



**UNIVERSIDAD NACIONAL AUTÓNOMA  
DE MÉXICO**

---

**Facultad de Ingeniería**

APLICACIÓN DE POSICIONAMIENTO PARA TELÉFONO  
CELULAR A TRAVÉS DE UN DISPOSITIVO GPS

**T E S I S**  
QUE PARA OBTENER EL TÍTULO DE  
**INGENIERO EN COMPUTACIÓN**

PRESENTA:  
JORGE ALBERTO SOLANO GALVEZ

DIRECTOR DE TESIS:  
M. en C. ALEJANDRO VELÁZQUEZ MENA





Universidad Nacional  
Autónoma de México

Dirección General de Bibliotecas de la UNAM

**Biblioteca Central**



**UNAM – Dirección General de Bibliotecas**  
**Tesis Digitales**  
**Restricciones de uso**

**DERECHOS RESERVADOS ©**  
**PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL**

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

# Agradecimientos

A mis padres, por su amor, comprensión y respeto recibido durante toda mi vida, inspirándome siempre a seguir adelante y superarme. Porque siempre han dado todo por mí sin esperar nada a cambio, porque su respaldo y apoyo en todo lo que hago, por ser una familia.

A mi madre, por la paciencia, tiempo y dedicación que ha tenido para conmigo, cómo sólo una madre lo puede hacer. Por el apoyo que me ha brindado, por la educación y los valores inculcados, por los consejos siempre atinados, por ese cariño inmenso que siempre me ha mostrado. Por ser mi madre.

A mi padre, porque siempre he sentido su apoyo y cariño, por las enseñanzas recibidas, por las aventuras vividas. Por los sacrificios que ha realizado para llegar a donde estamos ahorita como familia.

A Alejandro V. Mena, por las enseñanzas recibidas día con día, por la exigencia que provoca superación, por la confianza que genera compromiso, pero, sobre todo, por su amistad.

A Julius y Cano, por sus enseñanzas, porque nunca me han negado ayuda y siempre he aprendido de ellos.

A Circe, por el apoyo, ayuda y tiempo que me ha brindado, así como por su amistad sincera e incondicional.

A Koko, porque sin las pruebas y las críticas constructivas, la aplicación no hubiera podido realizarse.

# Dedicatoria

A mis padres, ya que este trabajo es la culminación de años de apoyo de ambos, por ello es mi deseo que se sientan partícipes de este éxito en mí vida, ya que, definitivamente, el triunfo es de los tres.

A la persona que más amor me ha demostrado en todo momento de mi vida sin reproches y con una humildad admirable, mi madre, como una recompensa al tiempo que me ha dedicado.

A la persona que más admiro, mi padre, que siempre impulsó mi desarrollo profesional, respetando y apoyando mis decisiones. Debido a que mi realización profesional era un sueño para él, aquí está, mi libro, mi dedicación y mi respeto.

# Índice

|   |           |
|---|-----------|
| Agradecimientos . . . . .   | i         |
| Dedicatoria . . . . .   | ii        |
| Índice . . . . .  | iii       |
| Introducción . . . . .  | 1         |
| <b>Capítulo 1. Marco Teórico . . . . .</b>  | <b>3</b>  |
| <b>1.1 GPS . . . . .</b>  | <b>3</b>  |
| 1.1.1 Historia . . . . .  | 3         |
| 1.1.2 Funcionamiento . . . . .  | 4         |
| <b>1.2 Datum WGS-84 . . . . .</b>   | <b>7</b>  |
| 1.2.1 Elipsoide . . . . .   | 7         |
| 1.2.2 Geoide . . . . .  | 8         |
| 1.2.3 Datum . . . . .   | 9         |
| 1.2.4 WGS-84 . . . . .  | 11        |
| <b>1.3 NMEA . . . . .</b>   | <b>13</b> |
| 1.3.1 Interfaz eléctrica . . . . .  | 13        |
| 1.3.2 Formato general de sentencias . . . . .                                     | 14        |
| <b>1.4 Bluetooth . . . . .</b>  | <b>15</b> |
| 1.4.1 Historia . . . . .  | 16        |
| 1.4.2 Especificaciones . . . . .  | 16        |
| 1.4.3 Topología de red . . . . .  | 17        |
| 1.4.4 Enlace de banda base . . . . .  | 17        |
| 1.4.5 Controlador de Interfaz del Host (Host Controller Interface, HCI) . . . . . | 18        |
| 1.4.6 Protocolos basados en software . . . . .                                    | 18        |
| 1.4.7 Perfiles de Bluetooth . . . . .   | 19        |
| 1.4.8 Características técnicas . . . . .  | 20        |
| 1.4.8.1 Espectro de Frecuencias . . . . .   | 20        |
| 1.4.8.2 Robustez . . . . .  | 20        |
| 1.4.8.3 Rango de Operación . . . . .  | 20        |
| 1.4.8.4 Consumo . . . . .   | 20        |
| 1.4.8.5 Velocidades de transmisión . . . . .                                      | 20        |
| 1.4.8.6 Otras características . . . . .   | 21        |

|  |    |
|--|----|
| Capítulo 2. Análisis de la Aplicación . . . . .                | 23 |
| 2.1 Ingeniería de Software . . . . .                           | 23 |
| 2.2 Paradigmas de programación . . . . .                       | 24 |
| 2.2.1 Metodologías de los paradigmas de programación . . . . . | 24 |
| 2.2.2 Modelos de ciclo de vida . . . . .                       | 25 |
| 2.2.2.1 Ciclo de vida lineal . . . . .                         | 25 |
| 2.2.2.2 Ciclo de vida en cascada puro . . . . .                | 26 |
| 2.2.2.3 Ciclo de vida en V . . . . .                           | 28 |
| 2.2.2.4 Ciclo de vida por prototipos . . . . .                 | 28 |
| 2.2.2.5 Ciclo de vida en espiral . . . . .                     | 29 |
| 2.2.2.6 Ciclo de vida orientado a objetos . . . . .            | 31 |
| 2.3 Ciclo de vida del software . . . . .                       | 32 |
| 2.3.1 Análisis y especificación de requisitos . . . . .        | 32 |
| 2.3.2 Diseño del sistema . . . . .                             | 32 |
| 2.3.3 Diseño detallado . . . . .                               | 33 |
| 2.3.4 Programación . . . . .                                   | 33 |
| 2.3.5 Validación y verificación . . . . .                      | 33 |
| 2.4 Comunicación entre el software y el GPS . . . . .          | 34 |
| 2.4.1 Conexión por puerto Serial . . . . .                     | 35 |
| 2.4.2 Conexión por puerto USB o por Bluetooth . . . . .        | 37 |
| 2.4.3 Conexión con PDA . . . . .                               | 39 |
| <br>   |    |
| Capítulo 3. Arquitectura de la Aplicación . . . . .            | 41 |
| 3.1 Arquitectura Conceptual . . . . .                          | 43 |
| 3.1.1 Casos de Uso . . . . .                                   | 43 |
| 3.1.2 Diagramas de Casos de Uso . . . . .                      | 44 |
| 3.2 Arquitectura Lógica . . . . .                              | 47 |
| 3.2.1 Niveles de Servicios . . . . .                           | 48 |
| 3.2.1.1 Servicios de Usuario . . . . .                         | 48 |
| 3.2.1.2 Servicios de Negocio . . . . .                         | 48 |
| 3.2.1.3 Servicios de Datos . . . . .                           | 49 |
| 3.2.2 Diagramas de Actividad . . . . .                         | 49 |
| 3.3 Arquitectura Física . . . . .                              | 52 |
| 3.3.1 Bocetos de la Aplicación . . . . .                       | 53 |
| <br>   |    |
| Capítulo 4. Implantación . . . . .                             | 55 |
| 4.1 Conexión entre el GPS y dispositivos . . . . .             | 55 |
| 4.1.1 Google Earth . . . . .                                   | 55 |
| 4.1.1.1 Ventajas . . . . .                                     | 55 |
| 4.1.1.2 Desventajas . . . . .                                  | 56 |
| 4.1.2 OziExplorer . . . . .                                    | 56 |
| 4.1.2.1 Ventajas . . . . .                                     | 56 |
| 4.1.2.2 Desventajas . . . . .                                  | 57 |

|   |   |           |
|---|---|-----------|
| 4.1.3   | OpenCPN . . . . .   | 57        |
| 4.1.3.1   | Ventajas . . . . .  | 57        |
| 4.1.3.2   | Desventajas . . . . .   | 58        |
| 4.1.4   | Turbo GPS . . . . .   | 58        |
| 4.1.4.1   | Ventajas . . . . .  | 58        |
| 4.1.4.2   | Desventajas . . . . .   | 59        |
| 4.1.5   | GPSToday . . . . .  | 59        |
| 4.1.5.1   | Ventajas . . . . .  | 59        |
| 4.1.5.2   | Desventajas . . . . .   | 60        |
| 4.1.6   | Aplicaciones para Celular . . . . .                                 | 60        |
| 4.1.6.1   | Nokia Maps . . . . .  | 60        |
| 4.1.6.2   | Nokia Sport Tracker . . . . .                                       | 60        |
| 4.1.6.3   | GPS Track . . . . .   | 60        |
| 4.2   | Ejecución de la aplicación . . . . .                                | 62        |
| 4.2.1   | Ejecución de OziExplorer . . . . .                                  | 62        |
| 4.2.2   | Ejecución de Open CPN . . . . .                                     | 65        |
| 4.2.3   | Ejecución de Turbo GPS . . . . .                                    | 69        |
| 4.2.4   | Ejecución de Nokia Maps . . . . .                                   | 72        |
| <b>Capítulo 5. Pruebas y resultados . . . . .</b> |   | <b>75</b> |
| 5.1   | Interacción entre la aplicación y el dispositivo GPS . . . . .      | 75        |
| 5.1.1   | Análisis de tramas NMEA . . . . .                                   | 76        |
| 5.1.2   | Relación mapa-coordenadas . . . . .                                 | 78        |
| 5.2   | Posición Geográfica desplegada en la pantalla del celular . . . . . | 80        |
| 5.2.1   | Botón “Mostrar Ubicación” . . . . .                                 | 81        |
| 5.2.2   | Botón “Ayuda” . . . . .   | 83        |
| 5.2.3   | Botón “Versión” . . . . .   | 83        |
| 5.2.4   | Botón “Salir” . . . . .   | 84        |
| 5.2.5   | Resultados . . . . .  | 85        |
| <b>Conclusiones . . . . .</b>                     |   | <b>87</b> |
| <b>Bibliografía . . . . .</b>                     |   | <b>91</b> |
| <b>Anexos . . . . .</b>                           |   | <b>95</b> |

# Introducción

El objetivo del presente trabajo es analizar la viabilidad de realizar una aplicación para teléfono celular que permita ubicar la posición geográfica en un mapa desplegado en la pantalla del mismo, con la ayuda de un localizador GPS que es el que proporcionará las coordenadas correspondientes a la posición de los dispositivos. Se hace mención a la posición de ambos dispositivos ya que la manera que se propone para comunicar al teléfono celular y al dispositivo GPS es mediante el protocolo de comunicación Bluetooth y este protocolo permite la conexión entre dispositivos en distancias cortas, por lo que se asume que los dispositivos están cerca el uno del otro y la posición es prácticamente la misma para ambos.

Para poder entender las vicisitudes del proyecto es importante comprender algunos términos que serán de importancia en el presente trabajo.

El término GPS tiene dos variantes. Por una parte, se refiere a un Sistema Global de Posicionamiento Satelital. Por otro lado, hace referencia a un dispositivo que se encarga de recibir las señales satelitales del sistema de posicionamiento para determinar las coordenadas de su posición.

También es importante conocer el protocolo en el que se basa el dispositivo GPS, es decir, el lenguaje que utiliza el dispositivo para manejar las coordenadas obtenidas, ya que al momento de sincronizar el teléfono celular con el dispositivo GPS éste último enviará las coordenadas obtenidas a través de dicho protocolo. El protocolo que utilizan los dispositivos GPS es el NMEA 0138, por esto es importante definirlo y mencionar sus principales características.

Debido a que la sincronización se llevará a cabo a través del protocolo Bluetooth, es importante definir este protocolo así como conocer algunas de sus características con el fin de entender y justificar su uso.

De igual manera, es importante analizar la viabilidad de la aplicación haciendo un comparativo con otras aplicaciones existentes en el mercado que tienen características similares a las propuestas para este proyecto, analizar que tan viables son para un teléfono celular, ventajas y desventajas, la manera en que obtienen la información (mapas cargados previamente, conexiones a la red, etc.), en fin analizar si la aplicación es vanguardista o sólo redundante.

Por último, se plantea proponer un diseño de la aplicación para desarrollarla casi sin problemas. Se intentará plantear la manera de afrontar cada percance al momento de poner en práctica la tarea de programar la aplicación.



# Capítulo 1. Marco Teórico

## 1.1 GPS

GPS es el acrónimo de Global Positioning System (Sistema Global de Posicionamiento). Es un sistema formado por una constelación de 24 satélites, llamados NAVSTAR, y 5 estaciones repartidas por la superficie terrestre.

Estos satélites se encuentran en órbitas situadas a 10.900 millas náuticas (20.200 km, aproximadamente) y realizan una circunvalación a la Tierra cada 12 horas. De los 24 satélites en funcionamiento, 21 se encuentran en servicio, mientras que los otros 3 están de reserva. Cada uno de estos satélites emite de manera continua una señal indicando su posición y la hora de su reloj atómico.

Las estaciones terrestres se encuentran repartidas en cinco puntos del globo: Hawái, Isla de Ascensión, Diego García, Atolón de Kwajalein y Colorado Springs. Estas estaciones se encargan de vigilar el estado operativo de los satélites y su correcta posición en el espacio. Una de las estaciones cumple las funciones de estación principal y transmite las correcciones a los satélites.

Gracias a este sistema, un usuario puede determinar con muy poco margen de error su posición en la esfera terrestre y la altitud sobre el nivel del mar en las que se encuentra. Para poder disfrutar de este sistema, el usuario únicamente debe disponer de una terminal receptora.

Las terminales receptoras, conocidas también como unidades o dispositivos GPS, indican la posición en la que estamos.

### 1.1.1 Historia

A comienzos de los años 60, la armada y la fuerza aérea norteamericanas decidieron crear un sistema de localización para su armamento, especialmente el nuclear. Este sistema debía ser muy preciso, estar disponible de manera continua, no verse afectado por las condiciones atmosféricas, funcionar en cualquier lugar del globo y de bajo costo.

Tras realizar inversiones multimillonarias, investigar diversos proyectos previos y diseñar los satélites que integrarían el sistema, en 1989 se lanzaron los primeros satélites que formaban el sistema. El lanzamiento de los satélites originales prosiguió hasta 1994, cuando se lanzó el 24º satélite del sistema.

Como sistema diseñado para la guerra, no fue hasta la guerra del Golfo Pérsico, en 1991 cuando el sistema se sometió a situación de combate. El GPS cumplió su papel a la perfección.

Afortunadamente, el uso del GPS no es exclusivo del ejército norteamericano. Tras un incidente internacional ocasionado en 1983, el entonces presidente de los EE.UU., Ronald Reagan, anunció que el GPS también estaría disponible para la comunidad civil internacional, aunque el sistema tendría una precisión inferior a la que gozaba el ejército norteamericano. En 2000, Bill Clinton eliminó esta restricción y actualmente se logran precisiones de hasta 15 metros en usos civiles. A

pesar de ello, y dado que el sistema está bajo el control, entre otros, del Departamento de Defensa norteamericano, los receptores no pueden ser capaces de funcionar a más de 18.000 metros de altitud ni a más de 900 nudos (1.667 km/hora) de velocidad. Además, el servicio puede verse sometido a restricciones temporales.

### 1.1.2 Funcionamiento

Como se mencionó anteriormente, el sistema está formado por 24 satélites y cinco estaciones terrestres, además del receptor del usuario. Estos satélites, a partir de la información incluida en ellos y la que reciben de las estaciones, generan una señal que transmiten a los receptores. Una vez que los receptores reciben esta señal, calculan la posición.

La base para determinar la posición de un receptor GPS es la trilateración a partir de la referencia proporcionada por los satélites en el espacio. Para llevar a cabo el proceso de trilateración, el receptor GPS calcula la distancia hasta el satélite midiendo el tiempo que tarda la señal en llegar hasta él. Para ello, el GPS necesita un sistema muy preciso para medir el tiempo. Además, es preciso conocer la posición exacta del satélite. Finalmente, la señal recibida debe corregirse para eliminar los retardos ocasionados.

Una vez que el receptor GPS recibe la posición de al menos cuatro satélites y conoce su distancia hasta cada uno de ellos, puede determinar su posición superponiendo las esferas imaginarias que generan.

