( EndModINET() != NO_ERROR) {
    WriteLog("Error %d in EndINETModule()", ReturnCode);
  }
  if( EndGSMModule() != NO_ERROR) {
    WriteLog("Error %d in EndGSMModule()", ReturnCode);
  }
  if( EndGPSModule() != NO_ERROR) {
    WriteLog("Error %d in EndGPSModule()", ReturnCode);
  }
  if( EndRTUModule() != NO_ERROR) {
    WriteLog("Error %d in EndRTUModule()", ReturnCode);
  }
//IOs MUST BE the last to finalize because it closes the ios driver.
```

```
  if( EndIOModule() != NO_ERROR) {
    WriteLog("Error %d in EndIOModule()", ReturnCode);
  }
  exit(EXIT_SUCCESS);
}
//--------------------------------------------------------------------//
// Function: CalculateBinaryChecksum()
// Input Params:
//              buffer with a binary message
//              buffer lenght

// Output Params:
//              calculated checksum
// Description:
//   Calculates the checksum of a binary Ublox message.
//--------------------------------------------------------------------//
INT CalculateBinaryChecksum( CHAR *Buff, INT len )
{
  INT   n;
  CHAR  CK_A=0, CK_B=0;

  for (n=0; n < len; n++) {   // Calculate checksum.
                  CK_A += *(Buff+n);
                  CK_B += CK_A;
        }
        return ((CK_A<<8) | CK_B);
}
//--------------------------------------------------------------------//
// Function: getEntry()
// Input Params:
//             none
// Output Params:
//             strEntry: pointer to the read string

// Description:
//   Reads input characters.
//--------------------------------------------------------------------//
VOID getEntry( CHAR *strEntry )
{
  CHAR c = 0;
  INT  i = 0;
  while( ( c = getchar()) != 0x0a){
    if( c == 0x0FF){
      strEntry [ i] = 0;
      return;
    }
    if( c == 0x08){ //BackSpace
      if( i>0)
        i--;
      else
```

```
            i = 0;
        } else
            strEntry[ i++] = c;
    }
    strEntry [ i] = 0;
    if ( strEntry[ i-1] == 0x0a)
        strEntry[ i-1] = 0;
}
//-------------------------------------------------------------------//
// Function: WriteLog()
// Input Params:
// Output Params:
// Description:
//-------------------------------------------------------------------//
VOID WriteLog( const char *format, ... )
{
        va_list arg_ptr;
        char    LineBuffer [512];
        // Get the system date/time
        time_t tmNow;
        tmNow = GetCurTime(NULL);
        stmNow = localtime(&tmNow);
        // Put the current date/time in front of the logline
        sprintf(LineBuffer, "%04d%02d%02d-%02d%02d%02d  ",
                        stmNow->tm_year + 1900, stmNow->tm_mon+1,
stmNow->tm_mday,
                        stmNow->tm_hour, stmNow->tm_min, stmNow-
>tm_sec);
        va_start(arg_ptr, format);
        vsprintf(LineBuffer + strlen (LineBuffer), format, arg_ptr);
        va_end(arg_ptr);
        // Ensure newline
        if(!strchr(LineBuffer, '\n')) {
                strcat(LineBuffer, "\n");
        }
#ifdef DEBUG_TRACES
        // Print it on screen, and in logfile (when available)
        printf(LineBuffer);
        if (logfd != NULL) {
                fwrite(LineBuffer, 1, strlen(LineBuffer), logfd);
                fflush(logfd);
        }
#endif
}
//-------------------------------------------------------------------//
// Function: time_t GetCurTime(time_t *p)
// Input Params:
        //      pointer to time_t, IN/OUT. gets current time
// Returns:
//      time_t: return current time
```

```
// Description:
//        Get current time (replacement for time_t time(time_t*)
//-------------------------------------------------------------------//
time_t GetCurTime( time_t *p )
{
        time_t      tmNow;
        if(p) {
                tmNow = time(p);
        }
        else {
                tmNow = time(NULL);
        }
        return tmNow;
}
//-------------------------------------------------------------------//
// Function: ComposeMessage()
// Input Params:
// Output Params:
//    *Message : string with the message.
// Description:
//    Creates the message with the available data.
//-------------------------------------------------------------------//
VOID ComposeMessage( CHAR *Message)
{
  TSYSTEM_TIME   system_time;
  INT          ReturnCode,retVal=0,value=0,valor;
  TGPS_POS      Pos;
  TGPS_SPEED    Speed;
  FILE          *fptr;
  if( ( ReturnCode = ( *FncGetSystemTime) ( &system_time)) != NO_ERROR){
    printf("GSM->Error on SetSystemTime %d\n", ReturnCode);
  }
  retVal = (*FncANAGIO_GetAnalogIn)( 0, &value);
  printf(" ANALOG INPUT 0 = %d\n", value);
    if( retVal == NO_ERROR ) {
      printf(" ANALOG INPUT 0 = %d\n", value);
    } else {
      printf(" Error %d reading ANALOG INPUT 0\n", retVal);
    }
  valor=(125*value)/(1024*2.5);
  GetGPSPosition( &Pos, &Speed);
  memset( Message, 0, BUFFER_SIZE);
  if( Pos.DataValid){
//      printf("Data valid\n");
    sprintf( Message,MESSAGE_GPS_DATA,
              owa3x_id,
              system_time.Year,
              system_time.Month,
              system_time.Day,
              ((system_time.Hour)-5),
```

```
                system_time.Minute,
                system_time.Second,
                Pos.Latitude.Degrees,
                Pos.Latitude.Minutes,
                Pos.Latitude.Seconds,

                Pos.Latitude.Dir,
                Pos.Longitude.Degrees,
                Pos.Longitude.Minutes,
                Pos.Longitude.Seconds,
                Pos.Longitude.Dir,
                Speed.speed_in_kmh,
                      valor);
  } else {
//      printf( "Position not available\n");
     sprintf( Message, MESSAGE_NO_GPS_DATA,
                owa3x_id,
                system_time.Year,
                system_time.Month,
                system_time.Day,
                ((system_time.Hour)-5),
                system_time.Minute,
                system_time.Second);
  }
  if( ( fptr = fopen( POS_FILE, "a+"))){
    fprintf( fptr, "%s\n\n", Message);
    fclose( fptr);
  }
}
//-----------------------------------------------------------------//
// Function: SendMessage()
// Input Params:
//    *Message : string with the message to send.
// Output Params:
// Description:
//    Sends the message using TCP or UDP socket.
//-----------------------------------------------------------------//
VOID SendMessage( CHAR *Message )
{
  INT  ReturnCode;
  //CHAR mensaje[]="PROBANDO";
ReturnCode = TCP_SendData((CHAR *)IPaddr, dest_port, Message,
strlen(Message));
  if(RemoteServerDisconnected || (ReturnCode == -1)) {
    //terminate=TRUE;
  }else {
    WriteLog("TCP-> All data are sent");
  }
}
//-----------------------------------------------------------------//
```

```
// Function: ComposeMessage()
// Input Params:
// Output Params:
//    *Message : string with the message.
// Description:
//    Creates the message with the available data.
//-------------------------------------------------------------------//
VOID ComposeMessageALA( CHAR *Message, INT alarm)
{
  TSYSTEM_TIME   system_time;
  INT          ReturnCode;
  FILE          *fptr;

  if( ( ReturnCode = ( *FncGetSystemTime) ( &system_time)) != NO_ERROR){
    printf("GSM->Error on SetSystemTime %d\n", ReturnCode);
  }
   //printf("Alarma: %d",alarm);
    memset( Message, 0, BUFFER_SIZE);
  if( alarm==0){
//      printf("Data valid\n");
    sprintf( Message,MESSAGE_ALAM_DATA,
              owa3x_id,
              system_time.Year,
              system_time.Month,
              system_time.Day,
              ((system_time.Hour)-5),
              system_time.Minute,
              system_time.Second);
  }
  if( alarm==1){
//      printf("Data valid\n");
    sprintf( Message,MESSAGE_ALAD_DATA,
              owa3x_id,
              system_time.Year,
              system_time.Month,
              system_time.Day,
              ((system_time.Hour)-5),
              system_time.Minute,
              system_time.Second);
  }
  printf("Menssage: %s",Message);
  if( ( fptr = fopen( ALA_FILE, "a+"))){
    fprintf( fptr, "%s\n\n", Message);
    fclose( fptr);
  }
}
```

Test_Module.h

```
#ifndef __Test_Module_H
  #define __Test_Module_H
//---------------------------------------------------------------------//
//System Includes
//---------------------------------------------------------------------//
#include <stdlib.h>
#include <dlfcn.h>
#include <pthread.h>
#include <semaphore.h>
#include <owa3x/INET_ModuleDefs.h>
#include "owa3x/owcomdefs.h"
#include "owa3x/GPS_ModuleDefs.h"
#include "owa3x/RTUControlDefs.h"
#include "owa3x/owa31/IOs_ModuleDefs.h"
#include "ConfigurationFile.h"
#include "socket.h"
//---------------------------------------------------------------------//
//Macros
//---------------------------------------------------------------------//
#ifdef DEBUG_TRACES
  #define DEBUG_(x) x
#endif
#define RES_(x) x
//   #define RES_(x)   // No printfs...
//---------------------------------------------------------------------//
//Defines
//---------------------------------------------------------------------//
#define APP_VERSION   "P1D"
#define CMD_QUIT                "quit\0"
   #define LEDON  1
   #define LEDOFF 2
  #define MAX_SIZE          100
  #define MAX_BUFFER_SIZE    255
  #define BUFFER_SIZE       1024
  #define       MESSAGE_GPS_DATA                        "G|%s|%04ld/%02d/%02d
%02d:%02d:%02d|%02d.%02d %02.02f%c,%02d.%02d %02.02f%c|%03.01f|%d"
  //#define    MESSAGE_GPS_DATA              "%s    -Time:    %04ld/%02d/%02d
%02d:%02d:%02d\nGPS         position:         lat         %02d.%02d'%02.02f-%c,long
%02d.%02d'%02.02f-%c\nGPS speed:%03.01f km/h"
  #define      MESSAGE_NO_GPS_DATA      "%s      -Time:      %04ld/%02d/%02d
%02d:%02d:%02d"
  #define       MESSAGE_ALAM_DATA                       "A|%s|%04ld/%02d/%02d
%02d:%02d:%02d|MOVIMIENTO"
#define       MESSAGE_ALAD_DATA                       "A|%s|%04ld/%02d/%02d
%02d:%02d:%02d|APERTURA DE PUERTA"
#define  POS_FILE       "/home/Positions"
  #define  ALA_FILE       "/home/Alarmas"
```

```
#define DISTANCE_TRIGGER   200   //Distance in METERS
#define  APP_CONF       "/home/owa.ini"
#define  GPRS_NET       "/home/gprs.net"
#define  TAG_PIN        "SIM_PIN"
#define  TAG_ME         "ME"
 //#define  TAG_TIMER      "TIMER_DATA_REPORT"
#define  TAG_IPADDR     "DATA_DESTINATION_IP"
#define  TAG_DEST_PORT  "DATA_DESTINATION_PORT"
#define  TAG_RECV_PORT  "DATA_RX_PORT"
#define  TAG_ID         "OWA3X_ID"
#define  TAG_NUM        "NUMERO"
#define  TAG_BODY       "BODY"
//GPRS TAGS
#define  TAG_USER       "GPRS_USER"
#define  TAG_PWD        "GPRS_PASSWORD"
#define  TAG_DNS1       "GPRS_DNS1"
#define  TAG_DNS2       "GPRS_DNS2"
#define  TAG_APN        "GPRS_APN"
#define TAG_TIME_SYNCH     "TIMESYNCH"
#define TAG_END_APP        "ENDAPPLICATION"
#define TAG_PING_APP       "AREYOUTHERE?"
#define TAG_GETPOSITION    "GETPOSITION"
#define SMS_FILE           "/home/InSMSs"
#define SMS_PRE  0
VOID  ComposeMessage    ( CHAR *Message);
VOID  SendMessage       ( CHAR *Message);
VOID ComposeMessageALA  ( CHAR *Message,INT alarm);
#ifdef __TEST_Module_C
  int main(int argc, char *argv[]);
  VOID getEntry( CHAR* strEntry );
  VOID WriteLog( const char *format, ... );
  time_t GetCurTime( time_t *p );
  INT CalculateBinaryChecksum( CHAR *Buff, INT len );
      CHAR                          RXBuffer[3000];
      CHAR                          TXBuffer[256];
      tANTENNA_NEW_STATUS           AnStatus;
      TUTC_DATE_TIME                CurDateTime;
  tUTM_COORDINATES          utm;
  tGEODETIC_COORDINATES     GeoCoord;
  TANTENNA_STATUS           AnStatusOld;
  tECEF_COORDINATES         ECEFCoord;
  extern VOID greenled( INT onoff, INT freq );
  extern VOID redled( INT onoff, INT freq );
  CHAR  sim_pin   [ 9];
  CHAR  me [ 16];
  UCHAR IPaddr    [ MAX_IP_SIZE];//Remote Server address
  INT   dest_port;            //Remote server port
  INT   recv_port;            //Local rx port
  CHAR  owa3x_id   [ 20];      //id
  CHAR  num_alerta   [ 11];
```

```
    CHAR  body_alerta   [ 25];
    Parameters_t Settings[] = {
      { ( void*)&sim_pin      , 9              , 0          , STRING_TYPE  , TAG_PIN},
      { ( void*)&me           , 16             , 0          , STRING_TYPE  ,
TAG_ME},
      { ( void*)&owa3x_id     , 20             , 0          , STRING_TYPE  ,
TAG_ID},
      { ( void*)&num_alerta   , 11             , 0          , STRING_TYPE  ,
TAG_NUM},
      { ( void*)&body_alerta  , 25             , 0          , STRING_TYPE  ,
TAG_BODY},
      { ( void*)&IPaddr       , MAX_IP_SIZE    , 0          , STRING_TYPE  ,
TAG_IPADDR},
      { ( void*)&dest_port    , sizeof( dest_port) , 0      , INT_TYPE     ,
TAG_DEST_PORT},
      { ( void*)&recv_port    , sizeof( recv_port) , 0      , INT_TYPE     ,
TAG_RECV_PORT},
      { ( void*)NULL          , 0              , 0          , 0            , ""}
    };
    //GPRS Configuration info
    CHAR gprsUser  [ MAX_USER_SIZE];
    CHAR gprsAPN   [ MAX_APN_SIZE];
    CHAR gprsDNS1  [ MAX_IP_SIZE];
    CHAR gprsDNS2  [ MAX_IP_SIZE];
    CHAR gprsPass  [ MAX_PWD_SIZE];
    Parameters_t GPRSSettings[ ]={
      { ( void*)&gprsUser   , MAX_USER_SIZE    , 0          , STRING_TYPE  ,
TAG_USER},
      { ( void*)&gprsAPN    , MAX_APN_SIZE     , 0          , STRING_TYPE  ,
TAG_APN},
      { ( void*)&gprsDNS1   , MAX_IP_SIZE      , 0          , STRING_TYPE  ,
TAG_DNS1},
      { ( void*)&gprsDNS2   , MAX_IP_SIZE      , 0          , STRING_TYPE  ,
TAG_DNS2},
      { ( void*)&gprsPass   , MAX_PWD_SIZE     , 0          , STRING_TYPE  ,
TAG_PWD},
      { ( void*)NULL        , 0              , 0          , 0            , ""}
    };
   /* //SMS Parser Configuration
    StringParser_t SMSCommands[]={
      { TAG_TIME_SYNCH  , &TimeSynchReceived},
      { TAG_END_APP     , &EndApplication},
      { TAG_PING_APP    , &PingApplication},
      { TAG_GETPOSITION , &GetPosApplication},
      { "\0",NULL}
    };*/
    #define  PERIODIC_POS     0x01
    #define  PERIODIC_SPEED   0x02
    #define  PERIODIC_NMEA    0x04
    sem_t                 Events;
```

```
  #else
    extern VOID WriteLog( const char *format, ... );
    extern CHAR  sim_pin   [ 9];
    extern CHAR  me [ 16];
    extern CHAR  owa3x_id   [ 20];      //id
    extern CHAR  num_alerta   [ 11];
    extern CHAR  body_alerta   [ 25];
    extern sem_t        Events;
    extern UCHAR IPaddr    [ MAX_IP_SIZE];//Remote Server address
    extern INT   dest_port;             //Remote server port
  #endif
#endif
```

Ios_module.c

```
#define __IO_MODULE_C
//-------------------------------------------------------------------//
//System Includes
//-------------------------------------------------------------------//
#include <stdio.h>
//-------------------------------------------------------------------//
//User Includes
//-------------------------------------------------------------------//
#include "LoadLibs.h"
#include "io_module.h"
#include "Test_Module.h"
//-------------------------------------------------------------------//
// Function: InitIOModule()
// Input Params:
// Output Params:
// Description:
//-------------------------------------------------------------------//
INT InitIOModule( VOID )
{
  INT                   ReturnCode;
  // load IOs. library and its functions
  LoadExternalLibrary((char *)LIBIO, &LibGPIOHandle);
  LoadIOsGenericFunctions(LibGPIOHandle);
        // configure and start IOsModule
  if( ( ReturnCode = (*FncIO_Initialize)( )) != NO_ERROR) {
    WriteLog("Error %d in IO_Initialize()", ReturnCode);
                UnloadExternalLibrary(LibGPIOHandle);
        return -1;
  }
        if( ( ReturnCode = (*FncIO_Start)()) != NO_ERROR) {
    WriteLog("Error %d in IO_Start()", ReturnCode);
                (*FncIO_Finalize)();
                UnloadExternalLibrary(LibGPIOHandle);
```

```
                return -1;
  }
  printf("\nIOs-> Module initialized & started");
  return NO_ERROR;
}
//-----------------------------------------------------------------//
// Function: EndModIO()
// Input Params:
// Output Params:
// Description:
//-----------------------------------------------------------------//
INT EndIOModule( VOID )
{
  (*FncIO_Finalize)();
  UnloadExternalLibrary(LibGPIOHandle);
  WriteLog("IO->Module finalized");
  return NO_ERROR;
}
//-----------------------------------------------------------------//
// Function: LoadIOsGenericFunctions()
// Input Params:
// Output Params:
// Description:
//-----------------------------------------------------------------//
VOID LoadIOsGenericFunctions( VOID *wLibHandle)
{
  FncIO_Initialize = ( INT ( *) ( VOID )) dlsym( wLibHandle, "IO_Initialize");
  if( dlerror() != NULL)  {
    WriteLog("No IO_Initialize() found");
  }
  FncIO_Start = ( INT ( *) ( VOID)) dlsym( wLibHandle, "IO_Start");
  if( dlerror() != NULL)  {
    WriteLog("No IO_Start() found");
  }
       FncIO_Finalize = ( INT ( *) ( VOID)) dlsym( wLibHandle, "IO_Finalize");
  if( dlerror() != NULL)  {
    WriteLog("No IO_Finalize() found");
  }
  FncIO_IsActive = ( INT ( *) ( INT *)) dlsym( wLibHandle, "IO_IsActive");
  if( dlerror() != NULL )  {
    WriteLog("No IO_IsActive() found");
  }
  FncDIGIO_Set_LED_SW1 = ( INT ( *) ( BYTE)) dlsym( wLibHandle,
"DIGIO_Set_LED_SW1");
  if( dlerror() != NULL )  {
    WriteLog("No DIGIO_Set_LED_SW1() found");
  }
  FncDIGIO_Set_LED_SW2 = ( INT ( *) ( BYTE)) dlsym( wLibHandle,
"DIGIO_Set_LED_SW2");
  if( dlerror() != NULL )  {
```