Podemos comprender mejor esta explicación con un ejemplo. Imaginemos que nos encontramos a 21.000 km de un primer satélite. Esta distancia nos indica que podemos encontrarnos en cualquier punto de la superficie de una esfera imaginaria de 21.000 km de radio. Ahora, imaginemos que nos encontramos a 24.000 km de un segundo satélite. De este modo, también nos encontramos en cualquier punto de la superficie de esta segunda esfera imaginaria de 24.000 km de radio. La intersección de estas dos esferas generará un círculo que disminuirá las posibilidades de situar nuestra posición. Por otro lado, imaginemos que un tercer satélite se encuentra a 26.000 km. Ahora nuestras posibilidades de posición se reducen a dos puntos, aquellos donde se unen la tercera esfera y el círculo generado por las otras dos. Aunque uno de estos dos puntos seguramente dará un valor absurdo (lejos de la Tierra, por ejemplo) y puede ser rechazado sin más, necesitamos un cuarto satélite que determine cuál de ellos es el correcto, aunque no es necesario por la razón anteriormente mencionada. A pesar de su aparente falta de utilidad, este cuarto satélite tendrá una función crucial en la medición de nuestra posición, como se verá más adelante.

Pero, ¿cómo mide el receptor GPS el tiempo que tardan las señales en llegar hasta él? Todos sabemos que la distancia resulta de multiplicar la velocidad por el tiempo (100 km/hora x 3 horas = 300 km). Dado que en el GPS estamos midiendo señales de radio, la velocidad que emplearemos en nuestros cálculos será la de la luz, es decir, 300.000 km/s. Ahora el problema se reduce a conocer la duración del viaje que realiza esta señal. Este cálculo plantea algunos problemas ya que, entre otros, su duración es muy pequeña (en algunos casos puede llegar a ser de 0.067 segundos). Pero asumiendo que disponemos de relojes muy precisos, ¿cómo medimos este tiempo?

Para entender cómo un receptor GPS mide este tiempo, veamos el siguiente ejemplo. Imaginemos que a mediodía pudiéramos sincronizar simultáneamente el receptor y el satélite. Una vez sincronizados, acordamos que a partir de un instante determinado receptor y satélite empiezan a realizar una cuenta (1, 2, 3, etc.). Cuando la señal procedente del satélite llega al receptor, esta llega con un cierto desfase como consecuencia de la distancia. Al receptor sólo le basta medir este desfase (podría ocurrir que la señal con la cuenta 100 llegue al receptor cuando éste va por la cuenta 170, lo cual representaría un desfase de 70). Una vez calculado este desfase, sólo tiene que multiplicar el tiempo desfasado por la velocidad de la luz (en nuestro ejemplo, y suponiendo que las cuentas se realizan en milisegundos,  $300.000 \text{ km/s} \times 0,07 \text{ s} = 21.000 \text{ km}$ ). Para realizar esta sincronización y esta cuenta, los emisores y los receptores del GPS utilizan un método denominado "Pseudo-Random Code" (código pseudoaleatorio) o PRC.

El código PRC es un elemento fundamental del GPS. Se trata de una señal digital (señal eléctrica que representa los valores "0" y "1") muy complicada que casi parece aleatoria, de ahí su nombre. Este código se transmite empleando una señal transportadora a una frecuencia de 1,575.42 MHz, e incluye un mensaje de estado (posición del satélite, correcciones horarias y otros estados del sistema). Los emisores también emplean una segunda frecuencia a 1,227.60 MHz, pero ésta únicamente tiene un uso militar, dada la precisión que permite su uso.

El código PRC es complejo para evitar errores accidentales, su falsificación por parte de un elemento hostil, la superposición de las señales de los distintos satélites y por su bajo costo (de dinero y de espacio). Gracias a la complejidad de esta señal, no es necesario emitir señales muy potentes ni transportar una antena parabólica para recibir la señal del satélite y distinguirla entre el ruido ambiental, como tradicionalmente ocurre con la televisión por satélite. Para distinguirla basta con compararla con el patrón almacenado en el receptor.

Ya hemos comentado que la precisión y la exactitud en la medida de la distancia a los satélites son cruciales para el perfecto funcionamiento del GPS. Para ello, debemos disponer de relojes enormemente precisos, ya que una milésima de segundo a la velocidad de la luz puede suponer un error de 300 km. Por ello los satélites disponen de un reloj atómico en su interior. El nombre de reloj atómico proviene de hecho de utilizar las oscilaciones de un átomo, determinado como "metrónomo".

Lamentablemente, dado el costo y el tamaño, es imposible disponer de un reloj atómico en un receptor. Para solucionar este problema, los ingenieros que desarrollaron el GPS incluyeron (simularon) un "reloj atómico" mediante la recepción de la señal de un satélite extra. La recepción de una señal extra permite que el receptor pueda calcular los errores producidos en la medición y comparación del tiempo y compensarlos, de ahí la necesidad de emplear cuatro satélites para la medición de nuestra posición, en lugar de tres como sería de esperar en un sistema tridimensional. Gracias a este "reloj atómico", los receptores pueden emplearse para algo más que el cálculo de posiciones, como la calibración de otros sistemas de navegación, la sincronización de sistemas informáticos u otros equipos, o la sincronización con el horario universal, entre otros.

Hemos visto que podemos calcular nuestra posición a partir de la posición conocida de cuatro o más satélites, pero, ¿cómo podemos conocer la posición de un satélite que se encuentra a más de 20.000 km de distancia y que da una vuelta a la tierra cada 12 horas?

## Marco teórico

Dado que en el espacio no hay atmósfera, se puede introducir satélites en órbitas invariables que seguirán modelos matemáticos previamente calculados. De este modo, siempre se podrá conocer la posición de cada uno de los satélites en un momento dado. Para ello, los receptores GPS disponen de unos almanaques programados que indican en qué lugar del espacio se encuentran los satélites en cada momento.

A pesar de que estas órbitas son suficientemente exactas, las estaciones de tierra comprueban constantemente sus posiciones. Para ello emplean radares muy precisos que permiten medir la posición y velocidad exactas, y calculan los posibles errores. Estos errores se denominan "errores de efemérides" ya que afectan a la órbita de los satélites o efemérides.

Estos errores se producen como consecuencia del efecto de las atracciones gravitacionales de la Luna y el Sol o por la presión de la radiación solar en los satélites. A pesar de todo, estos errores son mínimos, pero si queremos precisión, debemos tenerlos en cuenta.

Una vez detectados los errores, se retransmiten a los satélites para que éstos puedan incluir la nueva información en las señales emitidas. De este modo, la señal que incluye el PRC es algo más que una señal de sincronizado, es también una señal que contiene información sobre las efemérides.

A pesar de todas las correcciones realizadas hasta el momento, aún quedan una serie de errores por corregir. Hasta ahora se ha considerado que las señales viajan en el vacío y sin ningún obstáculo. Sin embargo, la Tierra está rodeada por la atmósfera, que afecta considerablemente a la recepción de las señales. Para reducir este error existen dos maneras.

La primera de ellas consiste en aplicar un modelo matemático que se actualice a partir de la información recibida de los satélites y que simule el comportamiento de la atmósfera. La otra forma consiste en la medición dual de frecuencias, un sistema que únicamente emplean los receptores militares y que utiliza las dos señales emitidas por los satélites.

Una vez que la señal llega a la superficie de la Tierra, ésta puede reflejarse en diversos obstáculos. De este modo, el receptor puede recibir una señal directa del satélite y, con un ligero desfase, la misma procedente de un reflejo. A este error se le denomina error de trayectoria múltiple. Para eliminarlo, los receptores únicamente tienen en cuenta la señal que llegó en primer lugar, la procedente directamente del satélite.

También pueden producirse errores en los satélites como consecuencia de desfases en los relojes o desvíos en las trayectorias. Además, estos errores pueden magnificarse por un principio denominado dilución geométrica de la precisión o GDOP (Geometric Dilution of Precision). Para reducir este error, los buenos receptores determinan qué satélites son los que proporcionan el menor GDOP, que son aquellos que se encuentran más distanciados.

## 1.2 Datum WGS-84

Mientras que una proyección se utiliza para trazar un mapa que defina la Tierra en una superficie plana, un Datum se utiliza para describir la forma real del planeta en términos matemáticos. Esto se debe a que la superficie terrestre no es una esfera perfecta, sino un elipsoide. Un Datum define también la asociación de coordenadas de latitud y longitud con puntos sobre la superficie de la Tierra y define la base de medidas de elevación.

La forma del planeta Tierra es bastante irregular: Ligeramente achatado en los polos y abultado en el Ecuador, con el hemisferio sur un poco más voluminoso que el norte, y con la rugosidad propia que da el relieve del terreno.

Además de la forma irregular de la Tierra, hay que agregar la forma correspondiente a la orografía de la superficie terrestre que es muy compleja, incluyendo los océanos. Por ello es que se utilizan aproximaciones en lugar de la forma general de la Tierra para muchas aplicaciones.

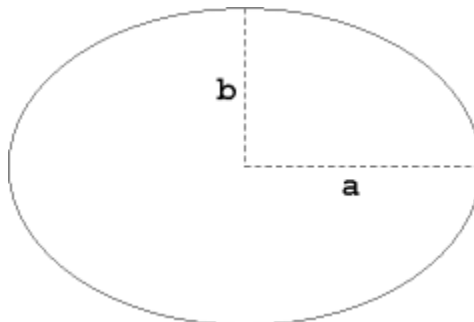
Para crear un mapa que represente las diferentes áreas del globo terráqueo, es necesario aplicar diferentes modelos matemáticos que sean capaces, en primer lugar, de representar de la mejor forma posible la forma de la Tierra.

### 1.2.1 Elipsoide

En general, es más práctico trabajar la forma de la Tierra como si fuera un elipsoide, sin considerar las ondulaciones propias de la topografía. Esto se debe a que el elipsoide es una figura matemática fácil de usar que es lo suficientemente parecida a la forma de la Tierra cuando se están trabajando las coordenadas en el plano: Latitud y Longitud.

Existen diferentes modelos de elipsoides utilizados en geodesia, denominados elipsoides de referencia. Las diferencias entre éstos vienen dadas por los valores asignados a sus parámetros más importantes:

- Semieje ecuatorial ( $a$ ) o Semieje mayor: Longitud del semieje correspondiente al ecuador, desde el centro de masas de la Tierra hasta la superficie terrestre.
- Semieje polar ( $b$ ) o Semieje menor: Longitud del semieje desde el centro de masas de la Tierra hasta uno de los polos. Alrededor de este eje se realiza la rotación de la elipse base.



**Figura 1.1.** Elipsoide de Referencia.

Matemáticamente, una elipse se define mediante la ecuación:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

Así mismo, el factor de achatamiento ( $f$ ) representa qué tan diferentes son los semiejes de una elipse entre sí. El factor de achatamiento es un parámetro que utilizan los sistemas de referencia como el WGS-84.

$$f = 1 - \frac{b}{a}$$

### 1.2.2 Geoide

A pesar de ser una figura matemática sencilla, el elipsoide no es adecuado cuando lo que deseamos medir son altitudes. Dado que la mayor parte de la Tierra está cubierta por mares y océanos, la superficie de referencia por excelencia para medir altitudes es el nivel medio del mar. Además, el nivel medio es una mejor aproximación a la forma real de la Tierra vista desde el espacio.

El nivel medio del mar depende de las irregularidades en el campo gravitatorio de la Tierra, que alteran su posición. El agua de los océanos busca estar en equilibrio y, por ello, tienden a seguir una superficie gravitatoria equipotencial.

Es por esto que se introduce una nueva figura llamada Geoide, que se define como la superficie equipotencial del campo gravitatorio de la Tierra que mejor se ajusta (en el sentido de mínimos cuadrados), al nivel medio global del mar. Una de las consecuencias de esta definición es que el geoide es siempre perpendicular al vector de gravedad local en cada punto.

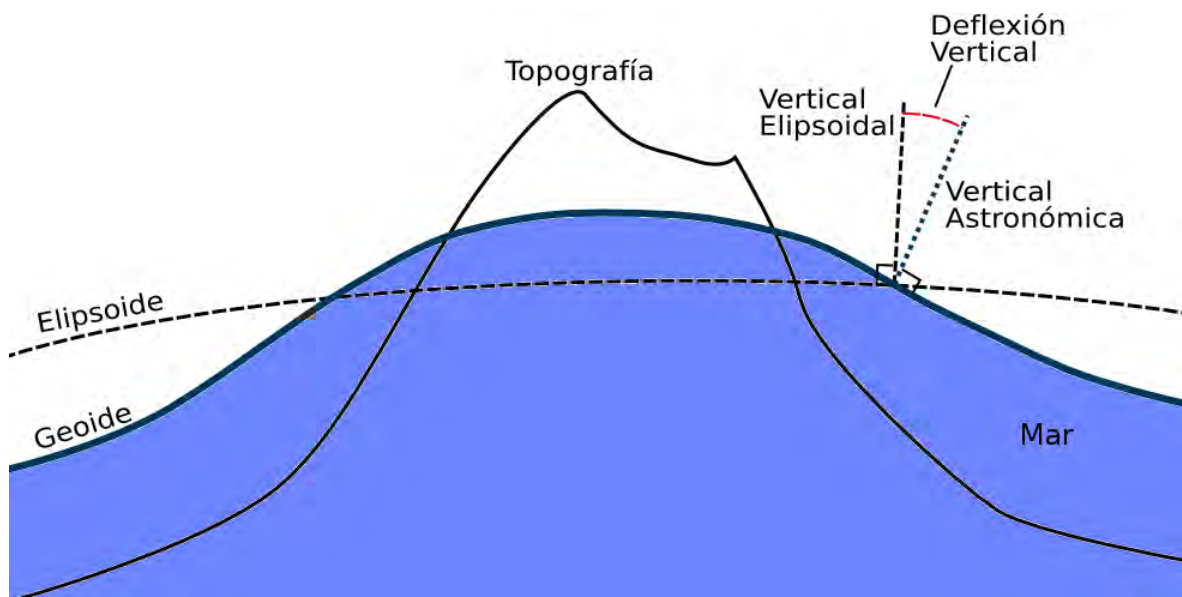
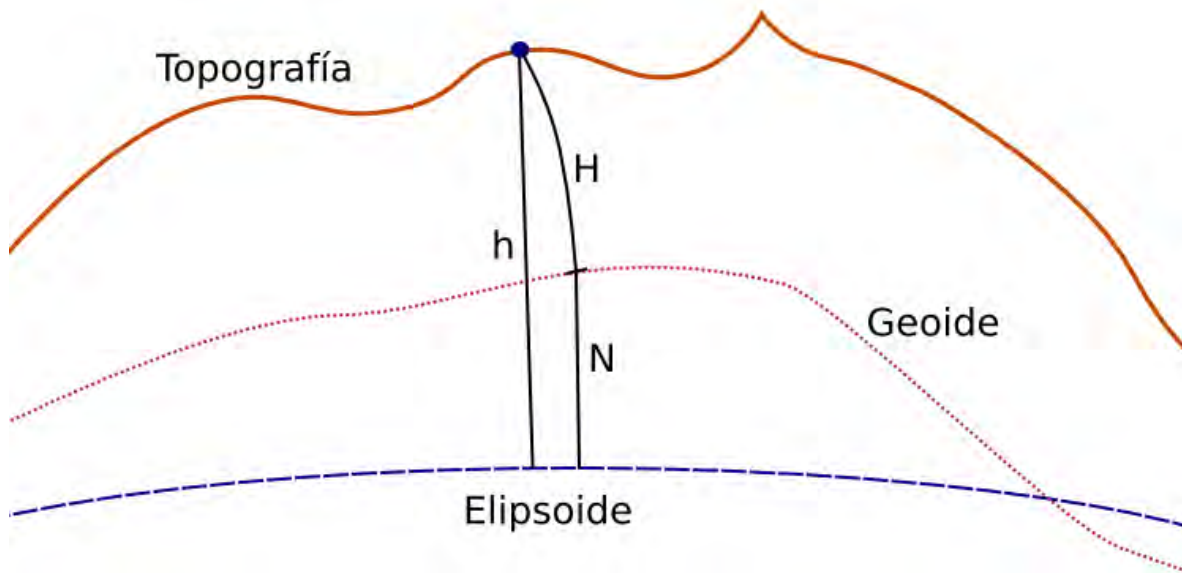


Figura 1.2. Comparación entre el Elipsoide y el Geoide.

Medir el geode a nivel mundial es una tarea difícil, pues la corteza terrestre no es homogénea y por tanto su densidad no es uniforme, lo que altera la fuerza de gravedad en un punto dado. Asimismo, grandes masas de rocas (como cordilleras, montañas y volcanes) pueden alterar también el vector de gravedad local.

Es posible relacionar matemáticamente al geode y el elipsoide mediante la expresión  $h = H + N$ . Donde  $h$  es la altura de un punto con respecto al elipsoide (altura elipsoidal),  $N$  es la altura del geode respecto al elipsoide (ondulación del geode) y  $H$  es la altura del punto con respecto al geode (llamada altura ortométrica).



**Figura 1.3.** Relación entre el Geode y el Elipsoide.

### 1.2.3 Datum

Datum se define como el punto tangente al elipsoide y al geode donde ambos son coincidentes. Cada Datum está compuesto por un elipsoide y un punto llamado "Fundamental" en el que el elipsoide y la Tierra son tangentes. Este punto se define por sus coordenadas geográficas (latitud y longitud) y su acimut (ángulo que se toma desde el punto cardinal norte en sentido horario y va de 0 a 360 grados).

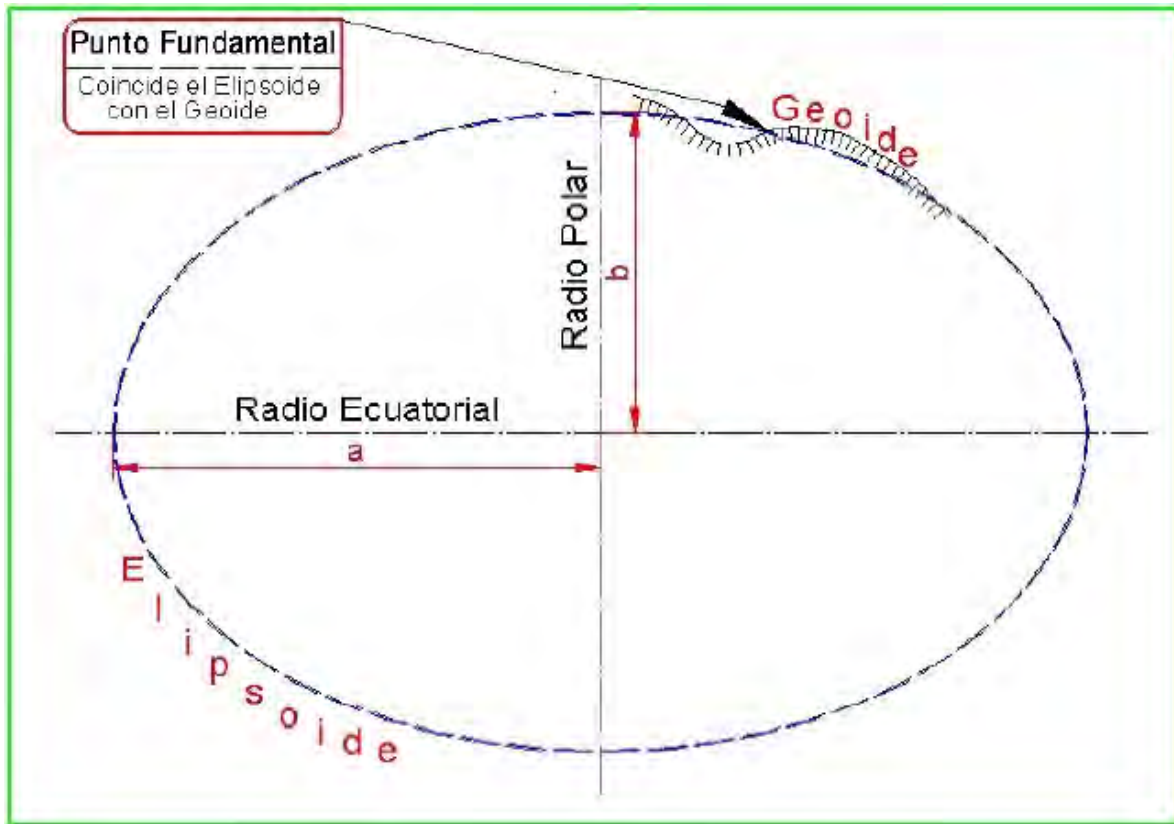


Figura 1.4. Ubicación del Punto Fundamental.

En el Punto Fundamental coincide el Elipsoide con la superficie real de la Tierra, es decir, las coordenadas astronómicas (las del Elipsoide) y las coordenadas geodésicas (las de la Tierra). Estas dos desviaciones vienen dadas al no coincidir la vertical perpendicular al Geoide, trazada por el Punto Fundamental, con la vertical perpendicular al Elipsoide. De esta manera el sistema queda definido al estar marcados estos ángulos en el Datum.

Con estos datos se puede elaborar la cartografía de cada lugar, pues se tienen parámetros de referencia que relacionan el punto de origen del geoide y del elipsoide con su posición geográfica, así como la dirección del sistema.

La desviación en la vertical (ETA) se genera porque no coinciden la vertical en el geoide con la vertical en el elipsoide y no pasa la perpendicular del elipsoide por el centro de la elipse de revolución, originando un punto imaginario "S".



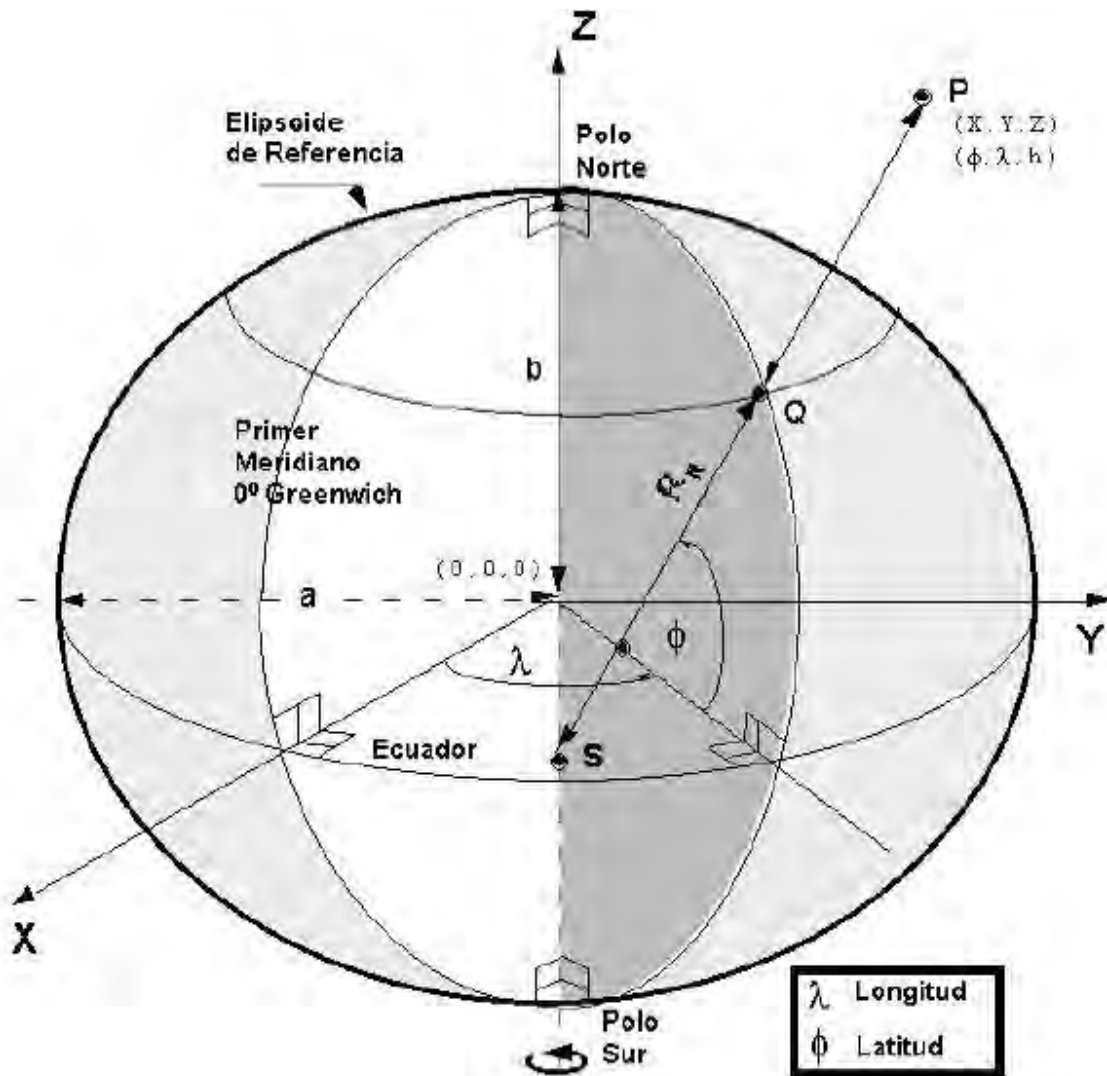


Figura 1.5. Medición de un punto.

#### 1.2.4 WGS-84

Con el uso de nuevas técnicas de posicionamiento, como el Sistema de Posicionamiento Global (GPS), es necesario disponer de un sistema que ayude a situar una posición geográfica con referencia a un Datum Universal con cobertura en toda la superficie terrestre.

Para ello fue creado el Sistema Geodésico Mundial WGS (World Geodetic System), denominado WGS-74. El sistema de referencia que actualmente se utiliza es el WGS84.

El WGS-84 es un sistema de referencia terrestre convencional (Conventional Terrestrial Reference System, CTRS). En su definición se siguen las recomendaciones del Servicio Internacional de Rotación Terrestre (International Earth Rotation Service, IERS). Se trata de un sistema de referencia geocéntrico fijo con la Tierra y orientado positivamente.

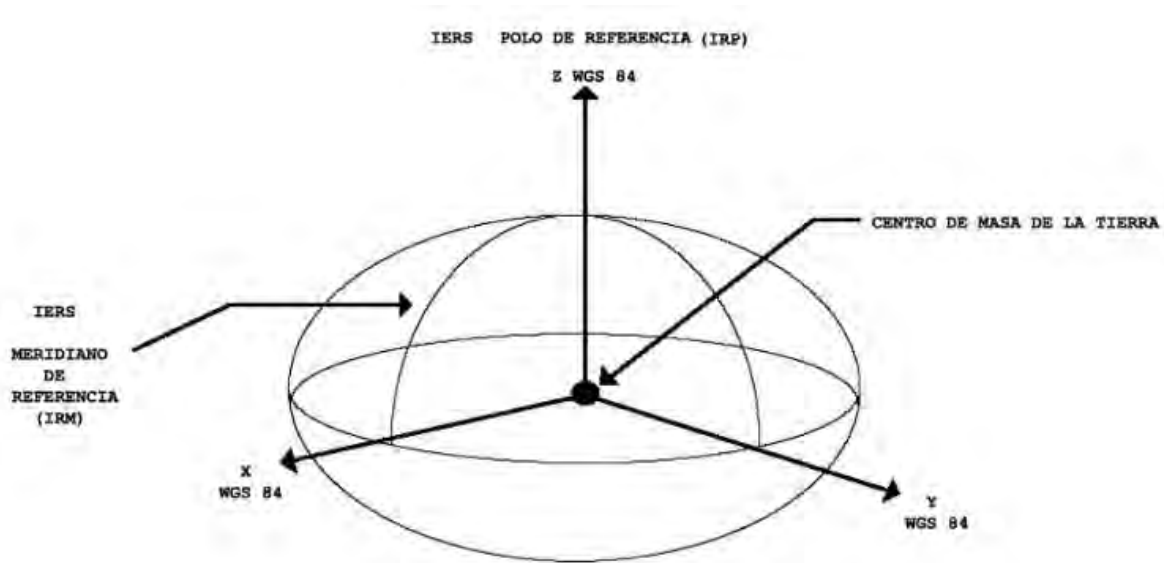


Figura 1.6. Sistemas de referencia de WGS-84.

Donde el origen O es el centro de masas de la Tierra. El eje Z es la dirección del Polo de Referencia IERS que coincide con el Polo Terrestre Convencional (CTP) de la Oficina Internacional de la Hora (Bureau International de l'Heuer). El eje x es la intersección del meridiano de referencia IERS y el plano perpendicular al eje Z por el origen de coordenadas. El eje Y completa un sistema cartesiano ortogonal orientado positivamente.

Asociado al sistema cartesiano se considera un sistema de coordenadas geodésicas definidas por un elipsoide de revolución cuyo centro y eje de revolución coinciden respectivamente con el origen de coordenadas y el eje Z.

Los parámetros que utiliza el WGS-84 se describen en la tabla 1.1.

| <i>Nombre</i>                         | <i>Símbolo</i>      | <i>Valor</i>                   |
|---------------------------------------|---------------------|--------------------------------|
| <i>Semieje mayor de la elipse</i>     | a                   | 6378,137000 km                 |
| <i>Semieje menor de la elipse</i>     | b                   | 6356,752314 km                 |
| <i>Factor de achatamiento</i>         | $f = \frac{a-b}{a}$ | 1/298,257223563                |
| <i>Velocidad angular de la Tierra</i> | $\omega_e$          | $7292115 \cdot 10^{-11}$ rad/s |

Tabla 1.1. Parámetros del WGS-84.

## 1.3 NMEA

NMEA (National Marine Electronics Association) es una asociación sin fines de lucro de fabricantes, distribuidores, instituciones educacionales y otros interesados en equipos periféricos marinos. El estándar NME 0183 define una interfaz eléctrica y un protocolo de datos para comunicaciones entre instrumentos marinos.

El NMEA 0183 es un estándar industrial voluntario, fue lanzado por primera vez en marzo de 1983. Ha sido actualizado algunas veces, su última actualización, versión 3.0, fue lanzada en julio de 2001.

NMEA también ha establecido un grupo de trabajo para desarrollar un nuevo estándar para la comunicación de datos entre dispositivos electrónicos en las embarcaciones. El nuevo estándar, NMEA 200, es una red de datos en serie bidireccional, multi-transmisora y multi-receptora.

### 1.3.1 Interfaz Eléctrica

Los dispositivos NMEA 0183 son designados tanto para interlocutores como para oyentes (algunos para ambos) empleando una interface serial asíncrona con los siguientes parámetros:

|                          |                   |
|--------------------------|-------------------|
| Tasa de Baudios:         | 4800              |
| Número de datos en bits: | 8 (el bit 7 es 0) |
| Bits para detener:       | 1 (o más)         |
| Paridad:                 | Ninguna           |
| Protocolo de Control:    | Ninguno           |

El NMEA 0183 permite a un interlocutor y varios oyentes en un circuito. La interconexión alámbrica recomendada es un escudo de par trenzado, con el escudo a tierra sólo para el interlocutor. El estándar no especifica el uso de un conector en particular.

Es recomendable que la salida del interlocutor cumpla con EIA RS-422, un sistema diferencial con dos señales, A y B. El manejador diferencial de señales no tiene referencia a tierra y es más inmune al ruido. Sin embargo, una línea terminal sencilla de nivel TTL es aceptada también. Los voltajes en la línea A corresponden al cable TTL sencillo mientras que los voltajes en B son invertidos (cuando la salida A sea +5 V, la salida B será 0 V y viceversa). Esta es la operación unipolar RS-422. En el modo bipolar se usan  $\pm 5$  V.

En cualquier caso se recomienda que el circuito receptor use un opto-aislador con un adecuado régimen de protección de circuitos eléctricos. La entrada debe ser aislada de la tierra del receptor. En la práctica, el cable sencillo, o el cable RS-422 A debe ser conectado directamente a una computadora con entrada RS-422. De hecho, muchos de los últimos productos, como los receptores GPS de mano, no tienen una salida RS-422 diferencial, sólo una sencilla línea con TTL o niveles de señales compatibles con 5V CMOS.

### 1.3.2 Formato general de sentencias

Todos los datos son transmitidos en forma de sentencias. Sólo los caracteres ASCII imprimibles son permitidos, además de CR (retorno de carro) y LF (avance de línea). Cada sentencia empieza con la señal "\$" y termina con "<CR><LF>". Hay tres tipos básicos de sentencias: sentencias del interlocutor, sentencias propietarias y sentencias de consulta.

Sentencias del interlocutor: El formato general para una sentencia de interlocutor es:

```
$tsss,d1,d2,...<CR><LF>
```

Las primeras dos letras que siguen a "\$" son los identificadores del interlocutor. Los siguientes 3 caracteres (sss) son los identificadores de sentencia, seguidos por un número de campos de datos separados por comas, seguidos por una comprobación opcional y terminando por el retorno de carro y el avance de línea. Los campos de datos son únicos definidos por cada tipo de sentencia. Un ejemplo de una sentencia de interlocutor es:

```
$HCHDM, 238, M<CR><LF>
```

donde "HC" especifica al interlocutor como una brújula magnética, el "HDM" especifica el mensaje del grado magnético. El "238" es el valor del grado y "M" designa el valor del grado como magnético.

Una sentencia puede contener hasta 80 caracteres además de "\$" y CR/LF. Si los datos para un campo no están disponibles, el campo se omite, pero las comas delimitadoras se siguen mandando sin espacios entre ellas. El campo de comprobación consiste en un "\*" y en dos dígitos hexadecimales que representan el OR exclusivo de todos los caracteres que están entre el símbolo "\$" y el símbolo "\*" (sin incluir estos símbolos).

Sentencias propietarias: El estándar permite a los fabricantes definir los formatos de este tipo de sentencias. Estas sentencias empiezan con "\$P" y luego un identificador de 3 letras del fabricante, seguido por los datos que el fabricante desee y por el formato general de las sentencias propietarias.

Sentencias de consulta: Una sentencia de consulta es un medio para que el oyente pida una sentencia particular al interlocutor. El formato general es:

```
$tllIQ,sss, [CR][LF]
```

Los primeros dos caracteres del campo de dirección son los identificadores del interlocutor del solicitante y los siguientes dos caracteres son los identificadores del interlocutor del dispositivo requerido (oyente). El quinto carácter es siempre una "Q" que define el mensaje como una consulta. El siguiente campo (sss) contiene el mnemónico de 3 letras de la sentencia requerida. Un ejemplo de sentencias de consulta es el siguiente:

```
$CCGPQ,GGA<CR><LF>
```

donde el dispositivo “CC” (Computadora) está solicitando del dispositivo “GP” (una unidad GPS) la sentencia “GGA”. El GPS transmitirá esta sentencia una vez cada segundo hasta que una consulta diferente sea solicitada.