```
    WriteLog("No DIGIO_Set_LED_SW2() found");
  }
  FncDIGIO_SetLED_SW1_BlinkFreq = ( INT ( *) ( DOUBLE)) dlsym(
wLibHandle, "DIGIO_SetLED_SW1_BlinkFreq");
  if( dlerror() != NULL)  {
    WriteLog("No DIGIO_SetLED_SW1_BlinkFreq() found");
  }
  FncDIGIO_SetLED_SW1_BlinkStatus = ( INT ( *) ( BYTE)) dlsym( wLibHandle,
"DIGIO_SetLED_SW1_BlinkStatus");
  if( dlerror() != NULL)  {
    WriteLog("No DIGIO_SetLED_SW1_BlinkStatus() found");
  }
  FncDIGIO_SetLED_SW2_BlinkFreq = ( INT ( *) ( DOUBLE)) dlsym(
wLibHandle, "DIGIO_SetLED_SW2_BlinkFreq");
  if( dlerror() != NULL)  {
    WriteLog("No DIGIO_SetLED_SW2_BlinkFreq() found");
  }
  FncDIGIO_SetLED_SW2_BlinkStatus = ( INT ( *) ( BYTE)) dlsym( wLibHandle,
"DIGIO_SetLED_SW2_BlinkStatus");
  if( dlerror() != NULL)  {
    WriteLog("No DIGIO_SetLED_SW2_BlinkStatus() found");
  }
  FncANAGIO_GetAnalogIn = ( INT ( *) ( INT, INT *)) dlsym( wLibHandle,
"ANAGIO_GetAnalogIn");
  if( dlerror() != NULL )  {
    WriteLog("No ANAGIO_GetAnalogIn() found");
  }
  FncDIGIO_ConfigureInterruptService = ( INT ( *) ( BYTE, BYTE, VOID ( *) (
input_int_t), USHORT)) dlsym( wLibHandle, "DIGIO_ConfigureInterruptService");
  if( dlerror() != NULL)  {
    WriteLog( "No DIGIO_ConfigureInterruptService() found");
  }
}
```

Ios_module.h

```
#ifndef __IO_MODULE_H
  #define __IO_MODULE_H
  #include <stdarg.h>
  #include <time.h>
  #include "owa3x/owcomdefs.h"
  #include "owa3x/owerrors.h"
  #include "owa3x/owa31/IOs_ModuleDefs.h"
#define LIBIO "/lib/libIOs_Module.so"
#ifdef __IO_MODULE_C
    VOID *LibGPIOHandle = NULL;
    INT InitIOModule( VOID );
    INT EndIOModule( VOID );
```

```
    VOID LoadIOsGenericFunctions( VOID *wLibHandle);
    //IOs functions
    INT (*FncIO_Initialize)( VOID );
    INT (*FncIO_Start)( VOID);
    INT (*FncIO_Finalize)( VOID);
    INT (*FncIO_IsActive)( INT *);
    INT (*FncDIGIO_Set_LED_SW1)( BYTE);
    INT (*FncDIGIO_Set_LED_SW2)( BYTE);
    INT (*FncDIGIO_SetLED_SW1_BlinkFreq)( DOUBLE);
    INT (*FncDIGIO_SetLED_SW1_BlinkStatus)( BYTE);
    INT (*FncDIGIO_SetLED_SW2_BlinkFreq)( DOUBLE);
    INT (*FncDIGIO_SetLED_SW2_BlinkStatus)( BYTE);
    INT (*FncANAGIO_GetAnalogIn)( INT, INT *);
    INT ( *FncDIGIO_ConfigureInterruptService)( BYTE wInput, BYTE wEdge,
VOID(*)(input_int_t), USHORT wNumInts );
    INT Config_Interrupts( VOID);
    VOID InputIntHandler( input_int_t wInState);
    INT Get_Interrupts( VOID);
    INT conta1=0;
  #else
    extern INT InitIOModule( VOID );
    extern INT EndIOModule( VOID );
    extern VOID LoadIOsGenericFunctions( VOID *wLibHandle);
    extern INT (*FncIO_Initialize)( VOID );
    extern INT (*FncIO_Start)( VOID);
    extern INT (*FncIO_Finalize)( VOID);
    extern INT (*FncIO_IsActive)( INT *);
    extern INT (*FncDIGIO_Set_LED_SW1)( BYTE);
    extern INT (*FncDIGIO_Set_LED_SW2)( BYTE);
    extern INT (*FncDIGIO_SetLED_SW1_BlinkFreq)( DOUBLE);
    extern INT (*FncDIGIO_SetLED_SW1_BlinkStatus)( BYTE);
    extern INT (*FncDIGIO_SetLED_SW2_BlinkFreq)( DOUBLE);
    extern INT (*FncDIGIO_SetLED_SW2_BlinkStatus)( BYTE);
    extern INT (*FncANAGIO_GetAnalogIn)( INT, INT *);
    extern INT ( *FncDIGIO_ConfigureInterruptService)( BYTE wInput, BYTE
wEdge, VOID(*)(input_int_t), USHORT wNumInts );extern INT Config_Interrupts(
VOID);
    extern INT Get_Interrupts( VOID);
  #endif
#endif
```

Gps_module.c

```
#define __GPS_MODULE_C
//-------------------------------------------------------------//
//System Includes
//-------------------------------------------------------------//
#include <stdio.h>
```

```
   #include <string.h>
   #include <stdarg.h>
   #include <time.h>
   #include <signal.h>
   #include <errno.h>
   #include <ctype.h>
//-------------------------------------------------------------------//
//User Includes
//-------------------------------------------------------------------//
#include "LoadLibs.h"
#include "gps_module.h"
#include "Test_Module.h"
   #include "owa3x/owcomdefs.h"
   #include "owa3x/owerrors.h"
   #include "owa3x/GPS_ModuleDefs.h"
//-------------------------------------------------------------------//
// Function: InitGPSModule()
// Input Params:
// Output Params:
// Description:
//-------------------------------------------------------------------//
INT InitGPSModule( VOID )
{
  TGPS_MODULE_CONFIGURATION  GPSConfiguration;
  CHAR                    GpsValidType[][20] = {"NONE", "GPS_UBLOX"};
  CHAR                    GpsValidProtocol[][10] = {"NMEA", "BINARY"};
  INT                   ReturnCode;
  if (InitGpsFirstTime) {
    // load GPS library and its functions
    LoadExternalLibrary((char *)LIBGPS, &LibGPSHandle);
    LoadGPSGenericFunctions(LibGPSHandle);
    InitGpsFirstTime = FALSE;
  }
  memset( &GPSConfiguration, 0, sizeof( TGPS_MODULE_CONFIGURATION));
  strcpy(GPSConfiguration.DeviceReceiverName, GpsValidType[1]);
  GPSConfiguration.ParamBaud = B9600;
  GPSConfiguration.ParamLength = CS8;
  GPSConfiguration.ParamParity = IGNPAR;
  strcpy(GPSConfiguration.ProtocolName, GpsValidProtocol[0]);
  GPSConfiguration.GPSPort = COM6;
        // GPS module initialization.
        if( ( ReturnCode = ( *FncGPS_Initialize)( ( void *) &GPSConfiguration)) !=
NO_ERROR) {
    WriteLog("Error %d in GPS_Initialize()", ReturnCode);
    if (ReturnCode != ERROR_GPS_ALREADY_INITIALIZED) {
          return -1;
    }
  }
  // GPS receiver startup
```

```
        if ( ( ReturnCode = (*FncGPS_Start)()) != NO_ERROR ) { //start GPS
receiver.
     WriteLog("Error %d in GPS_Start()", ReturnCode);
     if (ReturnCode != ERROR_GPS_ALREADY_STARTED) {
          return -1;
     }
     // Create thread that retrieves data from GPS
   }
   gps_thread_running = TRUE;
   pthread_create(&gps_data, NULL, GPS_RetrieveData, NULL);
   printf("\nGPS-> Module initialized & started");
   return NO_ERROR;
}
//-------------------------------------------------------------------//
// Function: EndGPSModule()
// Input Params:
// Output Params:
// Description:
//-------------------------------------------------------------------//
INT EndGPSModule( VOID )
{
   INT ReturnCode;
   gps_thread_running = FALSE;
   pthread_join(gps_data, NULL);
   if( ( ReturnCode = (*FncGPS_Finalize)()) != NO_ERROR ) {
     WriteLog("Error %d in GPS_Finalize()", ReturnCode);
     return -1;
   }
   if (FinalizeApplication) {
     UnloadExternalLibrary( LibGPSHandle );
   }
   WriteLog("GPS-> Module finalized");
   return NO_ERROR;
}
//-------------------------------------------------------------------//
// Function: LoadGPSGenericFunctions()
// Input Params:
// Output Params:
// Description:
//-------------------------------------------------------------------//
VOID LoadGPSGenericFunctions( VOID *wLibHandle )
{
        FncGPS_Initialize = ( INT ( *) ( VOID *)) dlsym( wLibHandle,
"GPS_Initialize");
   if( dlerror() != NULL) {
     WriteLog("No GPS_Initialize() found");
   }
        FncGPS_IsActive = ( INT ( *) ( INT *)) dlsym( wLibHandle,
"GPS_IsActive");
   if( dlerror() != NULL) {
```

```
    WriteLog("No GPS_IsActive() found");
  }
      FncGPS_Finalize = ( INT ( *) ( VOID)) dlsym( wLibHandle,
"GPS_Finalize");
  if( dlerror() != NULL) {
    WriteLog("No GPS_Finalize() found");
  }
      FncGPS_Start = ( INT ( *) (VOID)) dlsym( wLibHandle, "GPS_Start");
  if( dlerror() != NULL)  {
    WriteLog("No GPS_Start() found");
  }
      FncGPS_GetPosition_Polling = ( INT ( *) (TGPS_POS *)) dlsym(
wLibHandle, "GPS_GetPosition_Polling");
  if( dlerror() != NULL)  {
    WriteLog("No GPS_GetPosition_Polling() found");
  }
      FncGPS_GetSpeedCourse_Polling = ( INT ( *) (TGPS_SPEED *)) dlsym(
wLibHandle, "GPS_GetSpeedCourse_Polling");
  if( dlerror() != NULL)  {
    WriteLog("No GPS_GetSpeedCourse_Polling() found");
  }
      FncGPS_GetUTCDateTime = ( INT ( *) (TUTC_DATE_TIME *)) dlsym(
wLibHandle, "GPS_GetUTCDateTime");
  if( dlerror() != NULL)  {
    WriteLog("No GPS_GetUTCDateTime() found");
  }
      FncGPS_GetGeodetic_Coordinates = ( INT ( *)
(tGEODETIC_COORDINATES *)) dlsym( wLibHandle,
"GPS_GetGeodetic_Coordinates");
  if( dlerror() != NULL)  {
    WriteLog("No GPS_GetGeodetic_Coordinates() found");
  }
}
//-------------------------------------------------------------------//
// Function: GPS_RetrieveData()
// Input Params:
// Output Params:
// Description:
//   This thread retrieves the gps position and speed and signals
//   the fix type using the green led.
//-------------------------------------------------------------------//
VOID *GPS_RetrieveData( VOID *arg )
{
  INT       ReturnCode;
  TGPS_POS    Pos;
  TGPS_SPEED  Speed;
  while( gps_thread_running) {
    Pos.DataValid = FALSE;
    if( ( ReturnCode = (*FncGPS_GetPosition_Polling)( &Pos)) != NO_ERROR){
      WriteLog("GPS->error FncGPS_GetPosition_Polling %d", ReturnCode);
```

```
        printf("GPS->error FncGPS_GetPosition_Polling");
    }
    if((ReturnCode = (*FncGPS_GetSpeedCourse_Polling)( &Speed)) !=
NO_ERROR){
      WriteLog("GPS->error FncGPS_GetSpeedCourse_Polling %d", ReturnCode);
        printf("GPS->error FncGPS_GetSpeedCourse_Polling");
    }
    pthread_mutex_lock( &gpsDataLock);
    GPSNewPosition = TRUE;
    if( Pos.DataValid == TRUE){
      memcpy( &currentPos, &Pos, sizeof( TGPS_POS));
      memcpy( &currentSpeed, &Speed, sizeof( TGPS_SPEED));
    } else
      currentPos.DataValid = FALSE;
    pthread_mutex_unlock( &gpsDataLock);
    ReturnCode = ( *FncGPS_GetGeodetic_Coordinates)(&GeoCoord);
    if( ReturnCode != NO_ERROR )
      WriteLog("Error %d in GPS_GetGeodetic_Coordinates()", ReturnCode);
    /*else {
                printf( "\n---------------------------------------------------\n");
                    printf ("LATITUDE --> %02d degrees %02d minutes
%04.04f seconds %c\n",GeoCoord.Latitude.Degrees,GeoCoord.Latitude.Minutes,
        GeoCoord.Latitude.Seconds, GeoCoord.Latitude.Dir);
                    printf ("LONGITUDE --> %03d degrees %02d minutes
%04.04f seconds %c\n",GeoCoord.Longitude.Degrees,
GeoCoord.Longitude.Minutes, GeoCoord.Longitude.Seconds,
GeoCoord.Longitude.Dir);
                    printf ("ALTITUDE --> %04.04f meters\n",
GeoCoord.Altitude);
                    printf ("NAV STATUS --> %s\n", GeoCoord.NavStatus);
                    printf( "---------------------------------------------------\n");
    }*/
    if (strcmp(GeoCoord.NavStatus, "NF") == 0) { //NO fix
        greenled(LEDON, 1);
    }
    else if (strcmp(GeoCoord.NavStatus, "G2") == 0) { //2D fix
        greenled(LEDON, 5);
    }
    else if (strcmp(GeoCoord.NavStatus, "G3") == 0) { //3D fix
        greenled(LEDON, 10);
    }
    Wait(ONE_SECOND,0);
  }
  pthread_exit(NULL);
}
```

Gps_module.h

```
#ifndef __GPS_MODULE_H
  #define __GPS_MODULE_H
  //-----------------------------------------------------------------//
   //System Includes
   //-----------------------------------------------------------------//
   #include <string.h>
   #include <stdarg.h>
   #include <time.h>
   #include <pthread.h>
   //-----------------------------------------------------------------//
   //User Includes
   //-----------------------------------------------------------------//
   #include "owa3x/owcomdefs.h"
   #include "owa3x/owerrors.h"
   #include "owa3x/GPS_ModuleDefs.h"
   #define LIBGPS       "/lib/libGPS_Module.so"
  #ifdef __GPS_MODULE_C
    VOID *LibGPSHandle = NULL;
    BOOL InitGpsFirstTime = TRUE;
    BOOL FinalizeApplication=FALSE;
    INT InitGPSModule( VOID );
    INT EndGPSModule( VOID );
    VOID LoadGPSGenericFunctions( VOID *wLibHandle );
    INT CalculateTimerFromSpeed( FLOAT *wCurrentSpd, FLOAT *wPrevSpd );
    VOID GetGPSPosition( TGPS_POS *wPos, TGPS_SPEED  *wSpeed );
    VOID  *GPS_RetrieveData ( VOID *arg);

    pthread_t             gps_data;
    pthread_mutex_t       gpsDataLock = PTHREAD_MUTEX_INITIALIZER;
    BOOL                  gps_thread_running;
    BOOL                  GPSNewPosition = FALSE;
    TGPS_POS              currentPos;
    TGPS_SPEED            currentSpeed;
    tGEODETIC_COORDINATES   GeoCoord;
    //GPS functions
    INT (*FncGPS_Initialize)( VOID *);
    INT (*FncGPS_IsActive)( INT *);
    INT (*FncGPS_Finalize)( VOID);
    INT (*FncGPS_Start)( VOID);
    INT (*FncGPS_GetPosition_Polling)(TGPS_POS *);
    INT (*FncGPS_GetSpeedCourse_Polling)(TGPS_SPEED *);
    INT (*FncGPS_GetUTCDateTime)(TUTC_DATE_TIME *);
    INT (*FncGPS_GetGeodetic_Coordinates)( tGEODETIC_COORDINATES *);
    extern VOID greenled( INT onoff, INT freq );
  #else
    // Exported functions & variables
    extern INT InitGPSModule( VOID );
```

```
    extern INT EndGPSModule( VOID );
    extern BOOL FinalizeApplication;
    extern VOID LoadGPSGenericFunctions( VOID *wLibHandle );
    extern INT CalculateTimerFromSpeed( FLOAT *wCurrentSpd, FLOAT
*wPrevSpd );
    extern VOID GetGPSPosition( TGPS_POS *wPos, TGPS_SPEED  *wSpeed );
    extern INT (*FncGPS_Initialize)( VOID *);
    extern INT (*FncGPS_IsActive)( INT *);
    extern INT (*FncGPS_Finalize)( VOID);
    extern INT (*FncGPS_Start)( VOID);
    extern INT (*FncGPS_GetPosition_Polling)(TGPS_POS *);
    extern INT (*FncGPS_GetSpeedCourse_Polling)(TGPS_SPEED *);
    extern                INT                (*FncGPS_GetGeodetic_Coordinates)(
tGEODETIC_COORDINATES *);
    extern INT (*FncGPS_GetUTCDateTime)(TUTC_DATE_TIME *);
  #endif
#endif
```

Gsm_module.c

```
#define __GSM_MODULE_C
//-----------------------------------------------------------------//
//System Includes
//-----------------------------------------------------------------//
#include <stdio.h>
   #include <string.h>
   #include <stdarg.h>
   #include <time.h>
   #include <signal.h>
   #include <errno.h>
   #include <ctype.h>
   #include <semaphore.h>
//-----------------------------------------------------------------//
//User Includes
//-----------------------------------------------------------------//
#include "LoadLibs.h"
#include "gsm_module.h"
#include "Test_Module.h"
   #include "owa3x/owcomdefs.h"
   #include "owa3x/owerrors.h"
   #include "owa3x/GSM_ModuleDefs.h"
//-----------------------------------------------------------------//
// Function: InitGPSModule()
// Input Params:
// Output Params:
// Description:
//-----------------------------------------------------------------//
INT InitGSMModule( VOID )
```