Los dispositivos que generalmente utilizan el protocolo NMEA son:

- GPS
- Compás magnético
- Radar o Radar ARPA
- Ecosonda, profundidad
- Sensores de velocidad, magnéticos, dopler o mecánicos
- Instrumentos meteorológicos
- Transductores
- Reloj atómico, cuarzo, cronómetro
- Sistemas de navegación integrados
- Comunicaciones satelitales o de radio

## 1.4 Bluetooth

Bluetooth es una tecnología utilizada para conectividad inalámbrica de corto alcance entre dispositivos tales como PDAs (Personal Digital Assistance), teléfonos celulares, teclados, máquinas de fax, computadoras de escritorio y portátiles, modems, proyectores, impresoras, etc. El principal mercado es la transferencia de datos y voz entre dispositivos y computadoras personales. El enfoque de Bluetooth es similar a la tecnología de infrarrojo conocida como IrDA (Infrared Data Association). Sin embargo, Bluetooth, es una tecnología de radiofrecuencia (RF) que utiliza la banda de espectro disperso de 2.4 GHz. Muchas veces también se le confunde con el estándar IEEE 802.11, otra tecnología de RF de corto alcance. IEEE 802.11 ofrece más caudal eficaz pero necesita más potencia de transmisión y ofrece menos opciones de conectividad que Bluetooth para el caso de aplicaciones de voz. Las principales características del Bluetooth son robustez, bajo consumo de energía y bajo costo. Las especificaciones Bluetooth definen una estructura uniforme para que un amplio tipo de dispositivos puedan conectarse y comunicarse entre sí.

Bluetooth es el nombre común de la especificación industrial IEEE 802.15.1, que define un estándar global de comunicación inalámbrica que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia segura, globalmente y sin licencia de corto rango. El Bluetooth es una especificación o estándar para redes inalámbricas personales de área. Esta tecnología permite intercambiar información entre dispositivos, usando una frecuencia de radio de onda corta.

La tecnología Bluetooth ha conseguido tal aceptación a nivel mundial que cualquier dispositivo que disponga de esta tecnología puede comunicarse con otro dispositivo Bluetooth que se encuentre en su radio de proximidad. Estas comunicaciones inalámbricas entre dispositivos son pequeñas redes de corto alcance de tipo ad hoc, conocidas como piconets. Cada dispositivo puede establecer varias piconets simultáneamente. Las conexiones son establecidas dinámicamente y automáticamente cuando un dispositivo Bluetooth entra en un radio de proximidad de otro dispositivo Bluetooth.

Pero la característica más importante de la tecnología Bluetooth es la capacidad de realizar transmisiones de voz y datos de forma simultánea. Esto permite disfrutar de soluciones muy útiles, como son: los manos libres, los auriculares inalámbricos, impresoras inalámbricas, sincronizar nuestra PDA, etc.

### 1.4.1 Historia

Su nombre, procede del nombre un Vikingo de origen danés *Harald Blåtand*, debido a que su traducción al inglés sería *Harold Bluetooth*, conocido por buen comunicador y por unificar las tribus noruegas, suecas y danesas.

En el año 1994, la compañía Ericsson inició diversas investigaciones con el objetivo expreso de estudiar la viabilidad de la existencia de una nueva interfaz (de bajo consumo y costo), entre diversos aparatos, como teléfonos móviles u otros dispositivos. La versión 1.0 de la especificación Bluetooth fue liberada ese mismo año. El estudio demostró que el uso de enlaces de radio sería el más adecuado, ya que no es directivo y no necesita línea de vista; eran tan obvias estas ventajas con respecto a los enlaces vía infrarrojo que son utilizados para conectar dispositivos y teléfonos celulares.

El Bluetooth SIG (Special Interest Group, <http://www.bluetooth.com/>) es un grupo de compañías trabajando juntas para promover y definir la especificación Bluetooth. Bluetooth SIG fue fundado en Febrero de 1998 por las siguientes compañías: Ericsson, Intel, IBM, Toshiba y Nokia. En Mayo de 1998, se anuncia públicamente el Bluetooth SIG y se invita a otras compañías para que se unan a éste. Fue en julio de 1999 cuando el SIG publica la versión 1.0 de la especificación de Bluetooth. En diciembre de 1999, se unen otras compañías tales como Microsoft, Lucent, 3com y Motorola.

### 1.4.2 Especificaciones

A diferencia de otros estándares inalámbricos, las especificaciones inalámbricas de Bluetooth dan a los desarrolladores de equipos las definiciones de la capa de enlace y de la capa de aplicación, las cuales soportan aplicaciones de voz y datos.

Bluetooth opera en la banda 2.4 [GHz] bajo la tecnología de radio conocida como espectro disperso. La banda de operación está dividida en canales de 1 [MHz], a 1 mega-símbolo por segundo puede obtenerse al ancho de banda máximo por canal. Con el esquema de modulación empleado, GFSK (Gaussian Frequency Shift Keying), esto equivale a 1 [Mbps]. Utilizando GFSK, un 1 binario representa una desviación positiva de la portadora nominal de la frecuencia, mientras que un 0 representa una desviación negativa. Después de cada paquete, ambos dispositivos resintonizan su radio transmisor a una frecuencia diferente, saltando de un canal a otro canal de radio; esta técnica se le conoce como espectro disperso con salto en frecuencia (FHSS, Frequency Hopping Spread Spectrum). De esta manera, los dispositivos Bluetooth utilizan toda la banda de 2.4 GHz y si una transmisión se interfiere sobre un canal, una retransmisión siempre ocurrirá sobre un canal diferente con la esperanza de que este canal esté libre. Cada ranura de tiempo tiene una duración de 625 [µs] y generalmente los dispositivos saltan una vez por paquete, o sea, saltan cada ranura, cada 3 ranuras o cada 5 ranuras.

Aunado a las distancias cortas de conexión, Bluetooth en materia de ancho de banda soporta hasta 780 [Kbps], los cuales pueden ser utilizados para transferir unidireccionalmente 721 [Kbps] y 57.6 [Kbps] en la dirección de retorno o hasta 432.6 [Kbps] de manera simétrica en ambas direcciones. Aunque estas velocidades están limitadas para cierto tipo de aplicaciones como video, otras aplicaciones como la transferencia de archivos o la impresión caen perfectas en tal ancho de banda.

La especificación de Bluetooth cubre desde el transceptor de radio hasta varias interfaces de protocolos basados tanto en hardware como en software. Control de enlace: el hardware del control de enlace controla la transmisión y recepción de radio así como el procesamiento de la señal digital requerida para el protocolo de banda base. Sus funciones incluyen establecer conexiones, soporte para enlaces asíncronos (datos) y síncronos (voz), corrección de error y autenticación. El micro código del administrador de enlaces desempeña funciones a bajo nivel para el establecimiento de enlaces, autenticación y configuración de los enlaces.

#### 1.4.3 Topología de la red

Los dispositivos Bluetooth son generalmente organizados en grupos de 2 a 8 llamados picoceldas o picoredes (piconets), consisten de un dispositivo maestro y uno o más dispositivos esclavos. Un dispositivo puede pertenecer a más de una piconet y comportarse como un esclavo en ambas o un maestro en una piconet y como esclavo en otra.

Como Bluetooth opera en una banda de uso libre conocida como ISM (Industrial, Scientific, and Medical) donde otros dispositivos de uso común la utilizan como es el caso de puertas de cocheras, teléfonos inalámbricos, hornos de microondas, sólo por nombrar algunos. Para que los dispositivos Bluetooth puedan coexistir y operar confiablemente con los otros dispositivos, cada piconet es sincronizada a una frecuencia específica del patrón de salto por frecuencia. Este patrón, que salta a 1,600 frecuencias diferentes por segundo, es único para una piconet en particular. Cada "salto" de frecuencia es una ranura de tiempo durante la cual los paquetes de datos son transferidos. Un paquete puede abarcar hasta 5 ranuras de tiempo, en la cual la frecuencia permanece constante durante la duración de esa transferencia.

Si los dispositivos van a saltar a las nuevas frecuencias después de cada paquete, ellos deben ponerse de acuerdo en la secuencia de las frecuencias que utilizarán. Como los dispositivos Bluetooth operan en 2 modos (como maestro y como esclavo), si el maestro asigna la secuencia de salto de frecuencia, los esclavos sincronizan al dispositivo maestro en tiempo y frecuencia seguido de la secuencia de salto del dispositivo maestro.

#### 1.4.4 Enlace de banda base

La banda base de Bluetooth provee canales de transmisión para voz y datos y es capaz de soportar un enlace asíncrono de datos y hasta tres enlaces de voz asíncronos (o un enlace soportando ambos). Los enlaces orientados a conexión síncronos (SCO) son típicamente empleados para transmisiones de voz. Esos enlaces son conexiones simétricas punto a punto que reservan ranuras de tiempo para garantizar la transmisión a tiempo. Al dispositivo esclavo siempre se le permitirá responder durante la ranura de tiempo inmediatamente después de una transmisión tipo SCO del maestro. Un dispositivo maestro puede soportar hasta tres enlaces SCO a uno o varios esclavos,

pero un esclavo puede soportar sólo enlaces SCO para diferentes dispositivos maestros. Los paquetes SCO nunca son retransmitidos.

Los enlaces orientados a no-conexión (ACL, Asynchronous Connectionless) son típicamente empleados para transmisión de datos. Las transmisiones sobre estos enlaces son establecidas con base en ranuras (en ranuras no reservadas para enlaces SCO). Los enlaces ACL soportan transferencias punto-multipunto de datos asíncronos como síncronos. Después de una transmisión ACL del maestro, sólo el dispositivo esclavo direccionado puede responder durante la siguiente ranura de tiempo o si el dispositivo no está direccionado, los paquetes son considerados como mensajes difundidos (broadcast). La mayoría de los enlaces ACL incluyen retransmisión de paquetes. Administrador de enlaces: La máquina de estado de banda base es controlada por el administrador de enlaces. Este micro código provee el control del enlace basado en hardware para configuración, seguridad y control de enlaces. Sus capacidades incluyen autenticación y servicios de seguridad, monitoreo de calidad de servicio y control del estado de bandabase. El administrador de enlaces se comunica con los demás utilizando el protocolo LMP (Link Management Protocol), el cual utiliza los servicios básicos de banda base. Los paquetes LMP, los cuales son enviados sobre los enlaces ACL, son diferenciados de los paquetes L2CAP (Logical Link Control and Adaptation Protocol) por un bit en el encabezado del ACL. Ellos son siempre enviados como paquetes de una ranura y una prioridad más alta que los paquetes L2CAP. Esto ayuda a asegurar la integridad del enlace ante una alta demanda de tráfico.

### 1.4.5 Controlador de Interfaz del Host (Host Controller Interface, HCI)

Por encima del administrador de enlaces se encuentra el HCI. Este protocolo basado en hardware es usado para aislar la banda base de Bluetooth y el administrador de enlaces de un protocolo de transporte tal como el RS-232 o USB (Universal Serial Bus). Esto permite una interfaz estándar para el hardware de Bluetooth. Un manejador de dispositivos HCI en el host es usado para interactuar una aplicación Bluetooth con el protocolo de transporte. Actualmente existen tres mecanismos de transporte soportados: USB, RS-232 y el UART (Universal Asynchronous Receiver-Transmitter). Utilizando HCI, una aplicación Bluetooth puede acceder al hardware de Bluetooth sin el conocimiento de la capa de transporte u otros detalles de implementación del hardware.

### 1.4.6 Protocolos basados en software

El resto de los protocolos son implementados en software. La capa más baja de L2CAP provee la interfaz con el administrador de enlaces y permite la interoperabilidad entre dispositivos Bluetooth. Provee la multicanalización de protocolos, lo cual permite el soporte de otros protocolos de más alto nivel como TCP/IP. El L2CAP opera sobre un enlace del tipo ACL en banda base y provee enlaces punto-multipunto para transferencias síncronas y asíncronas. L2CAP provee servicios a los protocolos de los niveles superiores al transmitir paquetes de datos sobre los canales L2CAP. Existen tres tipos de canales L2CAP: canales bidireccionales que transportan comandos; canales orientados a conexión para conexiones punto-punto y bidireccionales; y canales unidireccionales orientados a no-conexión que soporten conexiones punto-multipunto, permitiendo que una entidad local L2CAP sea conectada a un grupo de dispositivos remotos.



Varios protocolos interactúan con la capa de enlace L2CAP tales como SDP y RFCOMM. El protocolo SDP (Service Discovery Protocol) provee un medio para determinar qué servicios Bluetooth están disponibles en un dispositivo en particular. Un dispositivo Bluetooth puede actuar como un cliente SDP solicitando servicios o como un servidor SDP proveyendo servicios, o ambos. Un simple dispositivo Bluetooth tendrá no más de un servidor SDP, pero puede actuar como un cliente para más de un dispositivo remoto. El protocolo SDP provee acceso sólo a información acerca de servicios, el uso de estos servicios deberá ser proveído por otro protocolo.

RFCOMM es un protocolo de transporte que provee transferencia de datos serial. Una entidad de emulación de puertos es usada para mapear la comunicación de la interfaz de la programación de aplicaciones (API, Applications Programming Interface) a los servicios de RFCOMM, permitiendo que el software legado opere en un dispositivo Bluetooth. TCS (Telephony Control Protocol Specification), un protocolo para aplicaciones de telefonía es proveído para control de llamadas de voz y datos a través de señalización. La señalización tanto para punto-punto y punto-multipunto son soportados utilizando los canales L2CAP, la voz o los datos son transferidos directamente desde la banda base sobre los enlaces SCO.

Bluetooth también soporta el protocolo de sesión conocido como IrOBEX (IrDA Object Exchange Protocol), definido por IrDA. Este protocolo puede operar sobre las capas de transporte, incluyendo RFCOMM y TCP/IP. Para dispositivos Bluetooth, solo OBEX orientado a conexión es soportado. Tres perfiles de aplicación han sido desarrollados usando OBEX. Estos incluyen funcionalidades de sincronización para directorios telefónicos, calendarios, mensajes, etc., funcionalidades de transferencia de archivos y Object Push para soporte de tarjetas de presentación.

#### 1.4.7 Perfiles de Bluetooth

Los perfiles son una parte muy importante en la tecnología Bluetooth. Los perfiles le proveen a Bluetooth una significativa ventaja sobre las otras tecnologías. Los perfiles, definidos por Bluetooth SIG, tienen la intención de asegurar la interoperabilidad entre las aplicaciones de Bluetooth y los dispositivos de diferentes fabricantes. Estos perfiles definen los roles y capacidades para aplicaciones específicas. Diferentes perfiles pueden abarcar diferentes capas y protocolos y diferentes grados de seguridad. Además de los requerimientos de interoperabilidad, los protocolos pueden definir servicios requeridos para otras aplicaciones o para usuarios finales.

Todos los dispositivos Bluetooth deben soportar el perfil de acceso genérico (Generic Access Profile) como mínimo. Este perfil en particular define el descubrimiento o hallazgo de dispositivos, procedimientos de conexión y procedimientos para varios niveles de seguridad. También se describen algunos requerimientos de interfaz al usuario. Otro perfil universal, aunque no es requerido, es el perfil de acceso a descubrimiento de servicios (Service Discovery Access Profile), el cual define los protocolos y parámetros asociados requeridos para acceder a los perfiles. Un número de perfiles han sido definidos incluyendo TCS, RFCOMM y OBEX. Algunos de estos requieren la implementación de otros, y todos ellos requieren la implementación de perfiles genéricos.

## 1.4.8 Características Técnicas

### 1.4.8.1 Espectro de frecuencia

La tecnología Bluetooth opera en banda libre (es decir, no hay que pagar licencias) entre 2.4 y 2.485 [Ghz] utilizando Frequency Hopping, siendo full duplex y realizando 1600 [hops/sec] (saltos de frecuencia por segundo). La banda de 2.4 [Ghz] está disponible y no requiere licencias en la mayoría de los países.

En los Estados Unidos y Europa, el intervalo de frecuencia de operación es de 2,400 a 2,483.5 MHz, con 79 canales de RF de 1 [MHz]. En la práctica, el intervalo de frecuencias es de 2,402 a 2,480 [MHz]. En México el intervalo de frecuencias va de 2,450 [MHz] a 2,485.5 [MHz]. En Japón, el intervalo de frecuencia es de 2,472 a 2,497 [MHz] con 23 canales de RF de 1 [MHz].

### 1.4.8.2 Robustez

La tecnología Bluetooth Adaptive Frequency Hopping (AFH) fue diseñada para reducir las interferencias entre tecnologías inalámbricas que compartan el mismo espectro de frecuencia en los 2.4 [Ghz]. AFH intenta trabajar con las bandas de frecuencia disponibles, es decir, las que no están en uso en un determinado momento. Esto lo hace detectando los otros dispositivos del espectro y evitando las frecuencias que éstos usan. Estos saltos de frecuencia adaptativos permiten transmisiones más eficientes, permitiendo gran robustez en las comunicaciones incluso si los dispositivos conviven junto a otras tecnologías inalámbricas. La señal salta a través de 79 frecuencias a intervalos de 1 [Mhz] lo que da un alto grado de inmunidad contra el ruido.

### 1.4.8.3 Rango de operación

Bluetooth fue diseñado para aplicaciones móviles de poca potencia, la potencia del radio transmisor debe ser minimizada. Tres diferentes niveles de potencias están definidas. El rango de operación depende de la clase del dispositivo:

- Clase 3.- Distancias aproximadas de 1 [m].
- Clase 2.- Esta es la clase más común de los dispositivos. Tiene un rango de alcance de unos 10 [m].
- Clase 1.- Usado principalmente es entornos industriales. Tiene un rango de alcance de unos 100 [m].

### 1.4.8.4 Consumo

El dispositivo más utilizado, que como comentábamos antes es el de clase 2, consume una potencia de 2.5 [mW], cumpliendo con la característica de que Bluetooth es de bajo consumo de energía.

### 1.4.8.5 Velocidades de transmisión

Para la versión del Core 1.2, las velocidades de transmisión son de hasta 1 [Mbps]. La versión del Core 2.0 soporta velocidades de hasta 3 [Mbps].

#### 1.4.8.6 Otras características

Los dispositivos en una piconet comparten un canal de comunicación de datos común. El canal tiene una capacidad total de 1 [Mbps]. Los encabezados y el control de llamada consumen cerca del 20% de esta capacidad; motivo por el cual el máximo caudal eficaz es de 780 [Kbps].

Una piconet tiene un dispositivo maestro y hasta siete dispositivos esclavos. Un dispositivo maestro transmite en ranuras de tiempo pares, los esclavos en ranuras de tiempo impares. Los paquetes pueden tener una magnitud de hasta 5 ranuras de tiempo. Los datos en un paquete pueden ser de hasta 2,745 bits de longitud.

Existen actualmente dos tipos de transferencia de datos entre dispositivos: Los orientados a conexión de tipo síncrono (SCO, Synchronous Connection Oriented) y los orientados a no-conexión de tipo asíncrono (ACL, Asynchronous Connectionless).

En una piconet, puede hacer hasta tres enlaces SCO de 64,000 bits cada uno. Para evitar problemas de sincronización y colisión, los enlaces SCO utilizan ranuras de tiempo reservadas asignadas por la estación maestra.

Un dispositivo maestro puede soportar hasta tres enlaces SCO con uno, dos o tres dispositivos esclavos. Las ranuras no reservadas para los enlaces SCO pueden ser usadas para enlaces ACL.

Un maestro y un esclavo pueden compartir un enlace ACL. Un enlace ACL puede ser punto a punto (maestro a un esclavo) o multipunto (maestro a todos los esclavos). Un ACL esclavo sólo puede transmitir cuando se lo solicite un maestro.

Como ya se ha mencionado, Bluetooth permite manipular simultáneamente transmisiones de voz y datos. Es capaz de soportar un canal de datos asíncrono y hasta tres canales de voz asíncronos o un canal que soporte ambos, voz y datos. La capacidad combinada con los dispositivos del tipo "ad hoc" permite soluciones superiores para dispositivos móviles y aplicaciones de Internet. Esta combinación permite soluciones innovadoras como un dispositivo de manos libres para llamadas de voz, impresión a máquinas de fax y sincronización automática a PDAs, laptops y aplicaciones de libreta de direcciones de teléfonos celulares.

# Capítulo 2. Análisis de la Aplicación

## 2.1 Ingeniería de Software

La ingeniería de software ha sido definida como: “El uso y establecimiento de principios de ingeniería sólidos, a fin de obtener un software que sea económicamente fiable y funcione eficientemente en máquinas reales”. Sin embargo, definiciones más comprensivas han estado proponiendo reforzar todos los requerimientos para la disciplina ingenieril en el desarrollo de software.

La ingeniería de software es el resultado del hardware y la ingeniería de sistemas. Esto abarca un conjunto de tres elementos claves (métodos, herramientas y procedimientos) que le permiten al administrador controlar el proceso de desarrollo de software y proporcionar al practicante fundamentos para construir software de alta calidad de una manera productiva.

Los métodos de la ingeniería de software proveen la técnica de los “cómos” para crear software. Los métodos abarcan una amplia gama de tareas que incluyen: planeación y estimación del proyecto; análisis de requerimientos del sistema y software; diseño de la estructura de datos, la arquitectura del programa y el procedimiento algorítmico, codificación, pruebas, y mantenimiento. Los métodos para la ingeniería de software a menudo presentan un lenguaje-orientado especial o una notación gráfica e introducen un conjunto de criterios para la calidad de software.

Las herramientas de ingeniería de software proveen apoyo automatizado o semi-automatizado para los métodos. Hoy en día, las herramientas existen para apoyar cada uno de los métodos mencionados anteriormente. Cuando las herramientas son integradas entonces la información creada por una herramienta puede ser usada por otra, un sistema para el apoyo del desarrollo de software, llamado Ingeniería de Software Asistida por Computadora (Computer-Aided Software Engineering, CASE) es establecido. CASE combina software, hardware y una base de datos de ingeniería de software (una estructura de datos que contiene información importante sobre análisis, diseño, código y pruebas) para crear un entorno de ingeniería de software que sea análogo a CAD/CAE [Diseño/ingeniería asistida(o) por computadora] para hardware.

Los procedimientos de ingeniería de software se encargan de unir los métodos y las herramientas y permiten, racional y oportunamente, el desarrollo de software de computadora. Los procedimientos definen la secuencia en la cual los métodos serán aplicados, los entregables (documentos, reportes, formularios, etc.) que sean requeridos, los controles que ayudan a asegurar la calidad y a coordinar los cambios, y los hitos que permiten a los administradores de software evaluar el progreso.

La ingeniería de software comprende un conjunto de pasos que abarcan métodos, herramientas y procedimientos discutidos anteriormente. Los caminos en los cuales estos pasos son aplicables son a menudo referidos como paradigmas de ingeniería de software. Un paradigma de ingeniería de software es elegido con base en la naturaleza del proyecto y aplicación, los métodos y herramientas a ser usados y las demandas de los clientes que han solicitado el software.

## Análisis de la aplicación

Para explorar la esencia de la ingeniería de software es importante ver algunas de las técnicas que permiten a los ingenieros controlar la complejidad de un producto.

La primera técnica se llama modelado: la construcción de un modelo que constituye una simplificación del mundo real, pero que al mismo tiempo es lo suficientemente realista para usarse en el desarrollo. El modelado es parte medular de cualquier disciplina de ingeniería.

La segunda técnica es la que se conoce como división del producto. Una técnica común en cualquier proyecto complejo es dividir el producto del proyecto en fragmentos rastreables. Estos trozos serán lo suficientemente pequeños y no muy complejos como para que pueda realizarlos un miembro del personal del proyecto. La división se lleva a cabo en toda la ingeniería. El objetivo principal de la división es reducir la complejidad.

La tercera técnica es similar a la división de un producto y se conoce como división del proceso. Implica dividir el desarrollo de un artefacto en tareas. En un producto convencional, estas tareas son la especificación, el diseño y la fabricación. La especificación es el proceso de describir las propiedades de un artefacto; el diseño es la definición de la arquitectura del producto y la fabricación es la construcción del artefacto a partir de piezas manufacturadas. La división de procesos reduce la complejidad al permitir que el productor se centre en una tarea a la vez.

## 2.2 Paradigmas de programación

La crisis del software alude a un conjunto de problemas que fueron encontrados en el desarrollo de software de computadora. El problema no se limitó al funcionamiento apropiado del software. Más bien, la crisis del software abarca problemas asociados con cómo se desarrolla el software, cómo se mantiene un volumen creciente de software existente y cómo se puede mantener el ritmo con la demanda de más software.

No hay un enfoque sencillo para solucionar la crisis del software. Sin embargo, combinando métodos exhaustivos para todas las fases del desarrollo de software, mejores herramientas para automatizar estos métodos, bloques de construcción más potentes para la implementación de software, mejores técnicas para garantizar la calidad del software y una filosofía general de coordinación, control y administración, podemos lograr una disciplina para el desarrollo de software. Esta disciplina es lo que conocemos como ingeniería de software.

Los paradigmas de programación son las estrategias para crear la estructura de un programa. Existen dos grupos: la programación imperativa y la declarativa. En la primera codificamos qué hacer y cómo, en la segunda sólo qué hacer, el lenguaje que utiliza hará el resto.

### 2.2.1 Metodologías de los paradigmas de programación

Existen dos metodologías que tienen analogía en la práctica con los paradigmas de programación, la metodología estructurada y la metodología orientada a objetos.

- Metodología estructurada: la orientación de esta metodología se dirige hacia los procesos que intervienen en el sistema a desarrollar, es decir, cada función a realizar por el sistema se descompone en pequeños módulos individuales. Es más fácil resolver problemas pequeños, y luego unir cada una de las soluciones, que abordar un problema grande.

- Metodología orientada a objetos: a diferencia de la metodología mencionada anteriormente, ésta no comprende los procesos como funciones sino que arma módulos basados en componentes, es decir, cada componente es independiente del otro. Esto nos permite que el código sea reutilizable. Es más fácil de mantener porque los cambios están localizados en cada uno de estos componentes.



**Figura 2.1.** Las distintas metodologías nos ayudarán a abordar el proceso de desarrollo.

### 2.2.2 Modelos de ciclo de vida

Las principales diferencias entre distintos modelos de ciclo de vida están divididas en tres grandes visiones:

El alcance del ciclo de vida: depende de hasta dónde se desea llegar con el proyecto para saber si es viable el desarrollo de un producto, el desarrollo completo o el desarrollo completo más las actualizaciones y el mantenimiento.

La cualidad y cantidad de las etapas en que se divida el ciclo de vida: según el ciclo de vida que se adopte y el proyecto para el cual se enfoque.

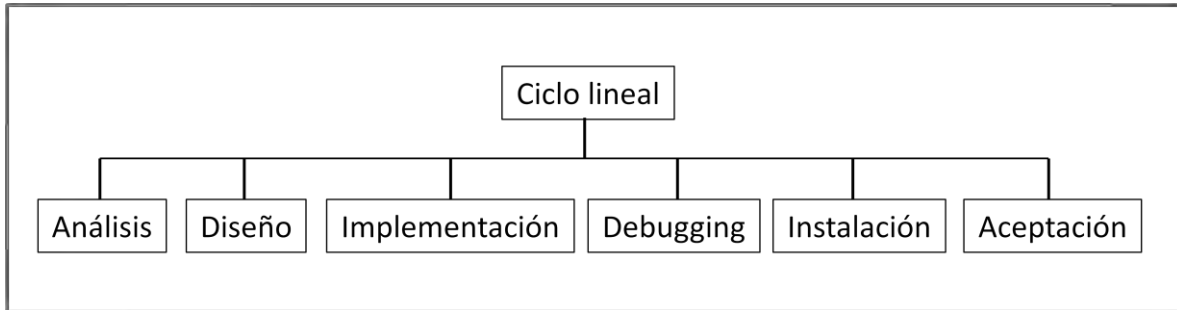
La estructura y la sucesión de las etapas: si hay realimentación entre ellas y si tenemos libertad de repetirlas (iterar).

#### 2.2.2.1 Ciclo de vida lineal

Es el más sencillo de todos los modelos. Consiste en descomponer la actividad global del proyecto en etapas separadas que son realizadas de manera lineal, es decir, cada etapa se realiza una sola vez, a continuación de la etapa anterior y antes de la etapa siguiente. Con un ciclo de vida lineal es muy fácil dividir las tareas, y prever los tiempos (sumando linealmente los de cada etapa).

## Análisis de la aplicación

Las actividades de cada una de las etapas mencionadas deben ser independientes entre sí, es decir, que es condición primordial que no haya retroalimentación entre ellas, aunque sí pueden admitirse ciertos supuestos de realimentación correctiva. Desde el punto de vista de la gestión, requiere también que se conozca desde el primer momento, con excesiva rigidez, lo que va a ocurrir en cada una de las distintas etapas antes de comenzarla. Esto último minimiza, también, las posibilidades de errores durante la codificación y reduce al mínimo la necesidad de requerir información del cliente o del usuario.



**Figura 2.2.** El ciclo de vida lineal es el más sencillo.

Se destaca como ventaja la sencillez de su gestión y administración tanto económica como temporal, ya que se acomoda perfectamente a proyectos internos de una empresa para programas muy pequeños de ABM (sistemas que realizan Altas, Bajas y Modificaciones sobre un conjunto de datos). Tiene como desventaja que no es apto para desarrollos que superen mínimamente requerimientos de retroalimentación entre etapas, es decir, es muy costoso retomar una etapa anterior al detectar alguna falla.

Es válido tomar este ciclo de vida cuando algún sector pequeño de una empresa necesita llevar un registro de datos acumulativos, sin necesidad de realizar procesos sobre ellos más que una consulta simple. Es decir, una aplicación que se dedique exclusivamente a almacenar datos, sea una base de datos o un archivo plano. Debido a que el desarrollo de las etapas es muy simple y el código muy sencillo.

### 2.2.2.2 Ciclo de vida en cascada puro

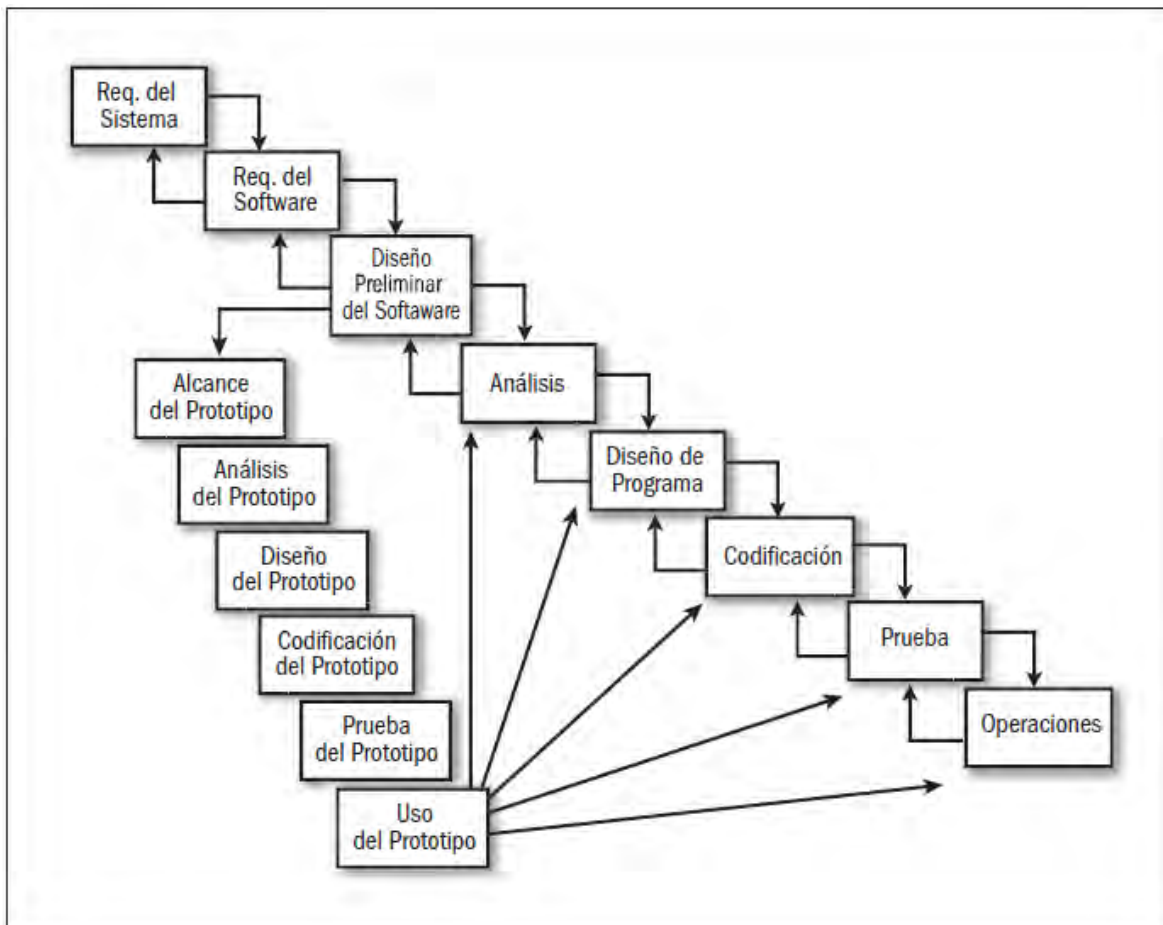
Este modelo de ciclo de vida fue propuesto por Winston Royce en el año 1970. Es un ciclo de vida que admite iteraciones, contrariamente a la creencia de que es un ciclo de vida secuencial como el lineal. Después de cada etapa se realiza una o varias revisiones para comprobar si se puede pasar a la siguiente. Es un modelo rígido, poco flexible, y con muchas restricciones. Aunque fue uno de los primeros, y sirvió de base para el resto de los modelos de ciclo de vida.