```
{
   TGSM_MODULE_CONFIGURATION  gsmConfig;
   INT ReturnCode = NO_ERROR;
   INT isActive;
    if (gsm_init_first_time){
     LoadExternalLibrary(GSMLIB, &LibHandleGSM);
     LoadGSMGenericFunctions( LibHandleGSM);
     gsm_init_first_time = FALSE;
    }
    strcpy( ( CHAR*)( gsmConfig.wCode), sim_pin);
    strcpy( ( CHAR*)( gsmConfig.wMECode), me);
    gsmConfig.gsm_action = gsm_event_handler;
    memset( &GsmEventBuffer, 0x00, sizeof(GsmEventBuffer));
    GsmEventsWr = 0;
    GsmEventsRd = 0;
    sem_init( &GsmEventsSem, 0, 0);
    if( ( ( *FncGSM_Initialize) ( ( VOID*) ( &gsmConfig))) > 0){
      printf( "OWASYS--> ERROR on GSM Initialization ( %d )\n", ReturnCode);
      exit( 1);
    }
    if ( ( ReturnCode = ( *FncGSM_Start) ( )) != NO_ERROR){
     printf( "OWASYS--> ERROR on GSM Start ( %d )\n", ReturnCode);
     exit( 1);
    }
    printf( "OWASYS--> OK on GSM Initialization \n");
    if( ( ReturnCode = (*FncGSM_IsActive)(&isActive)) != NO_ERROR) {
     WriteLog("GSM->Error %d in GSM_IsActive()", ReturnCode);
     return -1;
    }
     if( isActive == 1){
          printf("\n*******************    GSM    IS    UP    AND    RUNNING
************************\n");
     } else {
          printf("\n*******************    GSM    IS    NOOOOOT    RUNNING
************************\n");
     }
  if( ( ReturnCode = ( *FncGSM_SMSIndications) ( 1)) == NO_ERROR){
    printf( "OWASYS--> OK SMS Indications Active\n");
  } else{
    printf(  "OWASYS-->  ERROR  on  SMS  Indications  Activation  (  %d  )\n",
ReturnCode);
  }
  runGSMHandler = TRUE;
  pthread_create( &gsmEvents,NULL,GSMHandleEvents,NULL);
  return NO_ERROR;
}
//-------------------------------------------------------------//
// Function: EndGPSModule()
// Input Params:
// Output Params:
```

```
// Description:
//---------------------------------------------------------------//
INT EndGSMModule( VOID )
{
  INT ReturnCode;

  if( ( ReturnCode = (*FncGSM_Finalize)()) != NO_ERROR) {
    WriteLog("GSM->Error %d in GSM_Finalize()", ReturnCode);
    return -1;
  }
  if(runGSMHandler == TRUE) {
      runGSMHandler = FALSE;
      sem_post( &GsmEventsSem);
      pthread_join( gsmEvents, NULL);
  }
  sem_destroy( &GsmEventsSem);
  UnloadExternalLibrary( LibHandleGSM);
  printf("Module GSM Finalized");

  return NO_ERROR;
}
VOID gsm_event_handler( gsmEvents_s *pToEvent)
{
 INT auxi = (GsmEventsWr+1);
  GsmEventBuffer[GsmEventsWr] = *pToEvent;
  if( GsmEventsWr == GsmEventsRd) {
    GsmEventsWr = auxi;
  } else {
    if( auxi >= MAX_EVENTS) {
      auxi = 0;
    }
    if( auxi != GsmEventsRd) {
      GsmEventsWr = auxi;
    }
  }
  if( GsmEventsWr >= MAX_EVENTS) {
    GsmEventsWr = 0;
  }
   sem_post( &GsmEventsSem);
}
VOID* GSMHandleEvents( VOID *arg)
{
  gsmEvents_s *owEvents;
  gsmEvents_s LocalEvent;
  while( runGSMHandler ){
    sem_wait( &GsmEventsSem);
    if( GsmEventsRd == GsmEventsWr) {
      continue;
    }
    LocalEvent = GsmEventBuffer[GsmEventsRd++];
```

```
    owEvents = &LocalEvent;
    if( GsmEventsRd >= MAX_EVENTS) {
      GsmEventsRd = 0;
    }
    switch ( owEvents->gsmEventType){
     case GSM_COVERAGE:
        WriteLog( "GSM EVENT--> GSM COVERAGE INFO: %d\n", owEvents-
>evBuffer[ 0]);
        if( ( owEvents->evBuffer[ 0] == NO_NETWORK) ||
           ( owEvents->evBuffer[ 0] == NETWORK_SEARCH)) {
          gsm_coverage = FALSE;
        }
           break;
        case GSM_RING_VOICE:
         WriteLog("\n\nGSM->GSM RING VOICE signal Phone Number: %s \n",
owEvents->evBuffer);
        (*FncGSM_AnswerCall)();
        pthread_create(&HangupCall, NULL, HangupCallThread, NULL);
        pthread_detach(HangupCall);
        break;
         case GSM_RING_DATA:
        WriteLog("GSM->GSM RING DATA signal Phone Number: %s ",
owEvents->evBuffer);
        (*FncGSM_HangUp)( );
        break;
      case GSM_NEW_SMS:
        WriteLog("GSM->NEW SMS RECEIVED\n");
        pthread_mutex_lock(&GSMLock);
        EventInGSM = SMS_EVENT;
        pthread_mutex_unlock(&GSMLock);
        break;
      case GSM_FAILURE:
        printf( "----------- GSM ERROR ----------");
        gsmError = TRUE;
        break;
         case GSM_STOP_SENDING_DATA:
        printf( "GSM EVENT--> Stop Sending Data.\n");
        sendingData = FALSE;
        break;
      case GSM_START_SENDING_DATA:
        printf( "GSM EVENT--> Start Sending Data.\n");
        sendingData = TRUE;
        break;
         case GSM_CALL_RELEASED:
        callEnd = TRUE;
        printf( "----------- EVENT: Call Finalized by the peer ----------");
        printf(" \n>>");
        break;
      case GSM_GPRS_COVERAGE:
        if( ( owEvents->evBuffer[ 0] == GPRS_NO_NETWORK) ||
```

```
            ( owEvents->evBuffer[ 0] == GPRS_SEARCHING) ||
            ( owEvents->evBuffer[ 0] == 4)) { //4=UNKNOWN (not documented)
            WriteLog("GSM->GPRS NO net available");
            gprsRegistered = FALSE;
              } else{
            WriteLog("GSM->GPRS AVAILABLE NOW");
            gprsRegistered = TRUE;
              }
          break;
        default:
          printf( "OWASYS--> Signal Event not found ...%d \n", owEvents->gsmEventType);
             break;
      }
      if( EventInGSM){
        pthread_mutex_lock(&GSMLock);
        memcpy( GSMBuffer, owEvents->evBuffer, EVENTS_BUFFER_SIZE);
        pthread_mutex_unlock(&GSMLock);
        sem_post( &Events);
      }
    }
    pthread_exit(NULL);
}
VOID *HangupCallThread( VOID *arg )
{
    Wait(30,0);
    (*FncGSM_HangUp)();
    return NULL;
}
VOID GSM_SendSMS( CHAR *mynumber )
{
    INT ReturnCode;
    // Update gsm time, send sms to itself and catch time
    if( ( ReturnCode = (*FncGSM_SendSMS)((UCHAR *) body_alerta, (UCHAR*)mynumber, 0)) != NO_ERROR) {
      WriteLog("GSM->Error on GSM_SendSMS() %d", ReturnCode);
Wait( ONE_SECOND, 0);
    } else{
     printf("Se envio el mensaje de Alerta\n");
}
    //return NO_ERROR;
}
//-----------------------------------------------------------------//
// Function: GSMImsi()
// Input Params:
// Output Params:
//    *wImsi : IMSI string
// Description:
//    Gets the IMSI.
//-----------------------------------------------------------------//
```

```
INT GSMImsi( CHAR *wImsi, WORD wSize )
{
  return ((*FncGSM_GetIMSI)(wImsi, wSize));
}
//-----------------------------------------------------------------//
// Function: ReadSMS()
// Input Params:
//    wSMSIndex : sms index
// Output Params:
//    *wSMS : read sms string.
// Description:
//    This function reads the sms at the specified index.
//-----------------------------------------------------------------//
INT ReadSMS( INT wSMSIndex, SMS_s *wSMS )
{
  UCHAR         smsSize;
  INT           ReturnCode;
  FILE          *fptr;
  DATETIME      *Temp;
  memset( wSMS, 0, sizeof( SMS_s));
  if( ( ReturnCode = (*FncGSM_ReadSMS)(wSMS, &smsSize, wSMSIndex, 0)) !=
NO_ERROR) {
    WriteLog("GSM->Error on GSM_ReadSMS() %d", ReturnCode);
    return ReturnCode;
  }
  //Now Delete Incoming SMS....
  (*FncGSM_DeleteSMS) ( wSMSIndex, 0);
  //Log the SMS
  fptr = fopen(SMS_FILE, "a+");
  if(fptr) {
    Temp = &(wSMS->owSCDateTime);
    fprintf(fptr, "[%02d/%02d/%02d %02d:%02d:%02d] ->",
          Temp->day,  Temp->month,  Temp->year,  Temp->hour,  Temp->minute,
Temp->second);
    fprintf(fptr, " %s: %s\n", wSMS->owSA_DA, wSMS->owBody);
    fclose(fptr);
  }
  //printf("Mensaje Recibido: %s\n",wSMS->owBody);
  return NO_ERROR;
}
//-----------------------------------------------------------------//
// Function: LoadGPSGenericFunctions()
// Input Params:
// Output Params:
// Description:
//-----------------------------------------------------------------//
INT LoadGSMGenericFunctions( VOID *wLibHandle )
{
 INT retVal = NO_ERROR;
```

```
  FncGSM_Initialize = ( INT ( *) ( VOID *)) dlsym( wLibHandle,
"GSM_Initialize");
  if( dlerror() != NULL) {
    WriteLog( "OWASYS--> No GSM_Initialize found...\n");
    retVal = ERROR_LOADING_FUNCTION;
  }
  FncGSM_Start = ( INT ( *) ( VOID)) dlsym( wLibHandle, "GSM_Start");
  if( dlerror() != NULL) {
     WriteLog( "OWASYS--> No GSM_Start found...\n");
     retVal = ERROR_LOADING_FUNCTION;
  }
  FncGSM_IsActive = ( INT ( *) ( INT *)) dlsym( wLibHandle, "GSM_IsActive");
  if( dlerror() != NULL) {
     WriteLog( "OWASYS--> No GSM_IsActive found...\n");
     retVal = ERROR_LOADING_FUNCTION;
  }
  FncGSM_Finalize = ( INT ( *) ( VOID)) dlsym( wLibHandle, "GSM_Finalize");
  if( dlerror() != NULL) {
     WriteLog( "OWASYS--> No GSM_Finalize found...\n");
     retVal = ERROR_LOADING_FUNCTION;
  }
  FncGSM_GetIMEI = ( INT ( *) ( CHAR*, INT)) dlsym( wLibHandle,
"GSM_GetIMEI");
  if( dlerror() != NULL){
    WriteLog( "OWASYS--> No GSM_GetIMEI found...\n");
    retVal = ERROR_LOADING_FUNCTION;
  }
  FncGSM_GetIMSI = ( INT ( *) ( CHAR*, INT)) dlsym( wLibHandle,
"GSM_GetIMSI");
  if( dlerror() != NULL){
    WriteLog( "OWASYS--> No GSM_GetIMSI found...\n");
    retVal = ERROR_LOADING_FUNCTION;
  }
  FncGSM_SignalStrength = ( INT ( *) ( BYTE*)) dlsym( wLibHandle,
"GSM_GetSignalStrength");
  if( dlerror() != NULL)  {
    WriteLog( "OWASYS--> No GSM_GetSignalStrength found...\n");
    retVal = ERROR_LOADING_FUNCTION;
  }
  FncGSM_GetRegistration = ( INT ( *) ( CHAR* )) dlsym( wLibHandle,
"GSM_GetRegistration" );
  if( dlerror() != NULL)  {
    WriteLog( "OWASYS--> No GSM_GetRegistration found...\n");
    retVal = ERROR_LOADING_FUNCTION;
  }
  FncGSM_HangUp = ( INT ( *) ( )) dlsym( wLibHandle, "GSM_HangUp" );
  if( dlerror() != NULL)  {
    WriteLog( "OWASYS--> No GSM_HangUp found...\n");
    retVal = ERROR_LOADING_FUNCTION;
  }
```

```
  FncGSM_AnswerCall = ( INT ( *) ( )) dlsym( wLibHandle, "GSM_AnswerCall" );
  if( dlerror() != NULL)  {
    WriteLog( "OWASYS--> No GSM_AnswerCall found...\n");
    retVal = ERROR_LOADING_FUNCTION;
  }
  FncGSM_GetGPRSRegistration = ( INT ( *) ( BYTE* )) dlsym( wLibHandle,
"GSM_GetGPRSRegistration" );
  if( dlerror() != NULL)  {
    WriteLog( "OWASYS--> No GSM_GetGPRSRegistration found...\n");
    retVal = ERROR_LOADING_FUNCTION;
  }
  FncGSM_GetDateTime = ( INT ( *) ( DATETIME*)) dlsym( wLibHandle,
"GSM_GetDateTime");
  if( dlerror() != NULL)  {
    WriteLog( "OWASYS--> No GSM_GetDateTime found...\n");
    retVal = ERROR_LOADING_FUNCTION;
  }
  FncGSM_SetDateTime = ( INT ( *) ( DATETIME*)) dlsym( wLibHandle,
"GSM_SetDateTime");
  if( dlerror() != NULL)  {
    WriteLog( "OWASYS--> No GSM_SetDateTime found...\n");
  }
  FncGSM_SMSIndications = ( INT ( *) ( CHAR)) dlsym( wLibHandle,
"GSM_SMSIndications" );
  if( dlerror() != NULL)  {
    WriteLog( "OWASYS--> No GSM_SMSIndications found...\n");
    retVal = ERROR_LOADING_FUNCTION;
  }
  FncGSM_SendSMS = ( INT ( *) ( UCHAR*,UCHAR*,CHAR)) dlsym(
wLibHandle, "GSM_SendSMS" );
  if( dlerror() != NULL)  {
    WriteLog( "OWASYS--> No GSM_SendSMS found...\n");
    retVal = ERROR_LOADING_FUNCTION;
  }
  FncGSM_ReadSMS = ( INT ( *) ( SMS_s*, UCHAR*, CHAR, CHAR)) dlsym(
wLibHandle, "GSM_ReadSMS");
  if( dlerror() != NULL){
    WriteLog( "OWASYS--> No GSM_ReadSMS found...\n");
    retVal = ERROR_LOADING_FUNCTION;
  }
  FncGSM_DeleteSMS = ( INT ( *) ( CHAR, CHAR)) dlsym( wLibHandle,
"GSM_DeleteSMS" );
  if( dlerror() != NULL)  {
    WriteLog( "OWASYS--> No GSM_DeleteSMS found...\n");
    retVal = ERROR_LOADING_FUNCTION;
}
  return retVal;
}
```

Gsm_module.h

```
#ifndef __GSMS_MODULE_H
  #define __GSM_MODULE_H
  //---------------------------------------------------------------//
   //System Includes
   //---------------------------------------------------------------//
   #include <string.h>
   #include <stdarg.h>
   #include <time.h>
   #include <pthread.h>
   #include <semaphore.h>
   //---------------------------------------------------------------//
   //User Includes
   //---------------------------------------------------------------//
   #include "owa3x/owcomdefs.h"
   #include "owa3x/owerrors.h"
   #include "owa3x/GSM_ModuleDefs.h"
   #define GSMLIB       "/lib/libGSM_Module.so"
   #define ERROR_LOADING_LIBRARY                  3000
   #define ERROR_UNLOADING_LIBRARY                3001
   #define ERROR_LOADING_FUNCTION                 3002
   #define ERROR_UNLOADING_FUNCTION               3003
   #define  MAX_EVENTS     10
   #define MIN_GSM_COVERAGE   6
   //Events Definition
   #define GSM_NO_EVENT       0
   #define SMS_EVENT          1
   #define MAX_BUFFER_SIZE    255
   #define TAG_TIME_SYNCH     "TIMESYNCH"
  #ifdef __GSM_MODULE_C
    INT InitGSMModule( VOID );
    INT EndGSMModule( VOID );
    INT LoadGSMGenericFunctions( VOID *wLibHandle );
    VOID GSM_SendSMS( CHAR *mynumber );
    INT GSMImsi( CHAR *wImsi, WORD wSize );
    INT IsThereGSMEvent( CHAR *wBuffer );
    //INT ParseSMS( StringParser_t *wCommands, SMS_s wSMS );
    INT ReadSMS( INT wSMSIndex, SMS_s *wSMS );
    sem_t        GsmEventsSem;
    // GSM generic functions:
    INT ( *FncGSM_Initialize)  ( VOID*);
    INT ( *FncGSM_IsActive)    ( INT*);
    INT ( *FncGSM_Finalize)    ( VOID);
    INT ( *FncGSM_Start)       ( VOID);
    INT ( *FncGSM_GetIMEI)     ( CHAR*, INT);
    INT ( *FncGSM_GetIMSI)     ( CHAR*, INT);
    INT ( *FncGSM_SignalStrength)                ( BYTE*);
    INT ( *FncGSM_GetRegistration)               ( CHAR*);
```

```
  INT ( *FncGSM_HangUp)      ( );
  INT ( *FncGSM_AnswerCall)  ( );
  INT ( *FncGSM_GetGPRSRegistration)          ( BYTE*);
  INT ( *FncGSM_GetDateTime) ( DATETIME*);
  INT ( *FncGSM_SetDateTime) ( DATETIME*);
  INT ( *FncGSM_SMSIndications)      ( CHAR);
  INT ( *FncGSM_SendSMS)           ( UCHAR*, UCHAR*, CHAR);
  INT ( *FncGSM_ReadSMS)           ( SMS_s*, UCHAR*, CHAR ,CHAR );
  INT ( *FncGSM_DeleteSMS)         ( CHAR, CHAR);
  BOOL      gsm_init_first_time = TRUE;
  VOID      *LibHandleGSM = NULL;
  VOID      gsm_event_handler( gsmEvents_s *pToEvent);
  VOID      *GSMHandleEvents  ( VOID* arg);
  INT       GsmEventsWr = 0;
  INT       GsmEventsRd = 0;
  INT       InitGsmEventBuffer( VOID);
  gsmEvents_s   GsmEventBuffer[MAX_EVENTS];
  //INT         GsmEventsWr = 0;
  //INT         GsmEventsRd = 0;
  BOOL      runGSMHandler  = FALSE;
  BOOL          gsm_coverage = FALSE;
  pthread_t gsmEvents;
  pthread_t       HangupCall;
  pthread_mutex_t   GSMLock;
  INT           EventInGSM = GSM_NO_EVENT;
  BOOL          sendingData = FALSE;
  BOOL          gprsRegistered = FALSE;
  CHAR          GSMBuffer[ EVENTS_BUFFER_SIZE];
  BOOL  gsmError;
  BOOL  callEnd;
  extern VOID redled( INT onoff, INT freq );
#else
  // Exported functions & variables
  extern INT InitGSMModule( VOID );
  extern INT EndGSMModule( VOID );
  extern INT LoadGSMGenericFunctions( VOID *wLibHandle );
  // GSM generic functions:
  extern INT ( *FncGSM_Initialize)  ( VOID*);
  extern INT ( *FncGSM_IsActive)    ( INT*);
  extern INT ( *FncGSM_Finalize)    ( VOID);
  extern INT ( *FncGSM_Start)      ( VOID);
  extern INT ( *FncGSM_GetIMEI)     ( CHAR*, INT);
  extern INT ( *FncGSM_GetIMSI)     ( CHAR*, INT);
  extern INT ( *FncGSM_SignalStrength)          ( BYTE*);
  extern INT ( *FncGSM_GetRegistration)          ( CHAR*);
  extern INT ( *FncGSM_HangUp)      ( );
  extern INT ( *FncGSM_AnswerCall)  ( );
  extern INT ( *FncGSM_GetGPRSRegistration)          ( BYTE*);
  extern INT ( *FncGSM_GetDateTime) ( DATETIME*);
  extern INT ( *FncGSM_SetDateTime) ( DATETIME*);
```

```
    extern INT ( *FncGSM_SMSIndications)      ( CHAR);
    extern INT ( *FncGSM_SendSMS)             ( UCHAR*, UCHAR*, CHAR);
    extern INT ( *FncGSM_ReadSMS)                   ( SMS_s*, UCHAR*, CHAR
,CHAR );
    extern INT ( *FncGSM_DeleteSMS)           ( CHAR, CHAR);
    extern BOOL  gsmError;
    extern BOOL  callEnd;
    extern sem_t        GsmEventsSem;
    extern VOID GSM_SendSMS( CHAR *mynumber );
    extern INT GSMImsi( CHAR *wImsi, WORD wSize );
    extern INT IsThereGSMEvent( CHAR *wBuffer );
    extern INT ReadSMS( INT wSMSIndex, SMS_s *wSMS );
  #endif
#endif
```

Inet_module.c

```
#define __INET_MODULE_C
//-----------------------------------------------------------------------//
//System Includes
//-----------------------------------------------------------------------//
#include <stdio.h>
//-----------------------------------------------------------------------//
//User Includes
//-----------------------------------------------------------------------//
#include "inet_module.h"
#include "LoadLibs.h"
#include "rtu_module.h"
#include "gsm_module.h"
#include "Test_Module.h"
//-----------------------------------------------------------------------//
//Functions definition
//-----------------------------------------------------------------------//
//---------------------------------------------------------------//
// Function: InitModINET()
// Input Params:
//    Network GPRS configuration parameters.
// Output Params:
//    NO_ERROR : succeded function execution.
// Description:
//    Starts a GPRS session.
//---------------------------------------------------------------//
INT InitModINET( CHAR *user, CHAR *pass, CHAR *dns1, CHAR *dns2, CHAR
*apn )
{
  TINET_MODULE_CONFIGURATION          iNetConfiguration;
  GPRS_ENHANCED_CONFIGURATION         gprsConfiguration;
  INT           ReturnCode;
```

```
  if (inet_first_init) {
    LoadExternalLibrary(INETLIB, &LibHandleINET);
    LoadINETFunctions(LibHandleINET);
    inet_first_init = FALSE;
  }
  sem_init( &iNetHandlerSem,0,0);
  runiNetHandleEvents = TRUE;
  iNetInitialized = TRUE;
  pthread_create( &iNetEvents, NULL, iNetHandleEvents, NULL);
  pthread_detach( iNetEvents);
  strcpy( ( CHAR*) gprsConfiguration.gprsUser, user);
  strcpy( ( CHAR*) gprsConfiguration.gprsPass, pass);
  strcpy( ( CHAR*) gprsConfiguration.gprsDNS1, dns1);
  strcpy( ( CHAR*) gprsConfiguration.gprsDNS2, dns2);
  strcpy( ( CHAR*) gprsConfiguration.gprsAPN, apn);
   memset( &iNetConfiguration, 0, sizeof(
TINET_MODULE_CONFIGURATION));
  iNetConfiguration.wBearer = INET_BEARER_ENHANCED_GPRS;
  iNetConfiguration.inet_action = iNet_handler;
  memset( &InetEventBuffer, 0x00, sizeof(InetEventBuffer));
  InetEventsWr = 0;
  InetEventsRd = 0;
  iNetConfiguration.wBearerParameters = (VOID*) &gprsConfiguration;
  if( ( ReturnCode = ( (*Fnc_iNetInitialize)( ( VOID*) &iNetConfiguration))) !=
NO_ERROR) {
    WriteLog("INET->Error %d in iNetInitialize()", ReturnCode);
    return -1;
  }
  if( ( ReturnCode = (*Fnc_iNetStart) ( )) != NO_ERROR){
    WriteLog("INET->Error %d in iNetStart()", ReturnCode);
    return -1;
  }
  WriteLog("INET->iNet Module initialized & started");
  printf("INET->iNet Module initialized & started");
  return NO_ERROR;
}
//-------------------------------------------------------------------//
// Function: EndModINET()
// Input Params:
// Output Params:
//    NO_ERROR : succeded function execution.
// Description:
//    Stops the GPRS session.
//-------------------------------------------------------------------//
INT EndModINET( VOID )
{
  if( !iNetInitialized)
    return NO_ERROR;
  (*Fnc_iNetStop)();
  runiNetHandleEvents = FALSE;
```

```
   sem_post( &iNetHandlerSem);
   pthread_join( iNetEvents,NULL);
   sem_destroy(&iNetHandlerSem);
   iNetInitialized = FALSE;
   WriteLog("INET->INET Module finalized");
   return NO_ERROR;
}
//---------------------------------------------------------------//
// Function: iNet_handler()
// Input Params:
// Output Params:
// Description:
//    Handler for the iNet signal. Post the iNet event semaphore.
//---------------------------------------------------------------//
VOID iNet_handler( INET_Events *pToEvent )
{
  INT auxi = (InetEventsWr+1);
  InetEventBuffer[InetEventsWr] = *pToEvent;
   if( InetEventsWr == InetEventsRd) {
      InetEventsWr = auxi;
   } else {
      if( auxi >= MAX_EVENTS) {
         auxi = 0;
      }
      if( auxi != InetEventsRd) {
         InetEventsWr = auxi;
      }
   }
   if( InetEventsWr >= MAX_EVENTS) {
      InetEventsWr = 0;
   }
    sem_post(&iNetHandlerSem);
}
//---------------------------------------------------------------//
// Function: iNetHandleEvents()
// Input Params:
// Output Params:
// Description:
//    Thread to manage the iNet events.
//---------------------------------------------------------------//
VOID *iNetHandleEvents( VOID *arg )
{
   INET_Events  *iNet_Events;
   INET_Events LocalEvent;
   while( runiNetHandleEvents){
      sem_wait( &iNetHandlerSem);
      if( runiNetHandleEvents == FALSE) {
         break;
      }
      if( InetEventsRd == InetEventsWr) {
```

```
      continue;
    }
    LocalEvent = InetEventBuffer[InetEventsRd++];
    iNet_Events = &LocalEvent;
    if( InetEventsRd >= MAX_EVENTS) {
      InetEventsRd = 0;
    }
    switch ( iNet_Events->evType) {
      case INET_RELEASED:
        WriteLog("INET->INET RELEASED");
        iNetReleased = TRUE;
        break;
      default:
        WriteLog("INET->INET Thread Event %d not found", iNet_Events->evType);
    }
    sched_yield();
  }
  pthread_exit( NULL);
}
//-------------------------------------------------------------------//
// Function: GetIP()
// Input Params:
// Output Params:
//    wIP : IP string
// Description:
//    This function gets the IP address.
//-------------------------------------------------------------------//
INT GetIP( CHAR *wIP )
{
  return((*Fnc_iNetGetIP)(wIP));
}
/-------------------------------------------------------------------//
// Function: PrintDataCounter()
// Input Params:
// Output Params:
// Description:
//    This function prints the amount of tx/rx GPRS data.
//-------------------------------------------------------------------//
VOID PrintDataCounter( VOID )
{
  LONG TxBytes, RxBytes;
  INT  ReturnCode;
  ReturnCode = (*Fnc_iNetGetDataCounters)(&TxBytes, &RxBytes);
  if( ReturnCode == NO_ERROR){
    WriteLog("DATA over GPRS:\n\t->Tx: %ld\n\t->Rx: %ld\n", TxBytes, RxBytes);
  } else{
    WriteLog("Error %d on getting DATA over GPRS", ReturnCode);
  }
```

```
}
//-------------------------------------------------------------------//
// Function: LoadINETFunctions()
// Input Params:
// Output Params:
//    NO_ERROR : succeded function execution.
// Description:
//    Calls to dlsym to get the addresses where the functions of
//    the library are loaded to be used in the program.
//-------------------------------------------------------------------//
INT LoadINETFunctions( VOID *wLibHandle )
{
  Fnc_iNetInitialize = ( INT ( *) ( VOID*)) dlsym( wLibHandle, "iNet_Initialize");
  if( dlerror() != NULL)  {
    WriteLog("INET->No iNet_Initialize() found");
  }
  Fnc_iNetStart = ( INT ( *) ( VOID)) dlsym( LibHandleINET, "iNet_Start" );
  if( dlerror() != NULL)  {
    WriteLog("INET->No iNet_Start() found");
  }
  Fnc_iNetStop = ( INT ( *) ( VOID)) dlsym( LibHandleINET, "iNet_Finalize" );
  if( dlerror() != NULL)  {
    WriteLog("INET->No iNet_Finalize() found");
  }
  Fnc_iNetIsActive = ( INT ( *) ( INT*)) dlsym( wLibHandle, "iNet_IsActive");
  if( dlerror() != NULL)  {
   WriteLog( "OWASYS--> No iNet_IsActive found...\n");
  }
  Fnc_iNetGetIP = ( INT ( *) ( CHAR*)) dlsym( wLibHandle,
"iNet_GetIPAddress");
  if( dlerror() != NULL)  {
    WriteLog("INET->No iNet_GetIPAddress() found");
  }
  Fnc_iNetGetDataCounters = ( INT ( *) ( LONG*,LONG*)) dlsym( wLibHandle,
"iNet_GetDataCounters");
  if( dlerror() != NULL)  {
    WriteLog("INET->No iNet_GetDataCounters() found");
  }
  return NO_ERROR;}
```

Inet_module.h

```
#ifndef __INET_MODULE_H
  #define __INET_MODULE_H
//-------------------------------------------------------------------//
//System Includes
//-------------------------------------------------------------------//
#include <pthread.h>
```

```
#include <signal.h>
#include <semaphore.h>
#include <owa3x/INET_ModuleDefs.h>
#include "gsm_module.h"
//---------------------------------------------------------------//
//Defines
//---------------------------------------------------------------//
#define INETLIB "/lib/libINET_Module.so"
//---------------------------------------------------------------//
//Public Functions prototypes
//---------------------------------------------------------------//
INT  LoadINETFunctions ( VOID *wLibHandle );
VOID  iNet_handler     ( INET_Events *pToEvent);
VOID  *iNetHandleEvents ( VOID *arg );
INT   InitModINET       ( CHAR* user, CHAR* pass, CHAR* dns1, CHAR* dns2,
CHAR* apn );
INT   EndModINET       ( VOID );
INT   GetIP            ( CHAR* );
VOID  PrintDataCounter  ( VOID );
  #ifdef __INET_MODULE_C
    BOOL iNetReleased         = TRUE;
    BOOL inet_first_init      = TRUE;
    VOID *LibHandleINET       = NULL;
    BOOL runiNetHandleEvents   = FALSE;
    BOOL iNetPendingEvents     = FALSE;
    BOOL iNetInitialized      = FALSE;
    pthread_t iNetEvents;
    sem_t iNetHandlerSem;
    struct sigaction iNetSignalOld;
    INT sigINET;
    INT ( *Fnc_iNetStart)          ( );
    INT ( *Fnc_iNetStop)           ( );
    INT ( *Fnc_iNetIsActive)          ( INT*);
    INT ( *Fnc_iNetGetIP)          ( CHAR*);
    INT ( *Fnc_iNetGetDataCounters)  ( LONG*, LONG*);
    INET_Events   InetEventBuffer[MAX_EVENTS];
    INT          InetEventsWr = 0;
    INT          InetEventsRd = 0;
  #else
    // Exported symbols
    extern BOOL iNetReleased;
    extern INT ( *Fnc_iNetStart)          ( );
    extern INT ( *Fnc_iNetStop)           ( );
    extern INT ( *Fnc_iNetInitialize)      ( VOID*);
    extern INT ( *Fnc_iNetIsActive)          ( INT*);
    extern INT ( *Fnc_iNetGetIP)          ( CHAR*);
    extern INT ( *Fnc_iNetGetDataCounters)  ( LONG*, LONG*);
  #endif
#endif
```

Socket.c

```
#define __SOCKET_C
//---------------------------------------------------------------------------//
//System Includes
//---------------------------------------------------------------------------//
#include <fcntl.h>
#include <arpa/inet.h>
#include <errno.h>
//---------------------------------------------------------------------------//
//User Includes
//---------------------------------------------------------------------------//
#include "socket.h"
#include "LoadLibs.h"
#include "rtu_module.h"
//---------------------------------------------------------------------------//
//Functions definition
//---------------------------------------------------------------------------//
//-----------------------------------------------------------------//
// Function: IsThereRxEvent()
// Input Params:
// Output Params:
// Description:
//-----------------------------------------------------------------//
BOOL IsThereRxEvent( VOID )
{
  BOOL DataRx;

  pthread_mutex_lock(&RxLock);
  DataRx = rx_data_received;
  rx_data_received = FALSE;
  pthread_mutex_unlock(&RxLock);
  return DataRx;
}
//-----------------------------------------------------------------//
// Function: GetReceivedData()
// Input Params:
// Output Params:
// Description:
//-----------------------------------------------------------------//
VOID GetReceivedData( CHAR *Buffer )
{
  pthread_mutex_lock(&RxLock);
  memcpy(Buffer, rx_buffer, RX_BUFFER_SIZE);
  pthread_mutex_unlock(&RxLock);
}
//-----------------------------------------------------------------//
// Function: CreateTCPServerSocket()
// Input Params:
```

```
// Output Params:
// Description:
//---------------------------------------------------------------//
INT CreateTCPServerSocket( CHAR *ip, INT port )
{
  struct sockaddr_in  server;    // Local address
  INT                 SocketFD;
  INT                 error;
  struct hostent      *hp;
  WriteLog("SOCKET->TCP: CreateTCPServerSocket() start");
  // Obtain host data
  if ((hp = gethostbyname ((CHAR*)ip)) == 0) {
    perror("gethostbyname() failed");
    return -1;
  }
  else { // We can resolve the host so we continue
    memset((CHAR*)&server, 0, sizeof(server));
    // Construct the server address structure
    server.sin_family = AF_INET;
    server.sin_addr.s_addr = ((struct in_addr *)(hp->h_addr))->s_addr;
    server.sin_port = htons(port);
    // Obtain socket
    if ((SocketFD = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
      perror("socket() failed");
      return -1;
    }
    // Set non-blocking for the socket
    if (fcntl(SocketFD, F_SETFL, O_NONBLOCK)) {
      printf("SOCKET->TCP Error making socket file descriptor non blocking\n");
      close(SocketFD);
      return -1;
    }
    WriteLog("SOCKET->TCP: connecting to the server");
    // Conect with specified port of the server.
    // If the connection cannot be established immediately and O_NONBLOCK is set
    // for the file descriptor for the socket, connect() shall fail and set errno
    // to [EINPROGRESS], but the connection request shall not be aborted.
    // Subsequent calls to connect() for the same socket, before the connection is
    // established, shall fail and set errno to [EALREADY].
    do {
      error = connect(SocketFD, (struct sockaddr*)&server, sizeof(server));
      if ((error == -1) && (errno != EINPROGRESS) && (errno != EALREADY)
&& (errno != EAGAIN)) {
//        printf("connect:%s %d\n", strerror(errno), errno);
        perror("connect() failed");
        close(SocketFD);
        return -1;
      }
    }while(error != 0);
/*
```

```
    // Uncomment the following lines in case of Blocking socket.
    // Connect with specified port of the server
    if (connect(SocketFD, (struct sockaddr*)&server, sizeof(server)) == -1) {
      printf("connect: %s %d\n", strerror(errno), errno);
      close(SocketFD);
      perror("connect() failed");
      return -1;
    }
*/
    tcp_fd = SocketFD;
    WriteLog("SOCKET->TCP: socket connected");
    printf("\nSOCKET->TCP: socket connected\n");
  }

  WriteLog("SOCKET->TCP: CreateTCPServerSocket() end");
  return 0;
}
//--------------------------------------------------------------------//
// Function: TCP_SendData()
// Input Params:
// Output Params:
// Description:
//--------------------------------------------------------------------//
INT TCP_SendData( CHAR *ip, INT port, CHAR *buffer, INT buffer_length )
{
  struct sockaddr_in  server;
  struct hostent      *hp;
  INT                 sent_bytes = 0;
  INT                 total_bytes = 0;
  INT                 tries;
  WriteLog("\nSOCKET->TCP: TCP_SendData() start\n");
  TCP_StartRXThread();
  (*Fncusecsleep)(1, 0);
  /*sigpipe_signal.sa_handler = sigpipe_handler;
  sigpipe_signal.sa_flags = 0;
  sigaction(SIGPIPE, &sigpipe_signal, &sigpipe_signalOld);
*/
  // Obtain host data
  if ((hp = gethostbyname ((CHAR*)ip)) == 0) {
    perror("gethostbyname() failed");
    return -1;
  }
  else { // We can resolve the host so we continue
    memset((CHAR*)&server, 0, sizeof(server));
    // Construct the server address structure
    server.sin_family = AF_INET;
    server.sin_addr.s_addr = ((struct in_addr *)(hp->h_addr))->s_addr;
    server.sin_port = htons(port);
    //Send message to the server
    do {
```