Una de sus ventajas, además de su planificación sencilla, es la de proveer un producto con un elevado grado de calidad sin necesidad de un personal altamente calificado. Se pueden considerar como inconvenientes: la necesidad de contar con todos los requerimientos (o la mayoría) al comienzo del proyecto, y, si se han cometido errores y no se detectan en la etapa inmediata siguiente, es costoso y difícil volver atrás para realizar la corrección posterior.

Además, los resultados no los veremos hasta que no estemos en las etapas finales del ciclo, por lo que, cualquier error detectado nos trae retraso y aumenta el costo del desarrollo en función del tiempo que insume la corrección de éstos.

Es un ciclo adecuado para los proyectos en los que se dispone de todos los requerimientos al comienzo, para el desarrollo de un producto con funcionalidades conocidas o para proyectos que, aún siendo muy complejos, se entienden perfectamente desde el principio.

Se evidencia que es un modelo puramente teórico, ya que el usuario rara vez mantiene los requerimientos iniciales y existen muchas posibilidades de que debamos retomar alguna etapa anterior. Pero es mejor que no seguir ningún ciclo de vida.



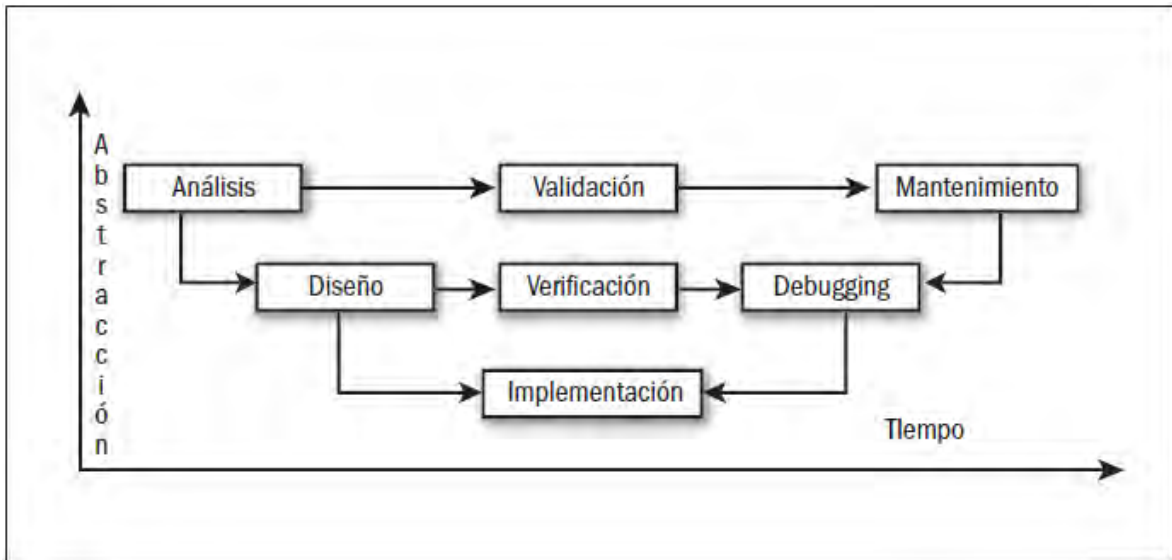
**Figura 2.3.** Este modelo de ciclo de vida permite iteraciones.



## Análisis de la aplicación

### 2.2.2.3 Ciclo de vida en V

Este ciclo fue diseñado por Alan Davis y contiene las mismas etapas que el ciclo de vida en cascada puro. A diferencia de aquél, a éste se le agregaron dos subetapas de retroalimentación entre las etapas de análisis y mantenimiento, y entre las de diseño y debugging.



**Figura 2.4.** Este modelo ofrece mayor garantía de corrección al terminar el proyecto.

Las ventajas y desventajas de este modelo son las mismas del ciclo anterior, con el agregado de los controles cruzados entre etapas para lograr una mayor corrección.

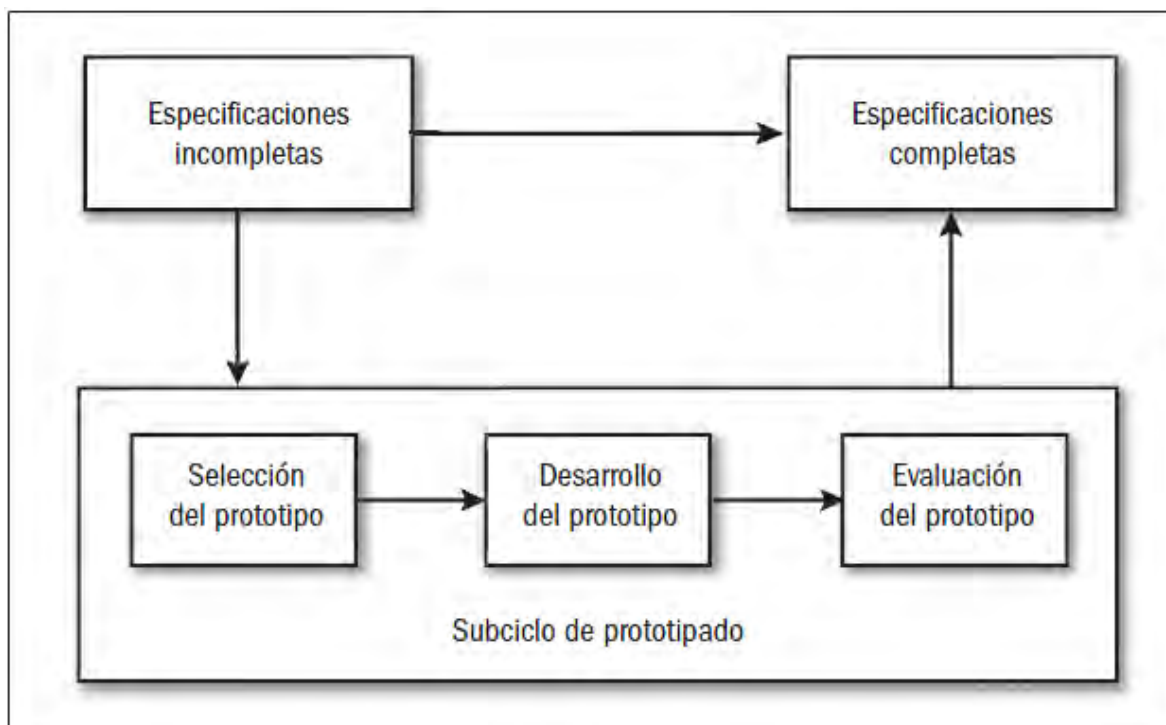
Este modelo de ciclo de vida se puede utilizar en aplicaciones, que si bien son simples (pequeñas transacciones sobre bases de datos, por ejemplo), necesitan una confiabilidad muy alta.

Por ejemplo en una aplicación de facturación no se puede permitir el lujo de cometer errores, si bien los procedimientos vistos individualmente son de codificación e interpretación sencilla, la aplicación en su conjunto puede tener matices complicados.

### 2.2.2.4 Ciclo de vida por prototipos

El uso de programas prototipo no es exclusivo del ciclo de vida iterativo. En la práctica, los prototipos se utilizan para validar los requerimientos de los usuarios en cualquier ciclo de vida.

Si no se conoce exactamente cómo desarrollar un determinado producto o cuáles son las especificaciones de forma precisa, suele recurrirse a definir especificaciones iniciales para hacer un prototipo, o sea, un producto parcial y provisional. En este modelo, el objetivo es lograr un producto intermedio, antes de realizar el producto final, para conocer mediante el prototipo cómo responderán las funcionalidades previstas para el producto final.



**Figura 2.5.** Este modelo permite suavizar la transición entre requerimientos.

Antes de adoptar este modelo de ciclo debemos evaluar si el esfuerzo por crear un prototipo vale realmente la pena.

Se utiliza, la mayoría de las veces, en desarrollos de productos con innovaciones importantes o en el uso de nuevas tecnologías o poco probadas en las que la incertidumbre sobre los resultados a obtener o la ignorancia sobre el comportamiento impiden iniciar un proyecto secuencial.

La ventaja de este ciclo es que se puede usar sobre desarrollos en los que no se conoce, a priori, sus especificaciones o la tecnología a utilizar. Sin embargo, tiene la desventaja de ser altamente costoso y difícil para la administración temporal.

#### 2.2.2.5 Ciclo de vida en espiral

Este ciclo puede considerarse una variación del modelo por prototipos. Fue diseñado por Boehm en el año 1988. El modelo se basa en una serie de ciclos repetitivos para ir ganando madurez en el producto final. Toma los beneficios de los ciclos de vida incremental y por prototipos pero se tiene más en cuenta el concepto de riesgo que aparece debido a las incertidumbres e ignorancia de los requerimientos proporcionados al principio del proyecto o que surgirán durante el desarrollo. A medida que el ciclo se cumple (el avance del espiral) se van obteniendo prototipos sucesivos que van ganando la satisfacción del cliente o usuario. A menudo la fuente de incertidumbre es el propio cliente o usuario que, en la mayoría de las oportunidades, no sabe con perfección todas las funcionalidades que debe tener el producto.

## Análisis de la aplicación

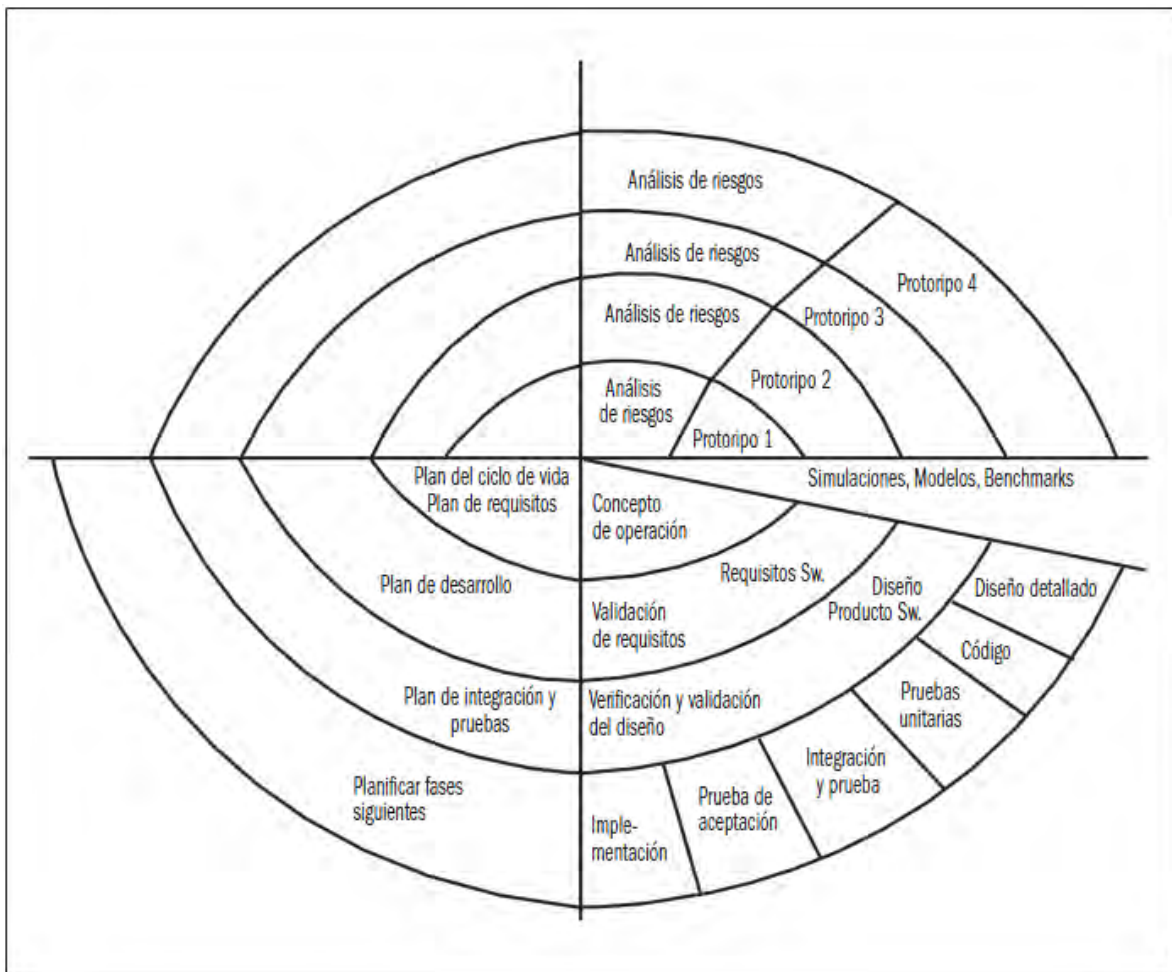
En este modelo hay cuatro actividades que envuelven las etapas.

- Planificación: Levantamiento de requerimientos iniciales o luego de una iteración.
- Análisis de riesgo: De acuerdo con el levantamiento de requerimientos decidir si se continúa con el desarrollo.
- Implementación: Se desarrolla un prototipo basado en los requerimientos.
- Evaluación: El cliente evalúa el prototipo, si da su conformidad.

La ventaja más notoria de este modelo de desarrollo de software es que puede comenzarse el proyecto con un alto grado de incertidumbre. Otra ventaja es el bajo riesgo de retraso en caso de detección de errores, ya que se puede solucionar en la próxima rama del espiral.

Algunas de las desventajas son: el costo temporal que suma cada vuelta en el ciclo de vida, la dificultad para evaluar los riesgos y la necesidad de la presencia o la comunicación continua con el cliente o usuario.

Se observa que es un modelo adecuado para grandes proyectos internos de una empresa en donde no es posible contar con todos los requerimientos desde el comienzo y el usuario está en nuestro mismo ambiente laboral.



**Figura 2.6.** El espiral se repite hasta satisfacer las necesidades del usuario.

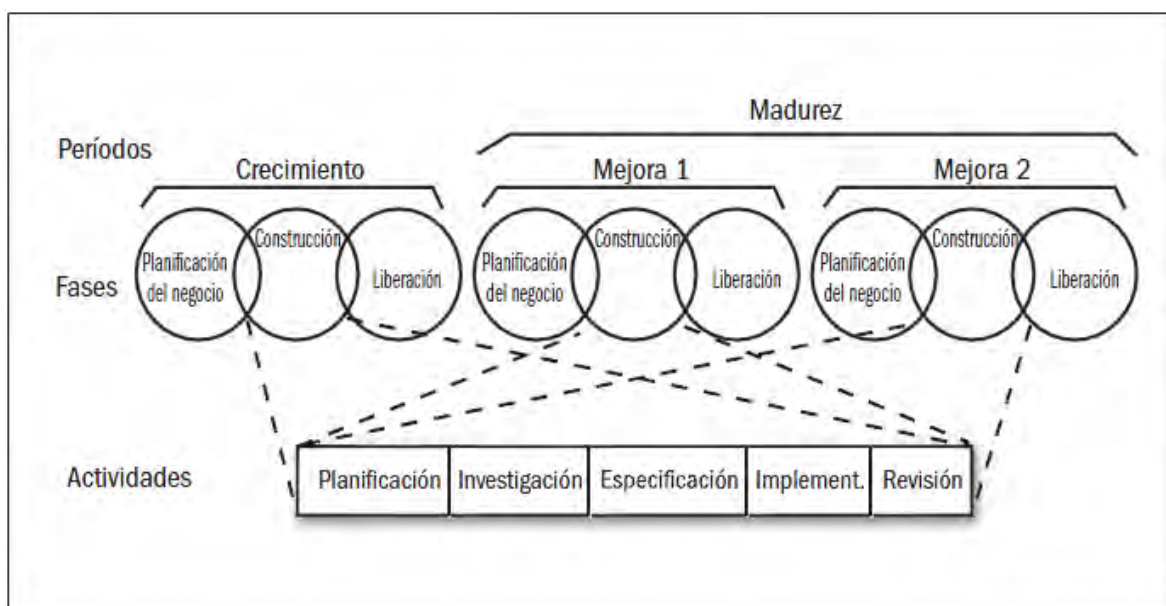
### 2.2.2.6 Ciclo de vida orientado a objetos

Esta técnica fue presentada en la década de los 90's, tal vez como una de las mejores metodologías a seguir para la creación de productos de software.

Puede considerarse como un modelo pleno a seguir, así como una alternativa dentro de los modelos anteriores.

Al igual que la filosofía del paradigma de la programación orientada a objetos, en esta metodología cada funcionalidad o requerimiento solicitado por el usuario es considerado un objeto. Los objetos están representados por un conjunto de propiedades, a los cuales se denominan atributos; por otra parte, el comportamiento que tendrán estos objetos los se denominan métodos.