```
      WriteLog("\nSOCKET->TCP: sending %s, %d bytes\n", buffer, buffer_length);
                  sent_bytes = sendto(tcp_fd, buffer + total_bytes, buffer_length - total_bytes, 0,
                    (struct sockaddr *)&server, sizeof(server));
                  if (sent_bytes == -1) {
        TCP_StopRXThread();
        close(tcp_fd);
        return -1;
      }
                  total_bytes += sent_bytes;
    }while(total_bytes < buffer_length);
    WriteLog("SOCKET->TCP: send message OK");
    //Get an answer from the server
    tries = 0;
    do {
      pthread_mutex_lock(&RxLock);
      if (!rx_data_received && (tries != 1)) {
        pthread_mutex_unlock(&RxLock);
        (*Fncusecsleep)(1, 0);
      }
      else {
        if(rx_data_received) {
          pthread_mutex_unlock(&RxLock);
          break;
        }
        else
          pthread_mutex_unlock(&RxLock);
      }
    }while(tries++ < 120); // If no answer from the server wait for 2 minutes

    TCP_StopRXThread();
  }
  RemoteServerDisconnected = FALSE;
  WriteLog("SOCKET->TCP: TCP_SendData() end");
  return 0;
}
//-------------------------------------------------------------------//
// Function: TCP_RX_Thread()
// Input Params:
// Output Params:
// Description:
//-------------------------------------------------------------------//
VOID *TCP_RX_Thread( VOID *arg )
{
  INT          received_bytes;
  fd_set       ReadFd;
  struct timeval  timeout;
  while(tcp_rx_thread_running) {
    FD_ZERO(&ReadFd);
```

```
    FD_SET(tcp_fd, &ReadFd);
    timeout.tv_sec = 1;
    timeout.tv_usec= 0;
    if (select(tcp_fd + 1, &ReadFd, NULL, NULL, &timeout) > 0) {
      if (FD_ISSET(tcp_fd, &ReadFd)) {
        memset(rx_buffer, 0, RX_BUFFER_SIZE);
        received_bytes = recv(tcp_fd, rx_buffer, RX_BUFFER_SIZE, 0);
        if (received_bytes == 0) {
          WriteLog(" Connection closed by peer");
//          printf("recv:%s %d\n", strerror(errno), errno);
          break;
        }
        if (received_bytes > 0) {
          pthread_mutex_lock( &RxLock);
          rx_data_received = TRUE;
          pthread_mutex_unlock( &RxLock);
          sem_post( &Events);
          WriteLog("SOCKET->TCP: received %d bytes", received_bytes);
          WriteLog("SOCKET->TCP: received %s", rx_buffer);
        }
      }
    }
  }
  pthread_exit(NULL);
}
//-----------------------------------------------------------------//
// Function: TCP_StartRXThread()
// Input Params:
// Output Params:
// Description:
//-----------------------------------------------------------------//
INT TCP_StartRXThread( VOID )
{
  WriteLog("SOCKET->TCP: TCP_StartRXThread() begin");
  tcp_rx_thread_running = TRUE;
  pthread_create(&tcp_rx_thread, NULL, TCP_RX_Thread, NULL);
  WriteLog("SOCKET->TCP: TCP_StartRXThread() end");
  return 0;
}
//-----------------------------------------------------------------//
// Function: TCP_StopRXThread()
// Input Params:
// Output Params:
// Description:
//-----------------------------------------------------------------//
INT TCP_StopRXThread( VOID )
{
  WriteLog("SOCKET->TCP: TCP_StopRXThread() begin");
  tcp_rx_thread_running = FALSE;
  pthread_join(tcp_rx_thread, NULL);
```

```
  WriteLog("SOCKET->TCP: TCP_StopRXThread() end");
  return 0;
}
//-------------------------------------------------------------------//
// Function: sigpipe_handler()
// Input Params:
// Output Params:
// Description:
//-------------------------------------------------------------------//
VOID sigpipe_handler( INT status )
{
  RemoteServerDisconnected = TRUE;
}
```

Socket.h

```
#ifndef __SOCKET_H
  #define __SOCKET_H
//-------------------------------------------------------------------------//
//System Includes
//-------------------------------------------------------------------------//
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <netdb.h>
#include <sys/time.h>
//-------------------------------------------------------------------------//
//User Includes
//-------------------------------------------------------------------------//
#include "Test_Module.h"
#include "rtu_module.h"
//-------------------------------------------------------------------//
//Defines
//-------------------------------------------------------------------//
#define RX_BUFFER_SIZE        1024
//-------------------------------------------------------------------//
//Public Functions prototypes
//-------------------------------------------------------------------//
BOOL  IsThereRxEvent ( );
VOID  GetReceivedData( CHAR *Buffer);

#ifdef __SOCKET_C
  INT CreateTCPServerSocket( CHAR *ip, INT port );
  INT TCP_SendData( CHAR *ip, INT port, CHAR *buffer, INT buffer_length );
  INT TCP_StartRXThread( VOID );
  INT TCP_StopRXThread( VOID );
  VOID sigpipe_handler( INT status );
```

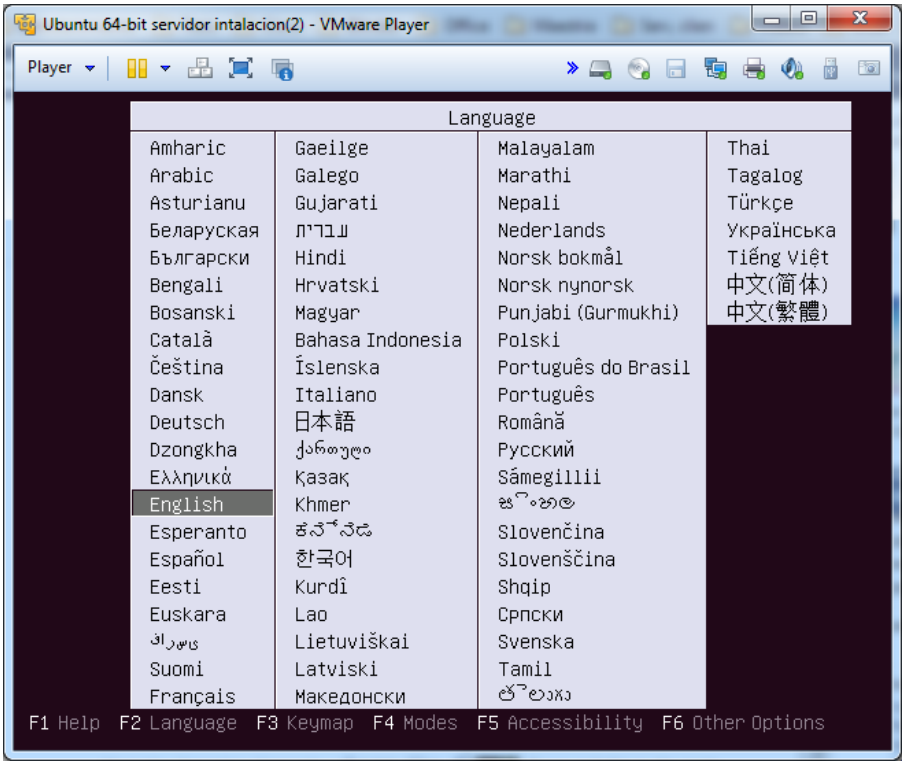
```
  BOOL            RemoteServerDisconnected = FALSE;
  BOOL            tcp_rx_thread_running;
  pthread_t       tcp_rx_thread;
  BOOL            udp_rx_thread_running;
  pthread_t       udp_rx_thread;
  INT             tcp_fd = -1;
  CHAR            rx_buffer[RX_BUFFER_SIZE];
  BOOL            rx_data_received;
  INT             udp_sock_fd;
  pthread_mutex_t  RxLock = PTHREAD_MUTEX_INITIALIZER;
#else
  // Exported functions & variables
  extern BOOL RemoteServerDisconnected;
  extern INT CreateTCPServerSocket( CHAR *ip, INT port );
  extern INT TCP_SendData( CHAR *ip, INT port, CHAR *buffer, INT
buffer_length );
  extern INT TCP_StartRXThread( VOID );
  extern INT TCP_StopRXThread( VOID );
#endif // SOCKET_C
#endif // SOCKET_H
```

ANEXO 2

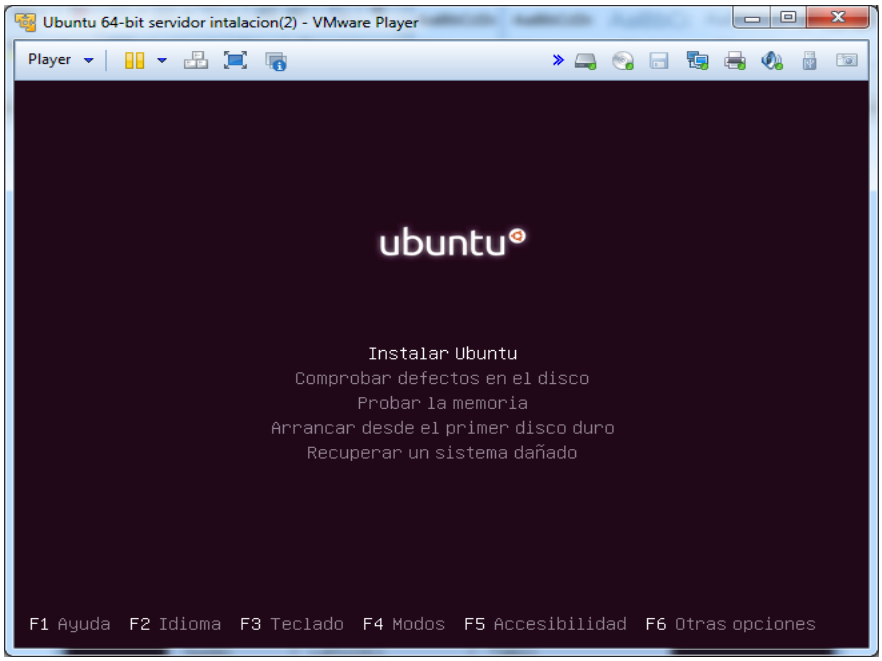
B. Instalación del Servidor WEB

Instalación del Sistema Operativo del Servidor

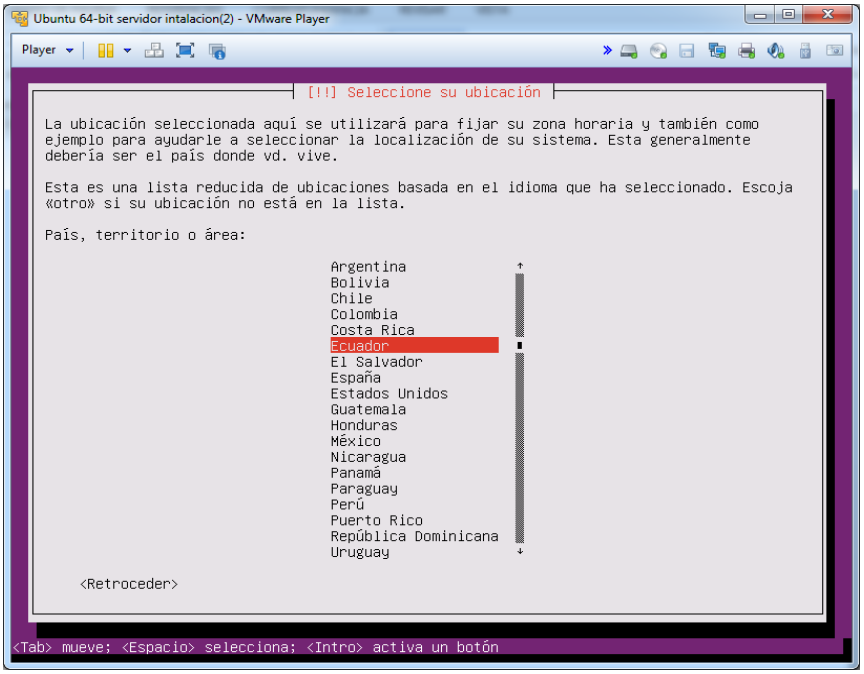
Primero se debe escoger el idioma, a lo cual se puede escoger el que se desee y para nuestro caso seleccionamos español.



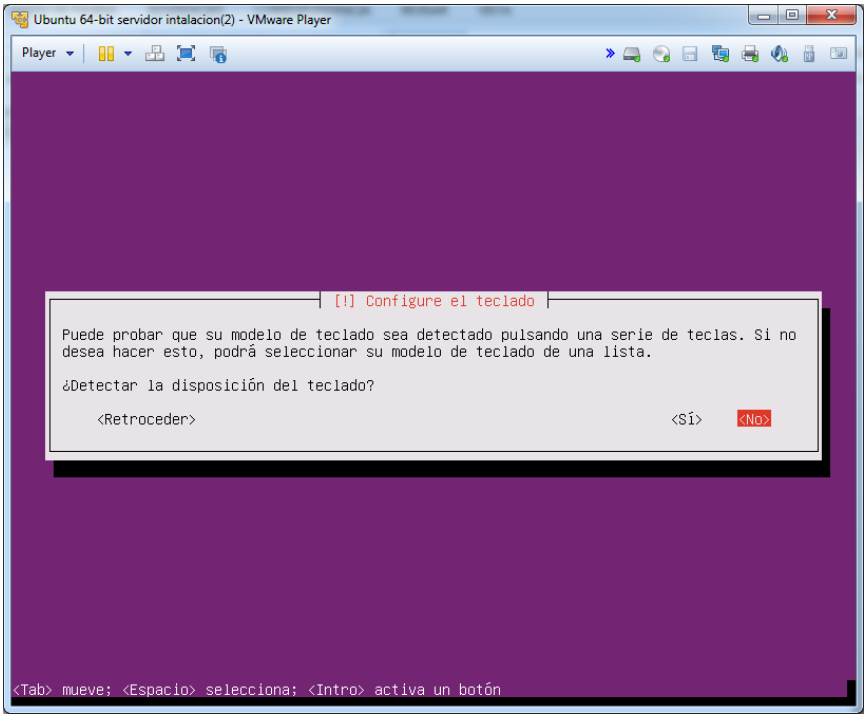
Seleccionamos Instalar Ubutu.



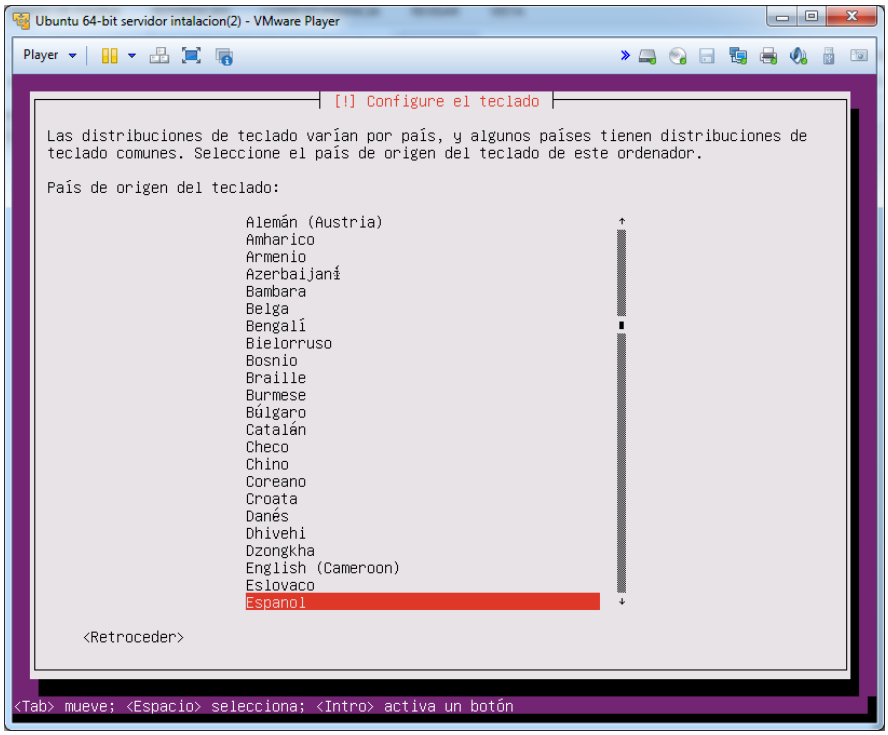
Seleccionamos el país, que en nuestro caso es Ecuador.



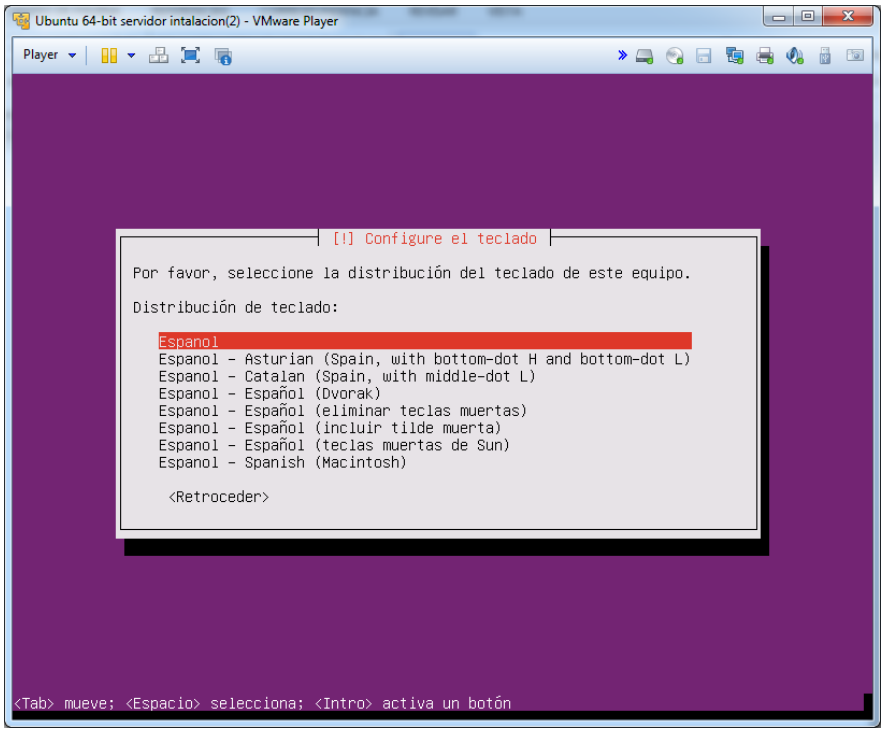
Si se desea probar el teclado poner en “SI” y en caso de pasar este paso poner “NO”.



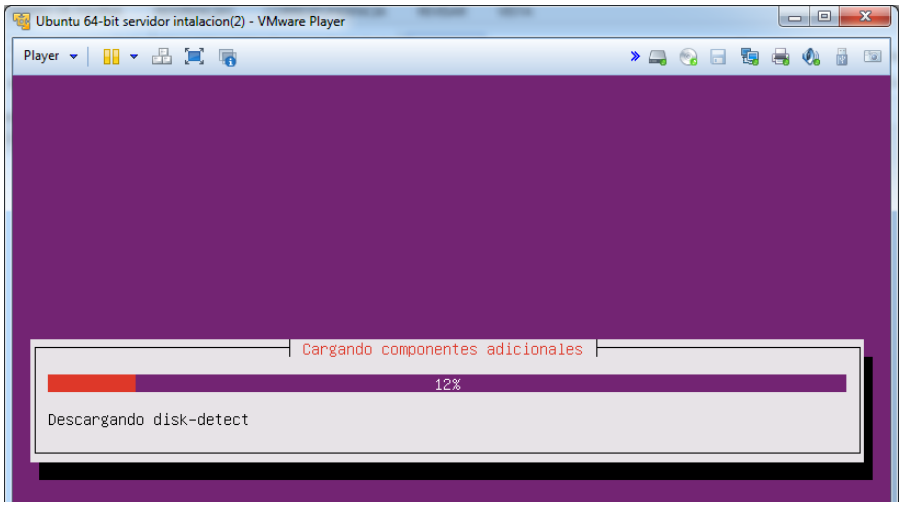
Se selecciona el tipo de distribución para el teclado.



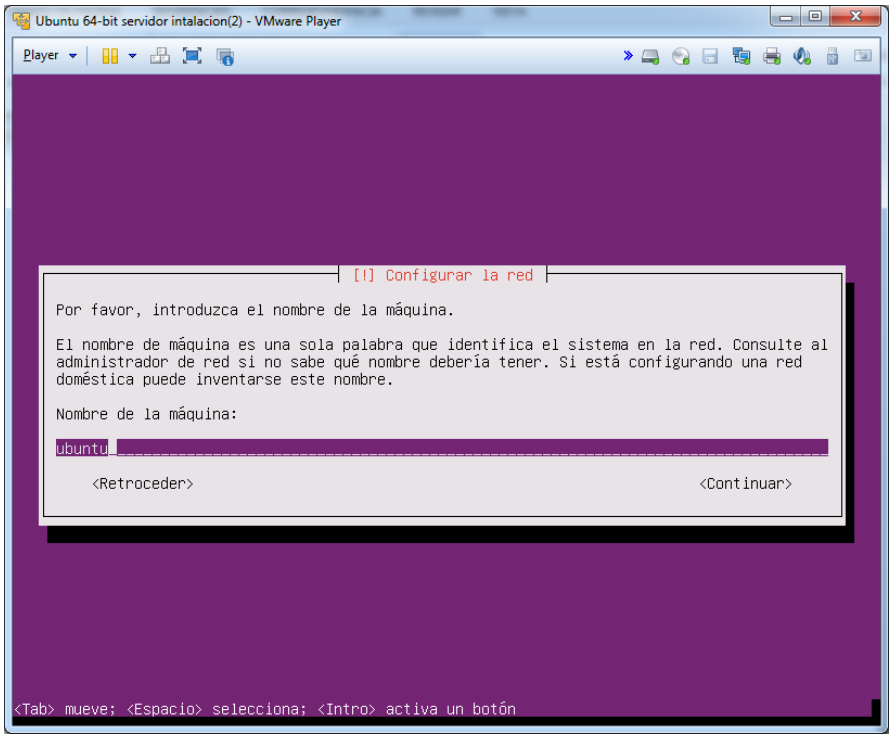
En este caso se selecciona Español.



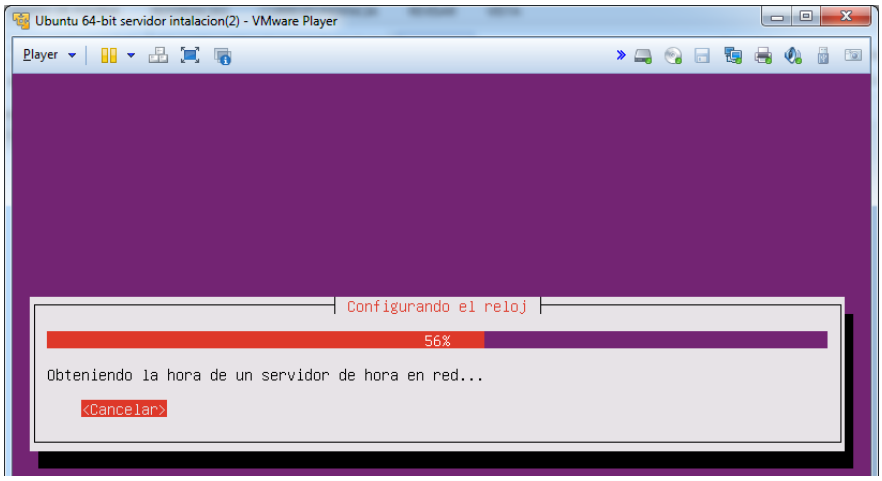
Muestra cómo se cargan los componentes adicionales para la instalación.



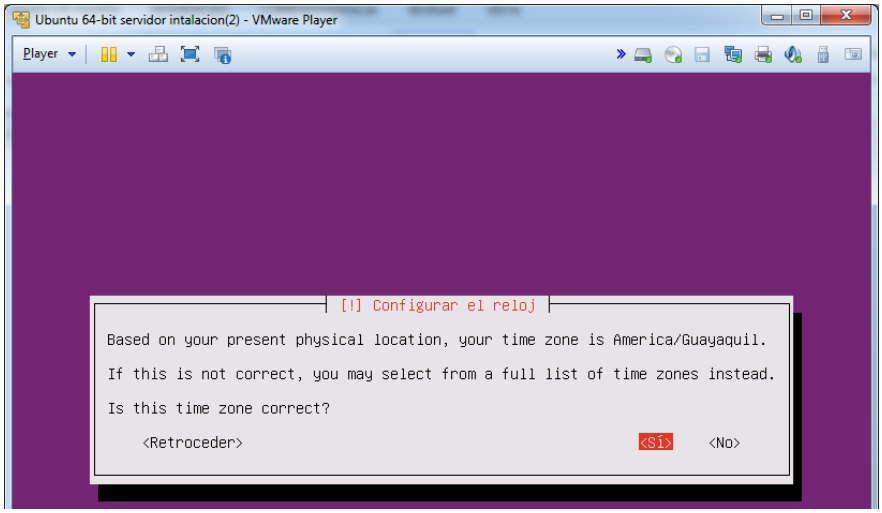
Se pone un nombre para el equipo.



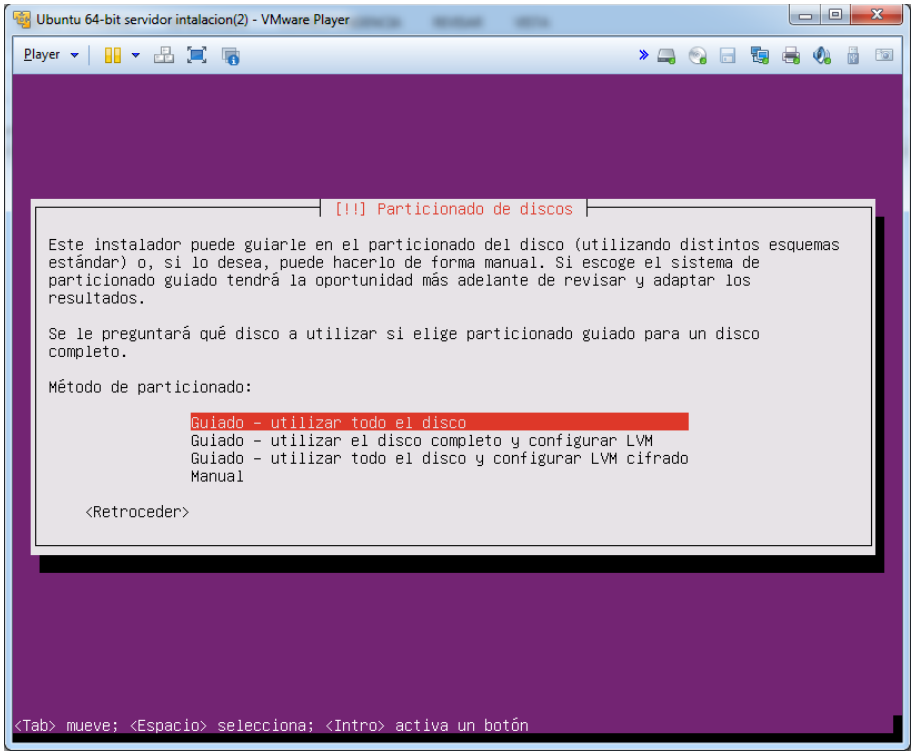
Configuración automática de los elementos del equipo.



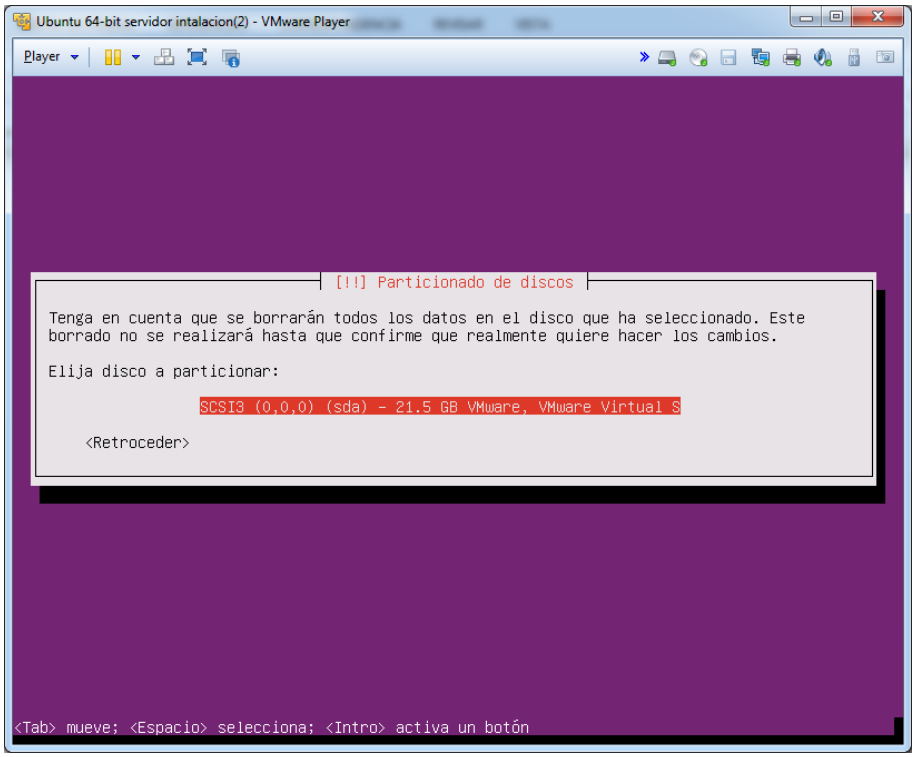
Basado en la configuración del internet, escoge una zona horaria para el reloj del equipo. En caso de estar correcto “SI”, sino colocar la zona horaria correcta.



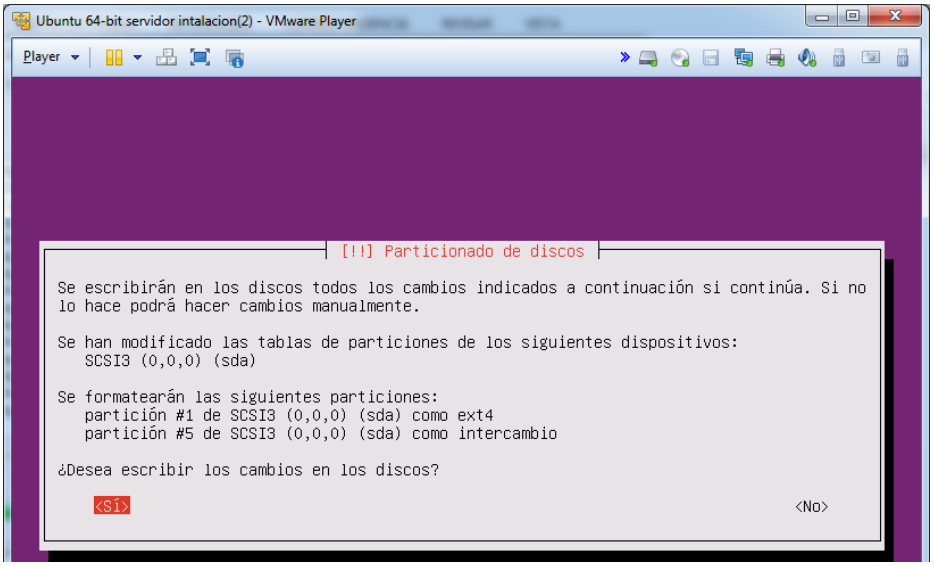
Se configura el tipo de particionado de los discos y se selecciona para utilizar todo el disco.



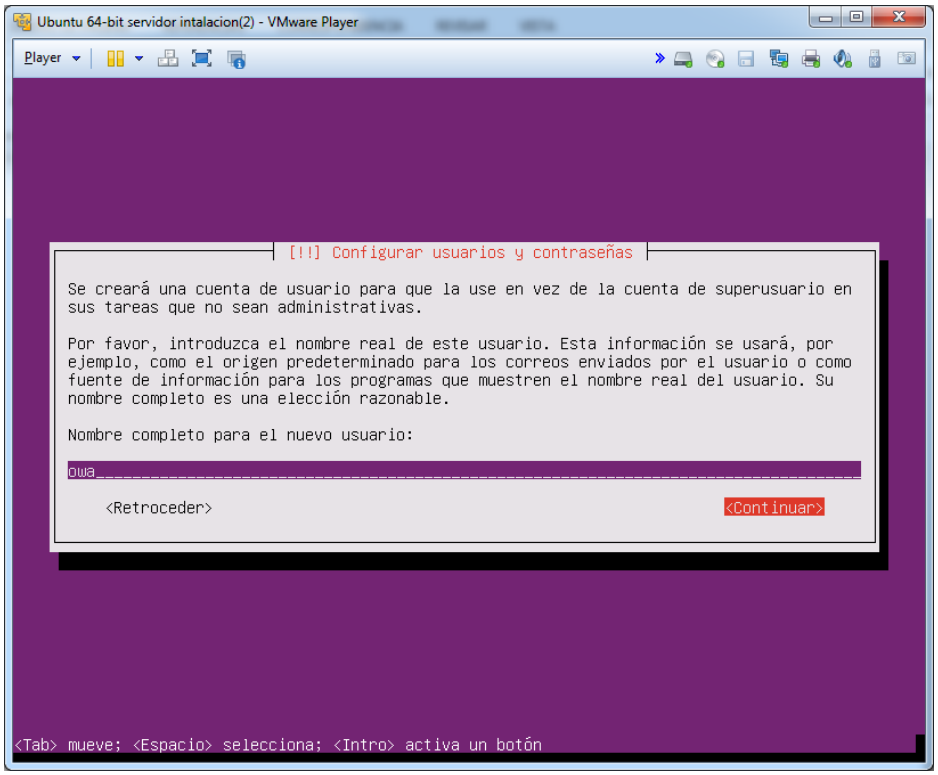
Se elige el disco.



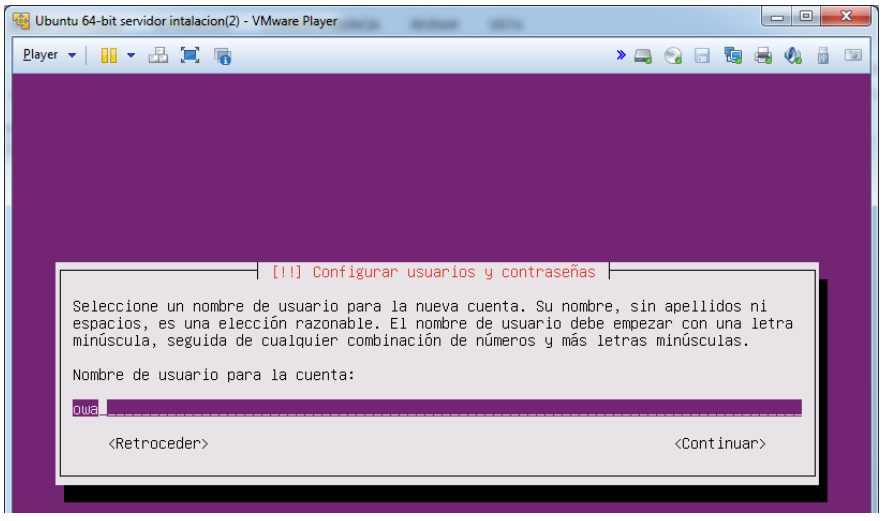
Se selecciona sobrescribir los cambios sobre los disco.



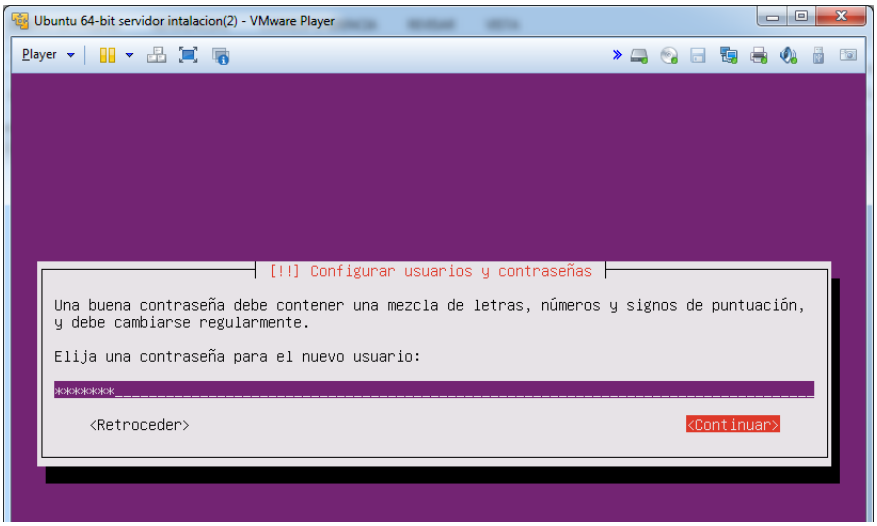
Se pone un nombre completo para el nuevo usuario.



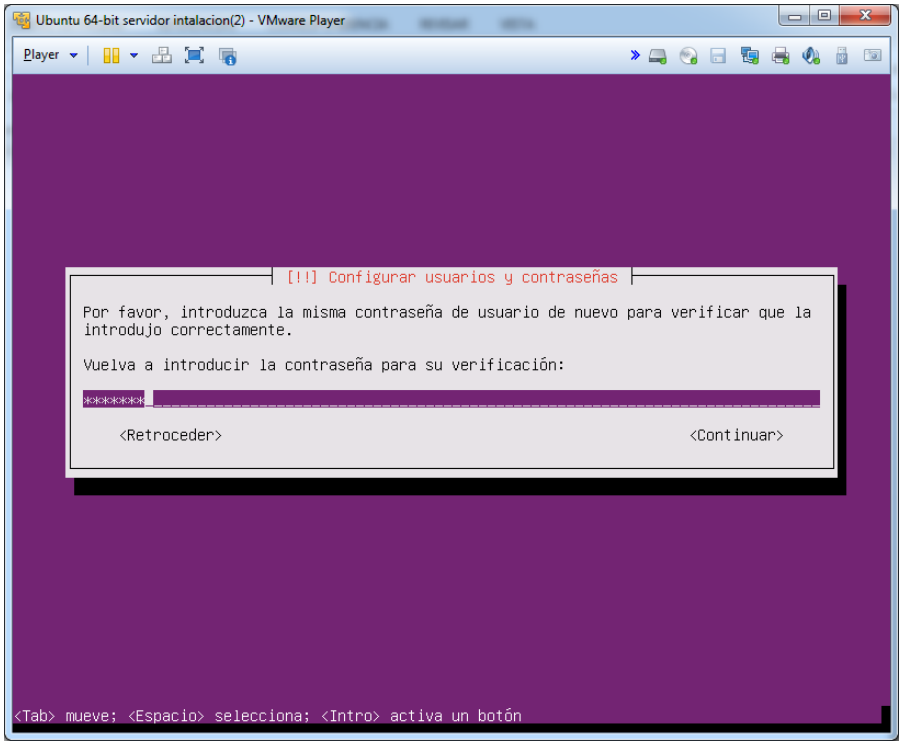
Se pone un nombre de usuario para la cuenta en la se que se puede poner el mismo que la anterior.



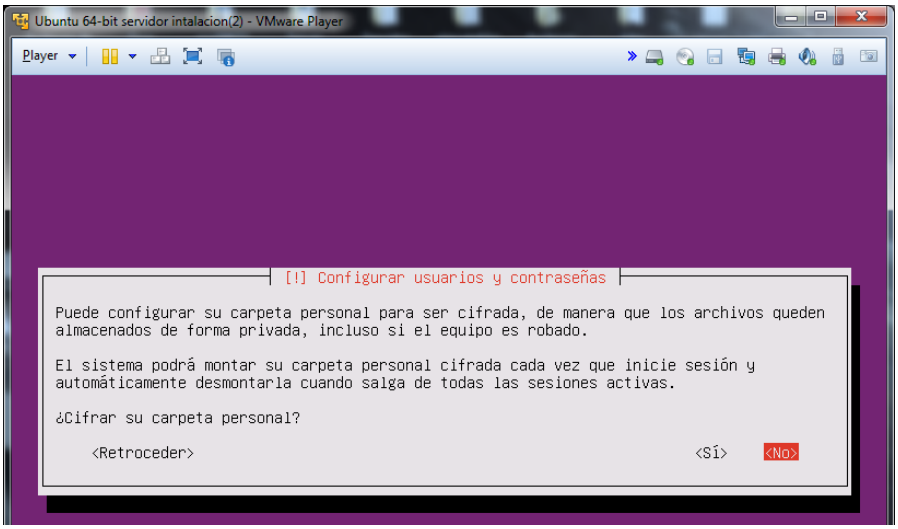
Se coloca una contraseña.



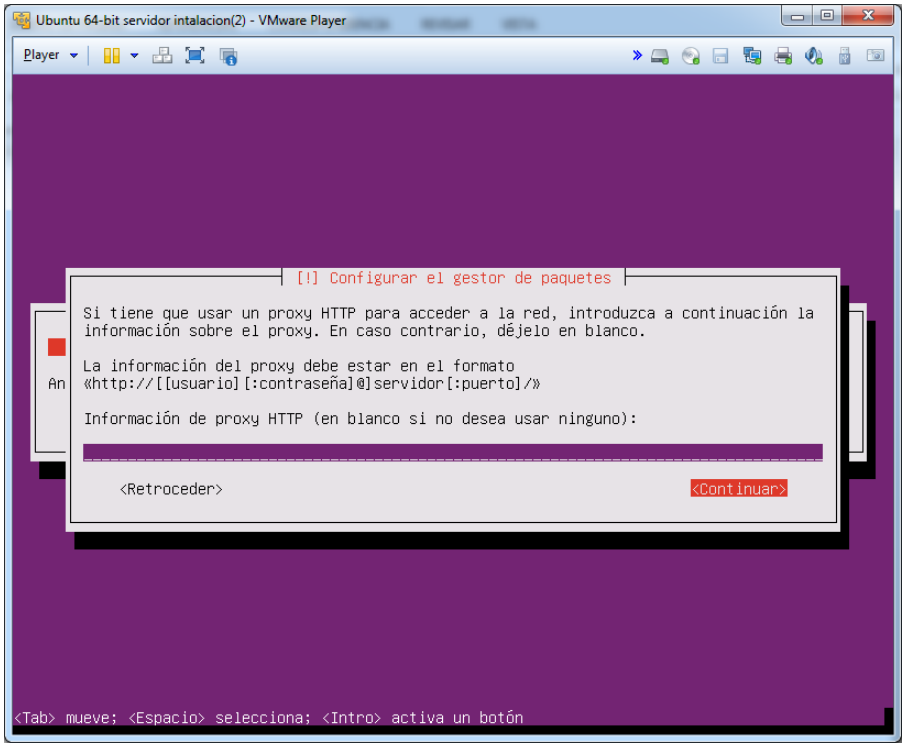
Se coloca de nuevo la contraseña para verificación.



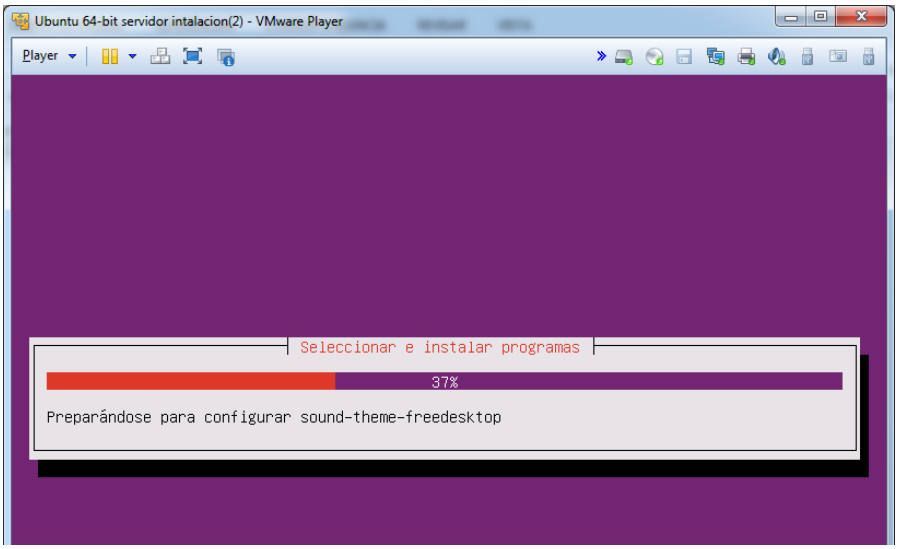
No necesitamos un carpeta cifrada, por lo tanto, se pone “NO”.



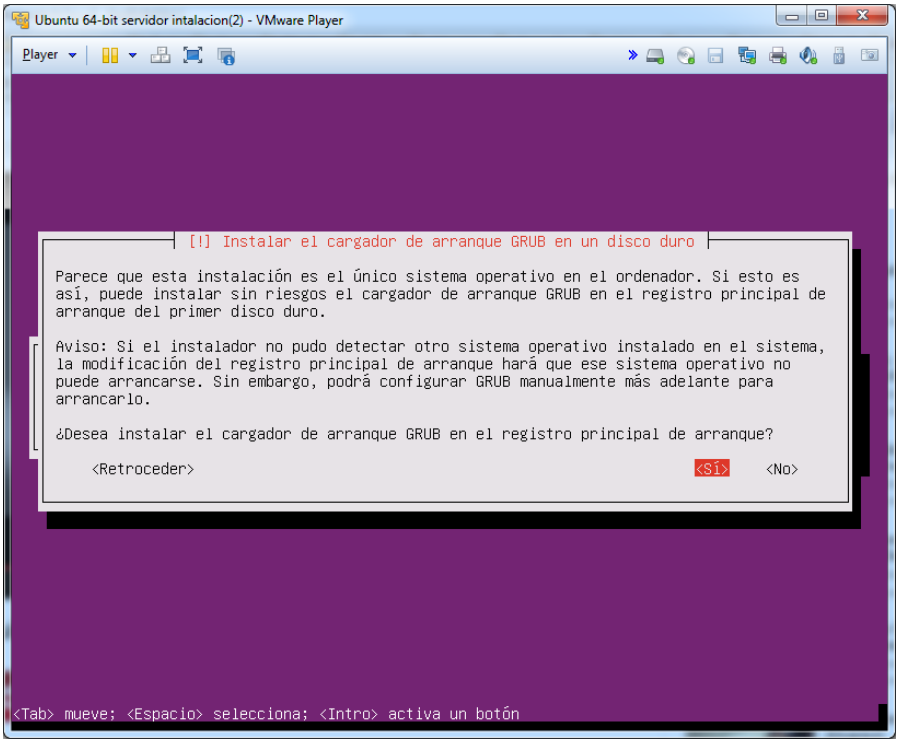
En caso de usar servidor proxy, ingresarlo. Caso contrario poner continuar.



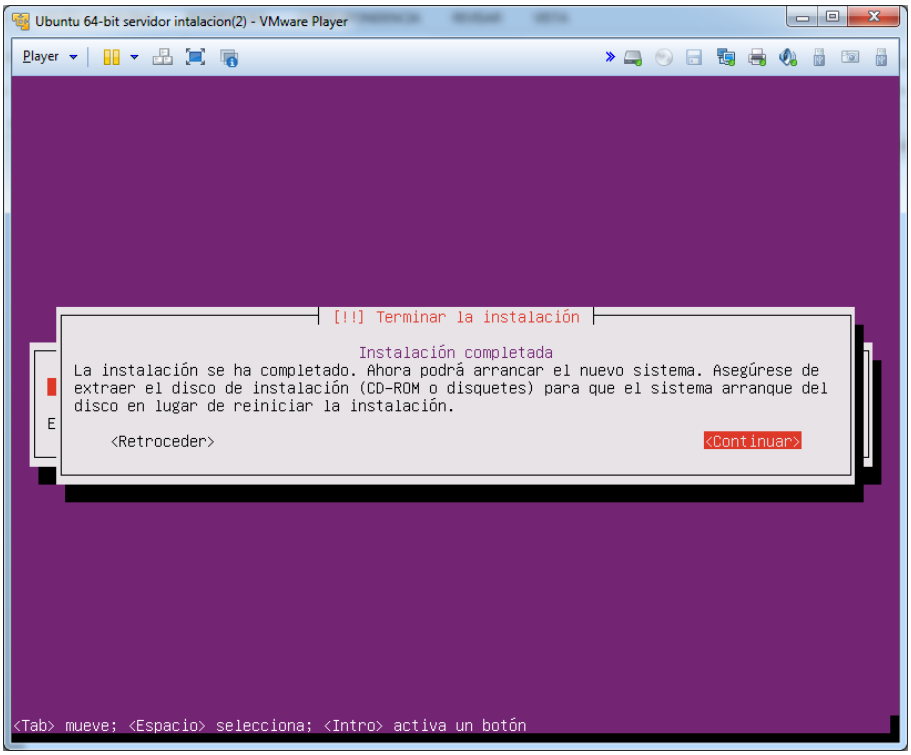
Instala todos los programas.



Es recomendable que se instale el cargador de Grub.



Por último se selecciona “Continuar” para terminar con la instalación.



Instalación del Lamp Ubuntu

El Lamp está conformado por tres partes importantes que son:

- Apache
- MySQL
- PHP

Los cuáles serán instalados de la siguiente manera.

Para la instalación de los archivos se debe tener abierto un terminal, y antes de instalar los paquetes se pone:

$ sudo apt-get update

Para instalar Apache:

$ sudo apt-get install apache2

Para instalar MySQL:

$ sudo apt-get install mysql-server

Para instalar PHP5:

$ sudo apt-get install php5 libapache2-mod-php5 php5-cli php5-mysql

Una vez instalados los complementos, se procede a hacer la comprobación para lo cual en el terminal se crea un archivo:

$ sudo gedit /var/www/test.php

Dentro del archivo se escribe:

```
<?php
phpinfo();
?>
```

Una vez hecho esto, se abre un browser y se escribe la siguiente dirección:

http://localhost/test.php

Al asomar la información con el PHP, quiere decir que está correctamente instalado.

ANEXO 3

C. Cambio de Puerto del Servidor WEB

Abrir el fichero **etc/apache2/ports.conf**, puede hacerlo con el editor VI o con gedit.

$ sudo vi etc/apache2/ports.conf

Buscar la línea con el texto:

Listen 80

En lugar de 80 se coloca el puerto que se desea y que en nuestro caso es el 9090.

Ahora de igual manera procedemos a abrir otro archivo con el siguiente comando:

$sudo etc/apache2/sites-enabled/000-default

Buscar:

<VirtualHost *:80>

En lugar de 80 se colocará 9090 que es el puerto a utilizar para el servidor. Por último queda en reiniciar apache con el siguiente comando:

$service apache2 restart

ANEXO 4

D. Archivos de Programación PHP y HTML

Index.php

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>OWASYS</title>
</head>
<body background="inicio.gif">
<br/><br/><br/><br/><br/><br/><br/><br/>
<form method='POST' action='index2.php'>
<h1 align="center"><strong>INGRESO</strong></h1>
<p align="center">Usuario    <input type='text'
name='usuario'/><p/>
<p align="center">Password <input type='password' name='clave'/><p/>
<p align="center"><input type='submit' value='Ingresar'/>
</form>
</body>
</html>
```

Index2.php

```
<?php
include ('funciones.php');
//usuario y clave pasados por el formulario
$usuario = $_POST['usuario'];
$clave = $_POST['clave'];
//usa la funcion conexiones() que se ubica dentro de funciones.php
if (conexiones($usuario, $clave)){
        //si es valido accedemos a ingreso.php
        header('Location:ingreso.php');
} else {
        //si no es valido volvemos al formulario inicial
        header('Location:index.php');
}
?>
```

Ingreso.php

```
<?php
include ('funciones.php');
//uso de la funcion verificar_usuario()
if (verificar_usuario()){
        //si el usuario es verificado puede acceder al contenido permitido a el
        print "Hola $_SESSION[usuario]<br/>Ingresa a otra parte del sistema
si deseas <a href='plan/home.php'>clic aqui</a><br/>";
        print "Desconectarse <a href='salir.php'/>aqui</a>";

} else {
        //si el usuario no es verificado volvera al formulario de ingreso
        header('Location:index.php');
}

if(is_readable('DbConnection.php')){
        require 'DbConnection.php';
        }else{
        throw new RuntimeException('No se pudo incluir Dbconnection.php
<br>');
        }

         $color_row=array('#cccccc', 'lightblue');
    $ind_color=0;
    $sql = "SELECT * FROM tablas";
    $result = array();
    $Obd = new DbConnection('localhost','root','gabogabo','datos');

    $Obd->connect();
    $result = $Obd->getAllRows($sql);
    $Obd->disconnect();
          echo " <table border=1 align='center'>";
echo " <tr>
<td>Fecha</td>
<td>Estado</td>
</tr>
          ";
        foreach ($result as $clave=>$key)
     {
        $ind_color++;
        $ind_color %= 2;
      echo"<tr bgcolor=${color_row[$ind_color]}>";
      echo"<td>".$key['fecha']."</td>";
echo"<td>".$key['valor']."</td>";
echo "</tr>";
     }
echo "</table>";
?>
```

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title></title>
<script type="text/javascript" src='jquery-1.7.1.min.js' ></script>
<script type="text/javascript" >
$(function(){
$("#receptor").load("grid.php",{accion: "sample1"});
$("#boton1").click(function() {
        $("#receptor").text("Espere....");
        $.post("index.php",{accion:"ejemplo2",var1:$("#texto1").val()},respons
e,'json')
});
});
function response(myvar){
                        $("#receptor").html(myvar.texto + "<br>" +
myvar.otrapropiedad);
                }
</script>
<style type="text/css">
.div1{
        border:1px solid red;
        width:200px;
        margin:0 auto;
        }
</style>
</head>
<body>
<div id="receptor"></div>
<!--<input type="button" id="boton1" value="Pinta">
<input type="text" id="texto1"> -->
</body>
</html>
```

Alarmas.php

```
<?php
//$lat = -0.206809 ; //mysql_query("SELECT latitud FROM `localizacion`");
$long = -78.508500; //mysql_query("SELECT longitud FROM `localizacion`");
$con = mysql_connect("localhost", "root", "gabogabo");
if (!$con) { die("Error: " . mysql_error());}
mysql_select_db("pruebas", $con);
$result = mysql_query("SELECT * FROM `posicion` ORDER BY
`posicion`.`fecha` DESC LIMIT 1");
$lat = mysql_result($result, 0, 1);
$long = mysql_result($result, 0, 2);
//echo $lat;echo $long;
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>OWA-TESIS</title>
<link href="menu.css" rel="stylesheet" type="text/css" />
<script src="http://www.openlayers.org/api/OpenLayers.js"></script>
<script
src="http://www.openstreetmap.org/openlayers/OpenStreetMap.js"></script>
<script type="text/javascript">
var zoom=16;
                var map; //complex object of type OpenLayers.Map
var lat="<?php echo $lat; ?>";
            //document.write("VariableJS = " + lat);
                var lon="<?php echo $long; ?>";
                        //document.write("VariableJS = " + lon);
                function init() {
                        map = new OpenLayers.Map ("map", {
                                controls:[
                                        new OpenLayers.Control.Navigation(),
                                        new OpenLayers.Control.PanZoomBar(),
                                        new OpenLayers.Control.LayerSwitcher(),
                                        new OpenLayers.Control.Attribution()],
                                maxExtent: new OpenLayers.Bounds(-
20037508.34,-20037508.34,20037508.34,20037508.34),
                                maxResolution: 156543.0399,
                                numZoomLevels: 19,
                                units: 'm',
                                projection: new
OpenLayers.Projection("EPSG:900913"),
                                displayProjection: new
OpenLayers.Projection("EPSG:4326")
                        } );
                        // Define the map layer
```