La característica principal de este modelo es la abstracción de los requerimientos del usuario, por lo que este modelo es mucho más flexible que los restantes, que son rígidos en requerimientos y definición, soportando mejor la incertidumbre que los anteriores, aunque sin garantizar la ausencia de riesgos. La abstracción permite analizar y desarrollar las características esenciales de un objeto (requerimiento) y evita preocuparse de las menos relevantes. Esto se puede observar en la figura 2.7.



**Figura 2.7.** Un modelo muy versátil, tanto para pequeños como grandes proyectos

Favorece la reducción de la complejidad del problema que deseamos abordar y permite el perfeccionamiento del producto.

En este modelo se utilizan las llamadas fichas CRC (Clase-Responsabilidades-Colaboración) como herramienta para obtener las abstracciones y mecanismos clave de un sistema analizando los requerimientos del usuario. En la ficha CRC se escribe el nombre de la clase u objeto, sus responsabilidades (los métodos) y sus colaboradores (otras clases u objetos de las cuales necesita). Además estas fichas ayudan a confeccionar los denominados casos de uso.

## Análisis de la aplicación

No es correcto suponer que este modelo sólo es útil cuando se escoge para la implementación de un lenguaje con orientación a objetos. Se puede usar independientemente del lenguaje elegido. Es un modelo a seguir, una técnica, y nos obliga a no utilizar un lenguaje en particular.

Es un método muy versátil y, por ser uno de los últimos en aparecer, aprendió mucho de los anteriores.

## 2.3 Ciclo de vida del software

La ISO (International Organization for Standardization) en su norma 12207 define al ciclo de vida de un software como un marco de referencia que contiene las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando desde la definición hasta la finalización de su uso.

En cada una de las etapas del ciclo de vida, se pueden establecer una serie de objetivos, tareas y actividades que lo caracterizan.

### 2.3.1 Análisis y especificación de requisitos

Es el proceso de averiguar qué es lo que requiere un cliente de un sistema de software (análisis de requisitos) y de definir estos requisitos en forma clara y concisa (especificación de requisitos). El productor de software se enfrenta al principio de un proyecto con un documento escrito por el cliente, en el cual expresa, en términos de la aplicación, qué se requiere del sistema. Este documento se conoce como declaración de requisitos.

El objetivo de la especificación de requisitos es producir una especificación de sistema, la cual, en teoría, es un documento en el cual se han eliminado las deficiencias de la declaración de requisitos y donde se presentan los requisitos del cliente en una forma fácil de comprender. La especificación del sistema deben contener una declaración de las funciones del sistema así como las restricciones con las cuales tendrá que trabajar el productor de software. También deben contener una gran cantidad de información adicional, por ejemplo, los detalles del hardware que se usará y la capacitación que tendrá que proporcionar el productor, así como una especificación de las herramientas de software especiales que se usarán en el proyecto.

### 2.3.2 Diseño del sistema

El diseño del sistema se encarga de desarrollar la arquitectura general de un sistema, la cual se expresa como una serie de fragmentos, por lo general subrutinas o procedimientos. El objetivo del diseño del sistema es producir una arquitectura que satisfaga las funciones de la especificación del sistema y al mismo tiempo respete las restricciones del documento.

En el diseño del sistema, el diseñador tiene que especificar el procesamiento que debe ocurrir en un componente individual y definir la interfaz entre los componentes.

El diseño del sistema debe realizar todas las funciones que se detallan en la especificación del sistema y, al mismo tiempo, satisfacer todas las restricciones. Representa una descripción del sistema en términos relacionados con las computadoras (unidades de programa, interfaces y datos), mientras que la especificación del sistema lo describe en términos orientados a la aplicación. Existe otra diferencia importante entre la especificación del sistema y su diseño. La primera establece lo que hará el sistema y el segundo describe cómo lo hará.

### 2.3.3 Diseño detallado

El diseño detallado es el proceso de definición de los algoritmos que conforman cada unidad de programa. Como el diseño del sistema consiste en una serie de fragmentos discretos, entonces se puede asignar un ingeniero de software a cada trozo o unidad de programa. La notación que se usa para describir el procesamiento que ocurre en cada unidad de programa se conoce como lenguaje de diseño de programas. Este lenguaje consiste en recursos junto con descripciones del procesamiento en lenguaje natural.

### 2.3.4 Programación

Una vez que se ha definido con un lenguaje el diseño de programa cuál será el procesamiento de las unidades que integran el sistema, el siguiente paso es la programación. Este proceso es el menos difícil pues sólo comprende la traducción de las unidades de programa, expresadas en lenguaje de diseño de programas, a código de programa. Esto casi siempre implica una traducción directa de los enunciados del lenguaje de diseño del programa a enunciados en lenguaje de programación.

### 2.3.5 Validación y verificación

En términos de ingeniería de software un sistema es correcto cuando refleja por completo los requisitos del usuario que se detallaron en la especificación del sistema. El proceso de revisar que un sistema sea correcto se describe mediante el término colectivo de validación y verificación: dos actividades separadas que aseguran que el sistema que se entrega cumple con los requisitos del usuario. La verificación es el proceso de revisar que la salida de una fase corresponda con la entrada a esa fase. En la fase de diseño del sistema, esto consiste en asegurar que el diseño del sistema sea un reflejo correcto de la especificación de sistema.

La validación es el proceso de asegurar que el sistema en desarrollo cumpla con los requisitos del usuario. Un ejemplo de validación lo representa las pruebas de aceptación: el proceso de ejecución de un sistema con datos de prueba para revisar que cumpla con los requisitos de los usuarios.

La validación y la verificación se llevan a cabo durante todo el ciclo de vida del software. Durante el proceso de análisis de requisitos y la especificación tienen lugar dos actividades de validación y verificación. La primera es una revisión de la especificación de requisitos contra la declaración del cliente. La revisión es una junta del personal del proyecto para revisar los documentos que se les presenta.

## Análisis de la aplicación

La segunda actividad consiste en extraer los requisitos de verificación. Esto tiene lugar al final de esta fase, al concluir la especificación del sistema, y comprende leer la especificación del sistema y extraer los requisitos, los cuales se probarán más adelante al revisar el proyecto para ver si se ajusta a los requisitos del usuario.

Durante el diseño del sistema hay varias actividades de verificación y validación importantes. En primer lugar están las revisiones del diseño, cuya función principal es asegurar que el diseño del sistema realice en forma adecuada los requisitos del cliente que se expresan en la especificación del sistema. Esta revisión del diseño contra la especificación se lleva a cabo examinando cada uno de los requisitos de verificación para luego rastrearlos hasta las unidades de programa que se ejecutarán.

Otra actividad de validación y verificación que se presenta durante el diseño del sistema es la ampliación de los requisitos de verificación para obtener pruebas más detalladas.

La última tarea de validación y verificación durante el diseño del sistema es determinar la estrategia de pruebas de integración. Las pruebas de integración ocurren durante la fase de programación del proyecto de software. En esta fase, las unidades de programa se programarán y probarán individualmente y después se agregarán gradualmente al sistema. El proceso de construir de manera gradual un sistema se conoce como integración, y las pruebas que ocurren en este proceso se denominan pruebas de integración. Esta técnica es preferible al hecho de unir y probar todas las unidades de programa al mismo tiempo.

La razón principal para adoptar este enfoque gradual tiene que ver con los errores. Si las unidades de programa que se han programado se unen al mismo tiempo, será extremadamente difícil determinar dónde ocurren los errores. Sin embargo, en un sistema que se construye unidad por unidad, si la prueba de integración detecta un error, lo más probable es que éste se encuentre en la unidad de programa que se está incorporando.

## 2.4 Comunicación entre el software y el GPS

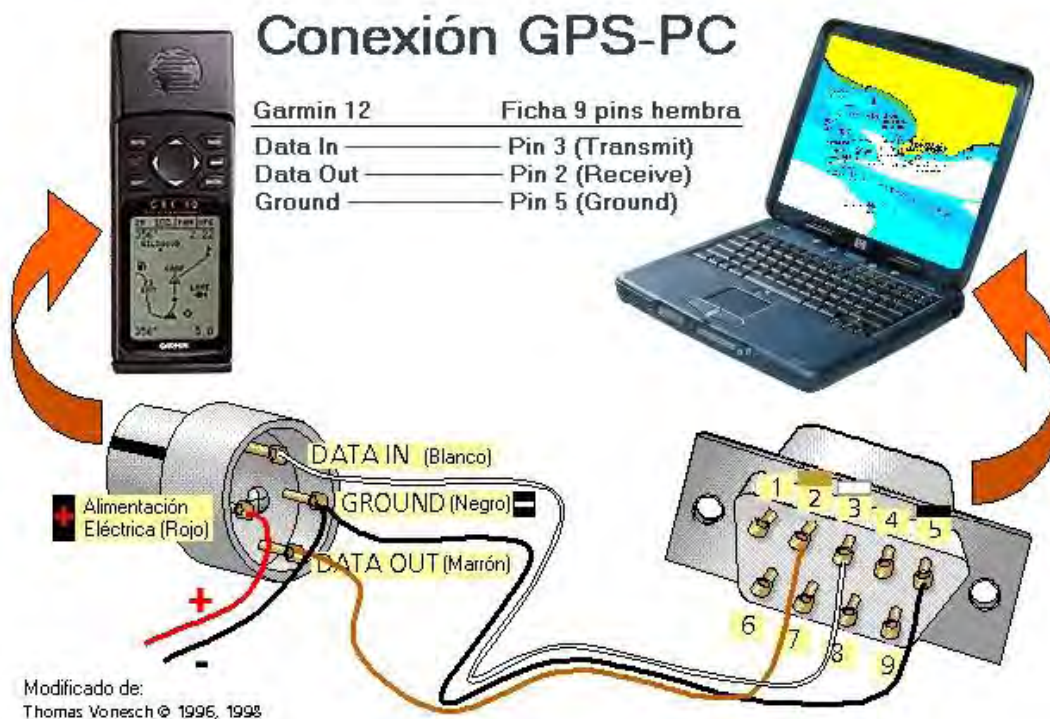
La conectividad entre un GPS y una PC es primordial para permitir el intercambio de información entre ambos dispositivos y, del mismo modo, lograr interactuar con los receptores más fácilmente, permitiendo crear, editar y manipular puntos de referencia (waypoints), rutas, pistas (track) y cartografía ya sea marítima o terrestre, con la ayuda de software específico.

Los GPS transmiten datos a través de un puerto serie o puerto COM. Los puertos COM son puertos de comunicación serie y estos pueden ser físicos (equipos antiguos incluían un conector de 9 pines en la placa base) o puertos virtuales (puertos que se crean al conectar un GPS por Bluetooth o USB).

### 2.4.1 Conexión por puerto Serial

El primer modelo de comunicación que se analizará es el puerto serial. Esta conexión GPS-PC casera es muy rudimentaria pero ayuda a conseguir el objetivo. Por un lado se cuenta con 4 terminales que van a ir conectadas al GPS, estas terminales son para la alimentación eléctrica, la tierra, los datos de entrada y los datos de salida. Los otros extremos del cable van conectados a un conector macho de 9 pines, de los cuales sólo se usarán el pin dos (salida de datos), el pin tres (entrada de datos) y el pin 5 (tierra).

Otro aspecto a tener en cuenta es el voltaje que soporta el GPS, ya que existen dispositivos que poseen un poste en el centro del conector, lo cual indica que el voltaje soportado es de 6 volts mientras que los que no poseen este poste trabajan con 12 volts.



**Figura 2.7.** Conexión GPS-PC por puerto serial (1)

Hay que tener en cuenta que existen diferentes entradas según el modelo de GPS que se esté utilizando, sólo hay que identificar los pines de datos y de corriente para realizar la conexión.



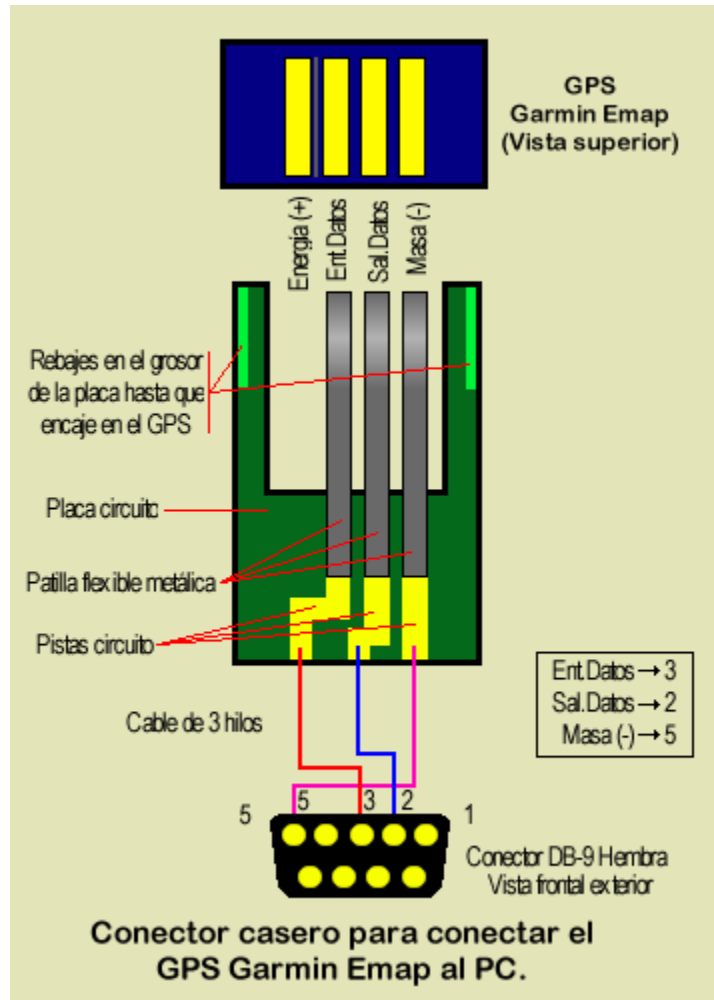


Figura 2.8. Conexión GPS-PC por puerto serial (2)

Ya que se posee conexión, se procede a verificar que funcione correctamente. Para ello se pueden pasar datos de la PC al GPS o viceversa o se puede iniciar alguna aplicación del fabricante. En este caso se va a ejemplificar con OziExplorer que es un software de Mapas para GPS y que tiene compatibilidad con diferentes dispositivos.

Lo primero que hay que hacer es configurar el GPS para que trabaje con el protocolo NMEA, para que el dispositivo quede trabajando con los parámetros "NMEA 0183 2.0" y a "4800 Baudios". Existen diferentes opciones para elegir el sistema que va a ocupar el dispositivo GPS, para este caso se elige la opción "Simulador" y se configuran la velocidad y el rumbo y/o posición que se le dará a nuestro simulador.

Una vez configurado el dispositivo GPS se procede a abrir la aplicación en la PC. Una vez ejecutado OziExplorer se despliega el menú "Mapa Móvil", se elige "Iniciar comunicación NMEA con el GPS". Si todo sale bien, en el mapa móvil se desplazará el simulador (un barco virtual).

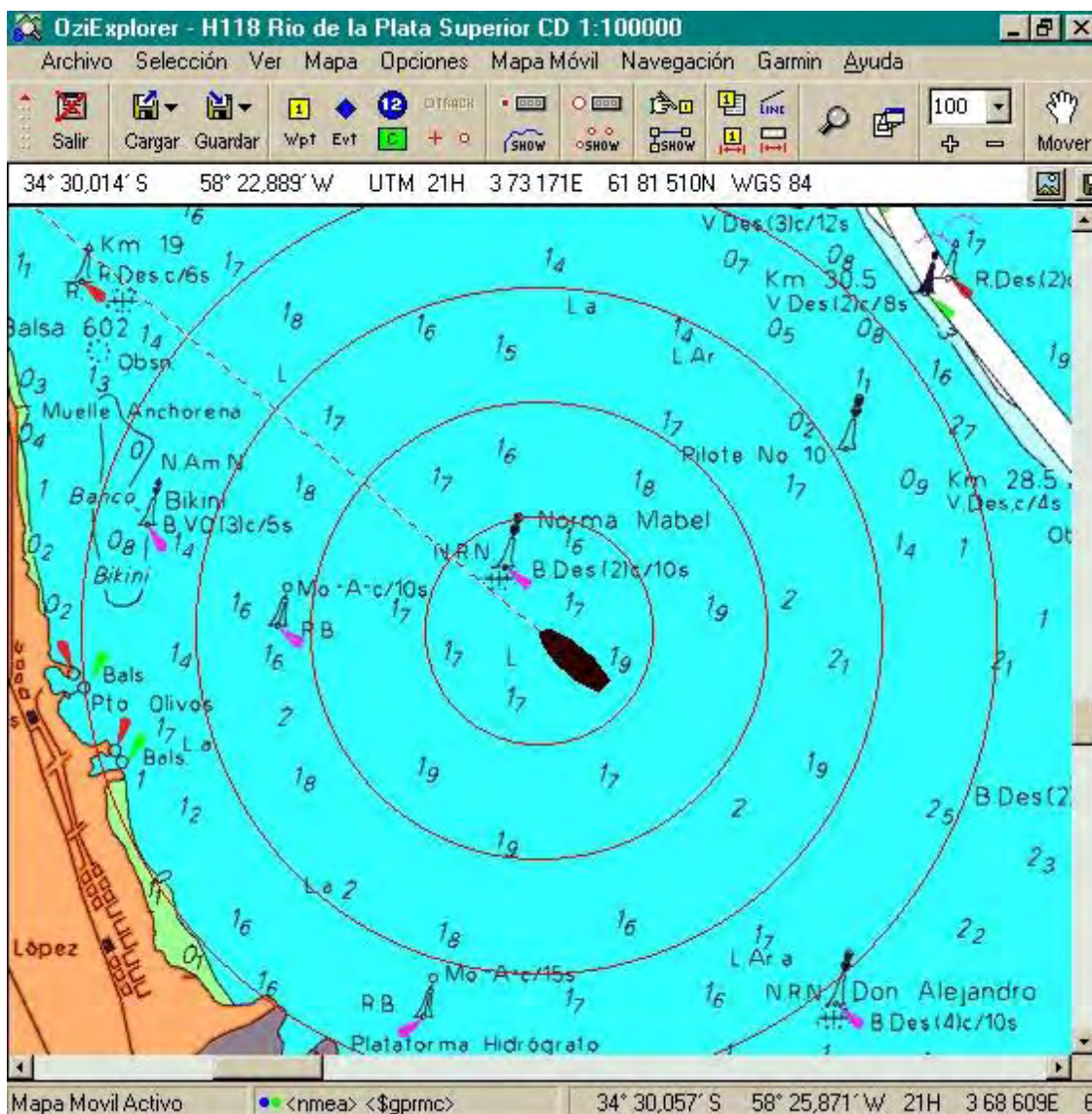


Figura 2.9. Conexión GPS con software OziExplorer.