```
                    // Here we use a predefined layer that will be kept up to
date with URL changes
                    layerMapnik = new
OpenLayers.Layer.OSM.Mapnik("Mapnik");
                    map.addLayer(layerMapnik);
                    layerCycleMap = new
OpenLayers.Layer.OSM.CycleMap("CycleMap");
                    map.addLayer(layerCycleMap);
                    layerMarkers = new
OpenLayers.Layer.Markers("Markers");
                    map.addLayer(layerMarkers);
                    // Add the Layer with the GPX Track
                    var lgpx = new OpenLayers.Layer.Vector("Lakeside
cycle ride", {
                        strategies: [new OpenLayers.Strategy.Fixed()],
                        protocol: new OpenLayers.Protocol.HTTP({
                            url: "/prueba2/around_lake.gpx",
                            format: new OpenLayers.Format.GPX()
                        }),
                        style: {strokeColor: "green", strokeWidth: 5,
strokeOpacity: 0.5},
                        projection: new
OpenLayers.Projection("EPSG:4326")
                    });
                    map.addLayer(lgpx);
                    var lonLat = new OpenLayers.LonLat(lon,
lat).transform(new OpenLayers.Projection("EPSG:4326"),
map.getProjectionObject());
                    map.setCenter(lonLat, zoom);
                    var size = new OpenLayers.Size(21, 25);
                    var offset = new OpenLayers.Pixel(-(size.w/2), -size.h);
                    //var icon = new
OpenLayers.Icon('http://www.openstreetmap.org/openlayers/img/marker.png'
,size,offset);
                var icon = new OpenLayers.Icon('images/car.png',size,offset);
                    layerMarkers.addMarker(new
OpenLayers.Marker(lonLat,icon));
                //setTimeout('', 1000);
                }
</script>
<body bgcolor="#CCCCCC" onload="init();">
<form id="form1" name="form1" method="post" action="">
<span class="titulo"><CENTER spry:hover="titulo">SISTEMA DE
MONITORIZACION</CENTER></span>
</form>
        <br>
<div id="navbar">
<div id="holder">
<ul>
<li><a href="home.php">Home</a></li>
```

```
<li><a href="localizacion.php">Localizacion</a></li>
<li><a href="alarmas.php" id="onlink">Mapa</a></li>
<li><a href="historial.php">Historial</a></li>
<li><a href="contactos.php">Alarmas</a></li></ul></div>
<!-- end navbar div -->
</div><!-- end holder div -->
<!--<iframe width="600" height="350" frameborder="0" scrolling="no"
marginheight="0" marginwidth="0"
src="http://maps.google.es/?ie=UTF8&amp;t=m&amp;ll=-0.201187,-
78.498001&amp;spn=0.240324,0.411987&amp;z=11&amp;output=embed">
</iframe><br /><small><a
href="http://maps.google.es/?ie=UTF8&amp;t=m&amp;ll=-0.201187,-
78.498001&amp;spn=0.240324,0.411987&amp;z=11&amp;source=embed"
style="color:#0000FF;text-align:left">Ver mapa mÃ¡s grande</a></small>-->
<!--<iframe width="425" height="350" frameborder="0" scrolling="no"
marginheight="0" marginwidth="0"
src="https://maps.google.es/?ie=UTF8&amp;t=m&amp;source=embed&amp;
ll=-0.213546,-
78.506584&amp;spn=0.188826,0.454903&amp;z=12&amp;output=embed">
</iframe>
<iframe width="425" height="350" frameborder="0" scrolling="no"
marginheight="0" marginwidth="0"
src="http://www.openstreetmap.org/export/embed.html?bbox=-78.49328,-
0.16915,-78.47605,-0.156&amp;layer=mapnik&amp;marker=-0.16258,-
78.48379" style="border: 1px solid black"></iframe>-->
<div id="map" style="width: 1319px; height: 523px; position: absolute; left:
13px; top: 148px;" ></div>
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><b
r><br><br><br><br><br><br><br><br><br>
</body>
</html>
```

Historial.php

```
<?php
$con = mysql_connect("localhost", "root", "gabogabo");
if (!$con) {
   die("Error: " . mysql_error());
}
mysql_select_db("pruebas", $con);
$result = mysql_query("SELECT * FROM temperatura");
?>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>OWA-TESIS</title>
<link href="menu.css" rel="stylesheet" type="text/css" />
```

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<script src="media/js/jquery.js" type="text/javascript"></script>
<script src="media/js/jquery.dataTables.js" type="text/javascript"></script>
<style type="text/css">
        @import "media/css/demo_table_jui.css";
        @import "media/themes/smoothness/jquery-ui-1.8.4.custom.css";
</style>
<style>
        *{
           font-family: arial;
        }
</style>
<script type="text/javascript" charset="utf-8">
        $(document).ready(function(){
           $('#datatables').dataTable({
              "sPaginationType":"full_numbers",
              "aaSorting":[[0, "desc"]],
              "bJQueryUI":true
           });
        });
</script>
</head>
<body bgcolor="#CCCCCC">
<form id="form1" name="form1" method="post" action="">
<span class="titulo"><CENTER spry:hover="titulo">SISTEMA DE
MONITORIZACION</CENTER></span>
</form>
<br>
<div id="navbar">
<div id="holder">
<ul>
<li><a href="home.php">Home</a></li>
<li><a href="localizacion.php">Localizacion</a></li>
<li><a href="alarmas.php">Mapa</a></li>
<li><a href="historial.php" id="onlink">Historial</a></li>
<li><a href="contactos.php">Alarmas</a></li>
</ul>
</div>
<!-- end navbar div -->
</div><!-- end holder div -->
<div>
<table id="datatables" class="display">
<thead>
<tr>
<th>FECHA Y HORA</th>
<th>TEMPERATURA</th>
</tr>
</thead>
<tbody>
<?php
```

```
                while ($row = mysql_fetch_array($result)) {
                   ?>
<tr>
<td><?=$row['fecha']?></td>
<td><?=$row['temp']?></td>
</tr>
<?php
                }
                ?>
</tbody>
</table>
</div>
</body>
</html>
```

Home.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>OWA-TESIS</title>
<link href="menu.css" rel="stylesheet" type="text/css" />
<script src="Scripts/swfobject_modified.js" type="text/javascript"></script>
</head>
<body bgcolor="#CCCCCC">
<form id="form1" name="form1" method="post" action="">
<span class="titulo"><CENTER spry:hover="titulo">SISTEMA DE
MONITORIZACION</CENTER></span>
</form>
<br>
<div id="navbar">
<div id="holder">
<ul>
<li><a href="home.php" id="onlink">Home</a></li>
<li><a href="localizacion.php">Localizacion</a></li>
<li><a href="alarmas.php">Mapa</a></li>
<li><a href="historial.php">Historial</a></li>
<li><a href="contactos.php">Alarmas</a></li>
</ul>
</div><!-- end navbar div -->
</div><!-- end holder div -->
<p>
<form id="form2" name="form2" method="post" action="">
<table width="1029" height="480" border="0">
<tr>
<th height="35" colspan="2" scope="col">TECNOLOGÃ?A M2M</th>
```

<th width="13" scope="col"> </th>
<th width="554" scope="col">COMUNICACIONES M2M</th>
</tr>
<tr>
<td colspan="2">
Controlar su domicilio o su empresa a distancia, comprobar la seguridad de sus hijos o la ubicaciÃ³n de un vehÃculo... O incluso asegurarse de que su animal de compaÃ±Ãa no se pierdaâ€¦ Todo esto es posible gracias a laÂ tecnologÃa M2M (MÃ¡quina a MÃ¡quina), que permite a los objetos comunicarse entre sÃ y abre nuevas perspectivas para los servicios del futuro. Actualmente, son las empresas las que estÃ¡n mÃ¡s interesadas en el desarrollo de esta tecnologÃa y sus aplicaciones.

<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" width="434" height="250" id="FLVPlayer">
<param name="movie" value="FLVPlayer_Progressive.swf" />
<param name="quality" value="high" />
<param name="wmode" value="opaque" />
<param name="scale" value="noscale" />
<param name="salign" value="lt" />
<param name="FlashVars" value="&MM_ComponentVersion=1&skinName=Clear_Skin_1&streamName=images/video&autoPlay=true&autoRewind=true" />
<param name="swfversion" value="8,0,0,0" />
<!-- Esta etiqueta param indica a los usuarios de Flash Player 6.0 r65 o posterior que descarguen la versiÃ³n mÃ¡s reciente de Flash Player. ElimÃ-nela si no desea que los usuarios vean el mensaje. -->
<param name="expressinstall" value="Scripts/expressInstall.swf" />
<!-- La siguiente etiqueta object es para navegadores distintos de IE. OcÃºltela a IE mediante IECC. -->
<!--[if !IE]>-->
<object type="application/x-shockwave-flash" data="FLVPlayer_Progressive.swf" width="434" height="250">
<!--<![endif]-->
<param name="quality" value="high" />
<param name="wmode" value="opaque" />
<param name="scale" value="noscale" />
<param name="salign" value="lt" />
<param name="FlashVars" value="&MM_ComponentVersion=1&skinName=Clear_Skin_1&streamName=images/video&autoPlay=true&autoRewind=true" />
<param name="swfversion" value="8,0,0,0" />
<param name="expressinstall" value="Scripts/expressInstall.swf" />
<!-- El navegador muestra el siguiente contenido alternativo para usuarios con Flash Player 6.0 o versiones anteriores. -->

<h4>El contenido de esta pÃ¡gina requiere una versiÃ³n mÃ¡s reciente de Adobe Flash Player.</h4>

```
<p><a href="http://www.adobe.com/go/getflashplayer"><img
src="http://www.adobe.com/images/shared/download_buttons/get_flash_player.gif" alt="Obtener Adobe Flash Player" /></a></p>
</div>
<!--[if !IE]>-->
</object>
<!--<![endif]-->
</object>
</p>
</div></td>
<td> </td>
<td><div align="center">
<p><img src="images/m2m.png" width="529" height="245" alt="" /></p>
<p align="justify"> La comunicaciÃ³n M2M (mÃ¡quina a mÃ¡quina) es un sistema informÃ¡tico que permite el intercambio de datos entre dos mÃ¡quinas remotas. Esta comunicaciÃ³n entre dispositivos tiene como objetivo el control y la supervisiÃ³n de diferentes procesos automatizados. Los aparatos con capacidad M2M se suelen usar en los servicios que implican telemetrÃa o telecontrol, como alarmas domÃ©sticas, los contadores de agua y gas, los paneles informativos en carreteras, los terminales punto de venta (TPV), las mÃ¡quinas de &amp;quot;vending&amp;quot; de las oficinas, etc.</p>
</div></td>
</tr>
<tr>
<td width="429"><p>
<label for="textfield"></label>
</p></td>
<td width="15"> </td>
<td> </td>
<td> </td>
</tr>
</table>
</form>
<br>
<script type="text/javascript">
swfobject.registerObject("FLVPlayer");
</script>
</body>
</html>
```

Localización.php

```
<?php
$con = mysql_connect("localhost", "root", "gabogabo");
if (!$con) {
    die("Error: " . mysql_error());
}
```

```
mysql_select_db("pruebas", $con);
$result = mysql_query("SELECT * FROM localizacion");
?>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>OWA-TESIS</title>
<link href="menu.css" rel="stylesheet" type="text/css" />
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<script src="media/js/jquery.js" type="text/javascript"></script>
<style type="text/css">
        @import "media/css/demo_table_jui.css";
        @import "media/themes/smoothness/jquery-ui-1.8.4.custom.css";
</style>
<style>
        *{
           font-family: arial;
        }
</style>
<script type="text/javascript" charset="utf-8">
        $(document).ready(function(){
           $('#datatables').dataTable({
              "sPaginationType":"full_numbers",
              "aaSorting":[[1, "desc"]],
              "bJQueryUI":true
           });
        });
</script>
</head>
<body bgcolor="#CCCCCC">
<form id="form1" name="form1" method="post" action="">
<span class="titulo"><CENTER spry:hover="titulo">SISTEMA DE
MONITORIZACION</CENTER></span>
</form>
        <br>
<div id="navbar">
<div id="holder">
<ul>
<li><a href="home.php">Home</a></li>
<li><a href="localizacion.php" id="onlink">Localizacion</a></li>
<li><a href="alarmas.php">Mapa</a></li>
<li><a href="historial.php">Historial</a></li>
<li><a href="contactos.php">Alarmas</a></li>
</ul>
</div> <!-- end navbar div -->
</div><!-- end holder div -->
<div>
<table id="datatables" class="display">
<thead>
```

```
<tr>
<th>NOMBRE</th>
                        <th>FECHA Y HORA</th>
                        <th>LATITUD</th>
                        <th>LONGITUD</th>
</tr>
</thead>
<tbody>
<?php
                while ($row = mysql_fetch_array($result)) {
                   ?>
<tr>
<td><?=$row['nombre']?></td>
<td><?=$row['fecha']?></td>
<td><?=$row['latitud']?></td>
<td><?=$row['longitud']?></td>
</tr>
<?php
                }
                ?>
</tbody>
</table>
</div>
<!--<iframe width="600" height="350" frameborder="0" scrolling="no"
marginheight="0" marginwidth="0"
src="http://maps.google.es/?ie=UTF8&amp;t=m&amp;ll=-0.201187,-
78.498001&amp;spn=0.240324,0.411987&amp;z=11&amp;output=embed">
</iframe>-->
<!--<iframe width="425" height="350" frameborder="0" scrolling="no"
marginheight="0" marginwidth="0"
src="http://www.openstreetmap.org/export/embed.html?bbox=-78.50328,-
0.17604,-78.46882,-0.14973&amp;layer=mapquest&amp;marker=-0.16258,-
78.48379" style="border: 1px solid black"></iframe>-->
</body>
</html>
```

ANEXO 5

E. Socket Servidor

Socket_Servidor

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
void dostuff(int);
void error(const char *msg)
 {
 perror(msg);
 exit(1);
 }
int main(int argc, char *argv[])
{
 int sockfd, newsockfd, portno, pid;
 socklen_t clilen;
 struct sockaddr_in serv_addr, cli_addr;
if (argc < 2)
 {
 fprintf(stderr,"ERROR, no hay ningún puerto\n");
exit(1);
 }
 sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd < 0)
 error("ERROR de apertura de socket");
bzero((char *) &serv_addr, sizeof(serv_addr));
 portno = atoi(argv[1]);
 serv_addr.sin_family = AF_INET;
 serv_addr.sin_addr.s_addr = INADDR_ANY;
 serv_addr.sin_port = htons(portno);
 if (bind(sockfd, (struct sockaddr *) &serv_addr,sizeof(serv_addr)) < 0)
error("ERROR en la conexión");
listen(sockfd,5);
 clilen = sizeof(cli_addr);
 while (1)
 {
 newsockfd = accept(sockfd,(struct sockaddr *) &cli_addr, &clilen);
if (newsockfd < 0)
 error("ERROR en aceptar");
```

```
 pid = fork();
 if (pid < 0)
 error("ERROR en fork");
if (pid == 0)
 {
 close(sockfd);
 dostuff(newsockfd);
 exit(0);
 }
 else
 close(newsockfd);
 }
 close(sockfd);
return 0;
}
/*Gestiona todas las comunicaciones una vez que la conexión ha sido
establecida*/
void dostuff (int sock)
 {
int n;
 char buffer[256];
 bzero(buffer,256);
 n = read(sock,buffer,255);
if (n < 0)
error("ERROR al leer del socket");
 printf("Aquí está el mensaje: %s\n",buffer);
 n = write(sock,"Tengo el mensaje",18);
 if (n < 0)
 error("ERROR al escribir en el socket");
}
```

ANEXO 6

F. MySQL con C/C++

Datos.c

```
#include <mysql++/mysql++.h>
#include <iostream>
using namespace std;
int main (){
  mysqlpp::Connection conn(false);
char server[] = "localhost:9090";
char user[]   = "root";  // O cualquier usuario
char pass[]   = "pruebas";
  if (conn.connect(NULL, server, user, pass))
  {
     // string consulta = "select * from ins_institutos";
    string consulta="SHOW DATABASES";
     mysqlpp::Query query = conn.query(consulta);
     if (mysqlpp::StoreQueryResult res = query.store())
{
   cout << "Bases de datos encontradas: "<<res.num_rows()<<endl;
if (res.num_rows() > 0)
      {
        bool m2m=false;
        // cout <<res[0]["nombre"] << endl;
        for   (mysqlpp::StoreQueryResult::iterator   it=res.begin();   it!=res.end();
++it)
          {
        if ((it)->at(0)=="m2m")
          {
            m2m=true;
break;
          }
        // Si queremos visualizar las bases de datos...
        cout <<"Base de datos: "<< (it)->at(0) << endl; // Se puede hacer con
(*it)[0] pero la documentación dice que es más lento
          }
        if (m2m)
          {
        // Miramos los contenidos de la tabla
        cout << "La base de datos se encuentra" << endl;
query.reset();
        query<<"SELECT * FROM `m2m`.`temperatura`";
        res=query.store();
        if (res.num_rows() >0)
```

```
        {
         for (mysqlpp::StoreQueryResult::iterator it=res.begin(); it!=res.end();
++it)
           {
         for (unsigned i=0; i<res.num_fields(); i++)
cout <<res.field_name(i)<<" = "<< (it)->at(i) << endl; // Se puede hacer con
(*it)[0] pero la documentación dice que es más lento
          cout << endl;
           }
         }
        else
         cout << "No se han obtenido datos" << endl;
         }
        else
         {
        // Lo construimos todo
        cout << "Creando base de datos" << endl;
if (conn.create_db("m2m"))
{
          cout << "Seleccionando base de datos" << endl;
if (conn.select_db("m2m"))
{
          cout << "Creando tabla" << endl;
          query.reset();
query=conn.query("CREATE TABLE IF NOT EXISTS `m2m`.`temperatura` ("
                  "`temp` varchar(4) NOT NULL ,"
                  "`hora` datetime NOT NULL,"
                  "PRIMARY KEY (`temp`)"
                  ")      ENGINE=MyISAM      DEFAULT      CHARSET=latin1
AUTO_INCREMENT=1 ;");
          if (query.execute())
           {
            query.reset();
            query << "INSERT INTO `temperatura` VALUES ('27', '2013-06-08
14:30:40');";
             if(!query.execute())
               cout << "Fallo al introducir datos"<<endl;
             query.reset();
             query << "INSERT INTO `temperatura` VALUES ('29', '2013-06-08
14:36:40');";
             if(!query.execute())
cout << "Fallo al introducir datos"<<endl;
query.reset();
             //query << "INSERT INTO `apertura` VALUES ('1', '2013-06-08
14:38:40');";
//if(!query.execute())
               //cout << "Fallo al introducir datos"<<endl;
           }
          else
            cout << "Error al crear tabla"<<endl;
```

```
            }
           else
             cout << "Fallo al seleccionar la base de datos" << endl;
          }
         else
          cout << "Fallo al crear base de datos" << endl;
          }
       }
     else
      cout << "Fallo al obtener bases de datos" << endl;
     }
     conn.disconnect();
  }
  else
    cout <<"Fallo al conectarse a la BD: "<<conn.error()<<endl;
}
```