### 2.4.2 Conexión por puerto USB o por Bluetooth

Existen otras dos maneras de realizar la conexión GPS-PC. La primera es usando un cable USB. La otra conexión se puede realizar mediante el protocolo Bluetooth. Para ambos casos, la PC crea un puerto COM virtual por el que recibe los datos de nuestro dispositivo GPS.

Para poder recibir los datos del dispositivo GPS es necesario saber en qué puerto COM se encuentra conectado, para ello presionamos el botón derecho del mouse sobre el ícono “Mi PC”, y se elige la opción “Propiedades”. Una vez abierta la pestaña “Propiedades del sistema”, se presiona la pestaña que dice “Hardware” y ahí se presiona el botón “Administrador de dispositivos”. Se abre una ventana que muestra los dispositivos con los que cuenta la PC. Entre estos dispositivos se encuentra la opción “Puertos (COM y LPT)”, ahí se debe encontrar la conexión con el dispositivo GPS.

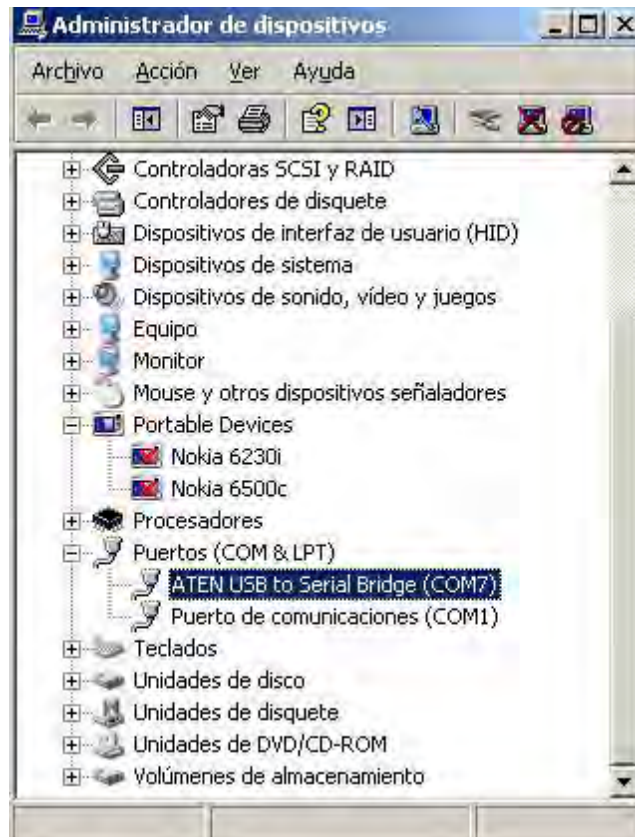


Figura 2.10. Administrador de dispositivos.

Una vez identificado el puerto COM con el que están conectados el dispositivo GPS y la PC, se lee la información con alguna aplicación que soporte este protocolo. Se va a utilizar OpenCPN que es un programa simple para navegación y cartografía náutica disponible bajo licencia GNU. Una vez abierta la aplicación, se presiona el ícono Herramientas, se abre una nueva ventana, en ella se elige la pestaña GPS y se busca (o se escribe) el puerto donde se encuentra conectado el GPS.

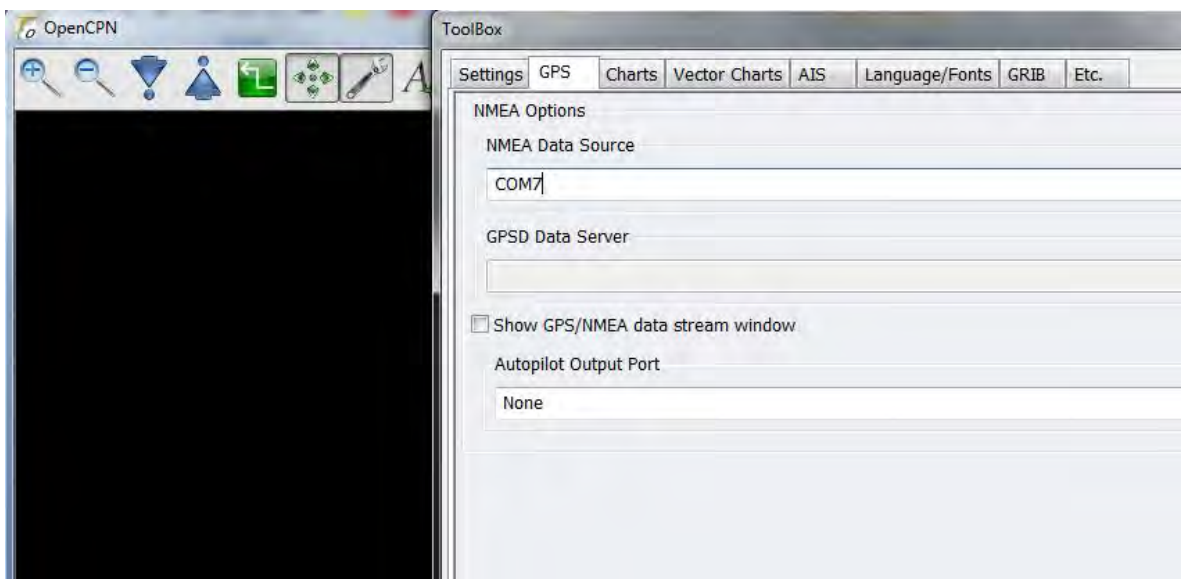
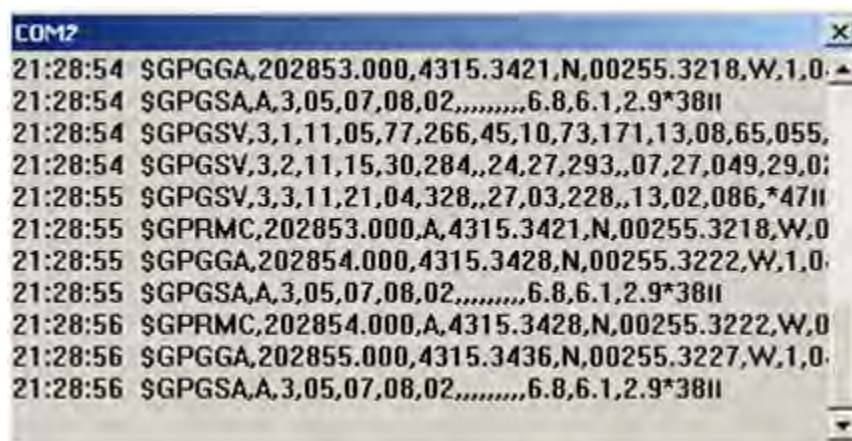


Figura 2.11. OpenCPN recibe datos del puerto COM7.

Para verificar que el enlace se realice correctamente, se activa la casilla “Mostrar datos de conexión” y una nueva ventana mostrará la información que recibe OpenCPN desde el GPS.



```

COM2
21:28:54 $GPGGA,202853.000,4315.3421,N,00255.3218,W,1,0,
21:28:54 $GPGSA,A,3,05,07,08,02,,,,,,,,,6.8,6.1,2.9*38II
21:28:54 $GPGSV,3,1,11,05,77,266.45,10,73,171,13,08,65,055,
21:28:54 $GPGSV,3,2,11,15,30,284,,24,27,293,,07,27,049,29,0;
21:28:55 $GPGSV,3,3,11,21,04,328,,27,03,228,,13,02,086,*47II
21:28:55 $GPRMC,202853.000,A,4315.3421,N,00255.3218,W,0
21:28:55 $GPGGA,202854.000,4315.3428,N,00255.3222,W,1,0,
21:28:55 $GPGSA,A,3,05,07,08,02,,,,,,,,,6.8,6.1,2.9*38II
21:28:56 $GPRMC,202854.000,A,4315.3428,N,00255.3222,W,0
21:28:56 $GPGGA,202855.000,4315.3436,N,00255.3227,W,1,0,
21:28:56 $GPGSA,A,3,05,07,08,02,,,,,,,,,6.8,6.1,2.9*38II

```

Figura 2.12. Datos recibidos del GPS en protocolo NMEA.

Una vez que el enlace se haya llevado a cabo con éxito, se puede empezar a interactuar con el software y la información enviada por el GPS.

### 2.4.3 Conexión con PDA

Un dispositivo GPS también puede ser conectado a un PDA (Asistente Digital Personal) mediante cable o por Bluetooth.

Para llevar a cabo una conexión por Bluetooth, es necesario que el PDA posea un puerto receptor para este protocolo de comunicación y estar provisto de los drivers necesarios para su correcto funcionamiento. De igual manera, si el dispositivo PDA no dispone de Bluetooth, puede adquirirse un puerto externo.

Una vez que los puertos Bluetooth se encuentran correctamente configurados, para conectar el GPS al PDA, simplemente se debe ejecutar el software adecuado que lea la información del GPS.

Otra manera de conectar el GPS al PDA es mediante un cable, en este caso, los puertos del PDA deben encontrarse funcionando correctamente con los drivers correctamente instalados. La mayoría de los sistemas operativos actuales que trabajan en los PDA, poseen integrados estos drivers, tanto para conexiones PS/2 como USB.

Se debe disponer de un cable conector apropiado. Debe tenerse en cuenta el tipo de conector que dispone tanto el GPS como el PDA, ya que, como se observó al principio de este tema, los conectores de dispositivos antiguos, varían en la cantidad de pines que poseen, lo que puede llevar a realizar modificaciones en el cable conector.

## Análisis de la aplicación

En la figura 2.13 se presenta un diagrama que ejemplifica una conexión serial para PDA.

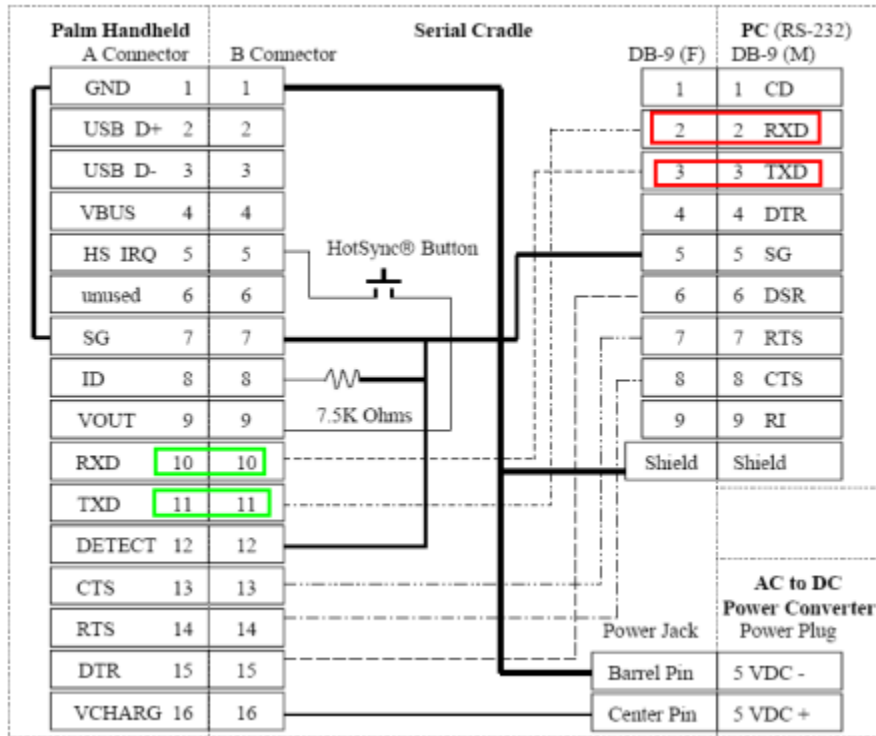


Figura 2.13. Diagrama de conexión serial para PDA.

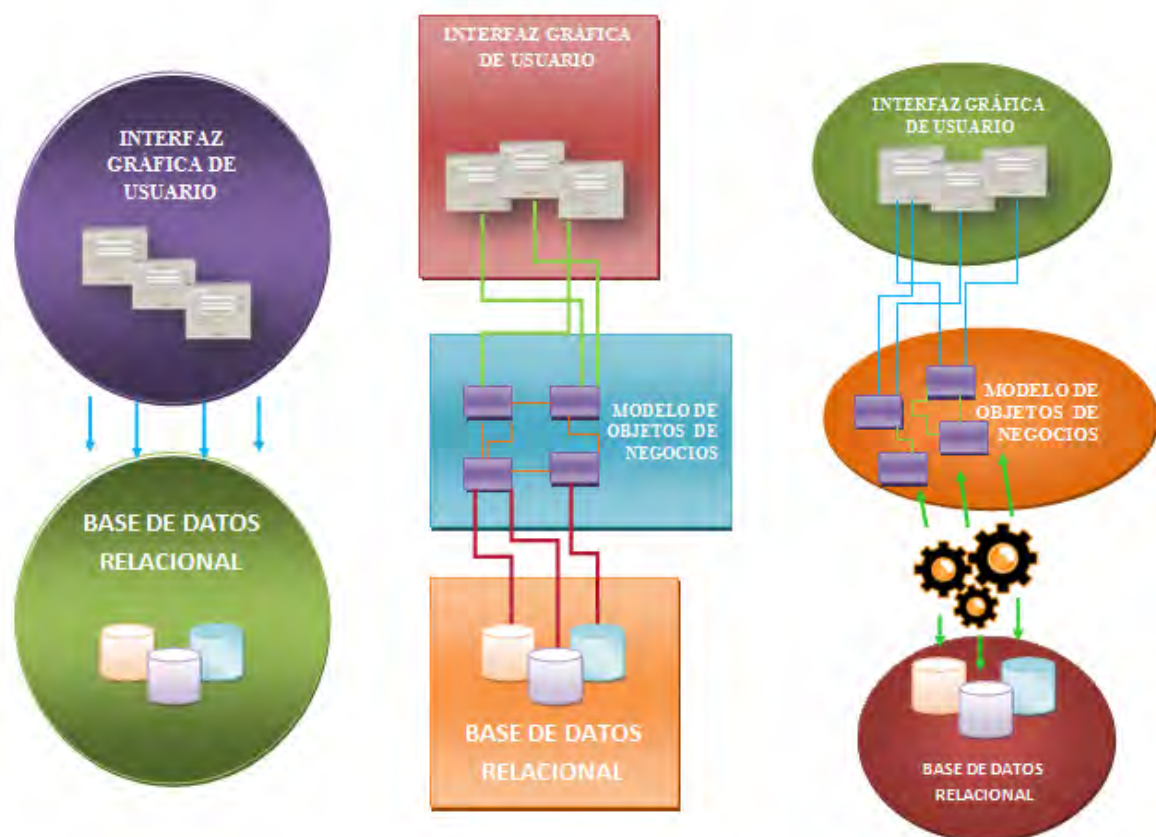
En el diagrama podemos observar que es necesario utilizar los pines 10 y 11 del conector que va en el PDA para los datos y el 7 o 12 para la tierra del GPS. Una vez terminado el cable y lograda la conexión, se debe ejecutar el software que recibe la señal del GPS en el PDA.



Figura 2.15. Conexión entre GPS y PDA.

## Capítulo 3. Arquitectura de la Aplicación

Arquitectura de la aplicación es un término usado al diseñar aplicaciones, particularmente del tipo Cliente-Servidor. Esta arquitectura se refiere a la manera en la que es diseñada tanto física como lógicamente y/o conceptualmente una aplicación. La figura 3.1 muestra las 3 diferentes arquitecturas: conceptual, lógica y física.



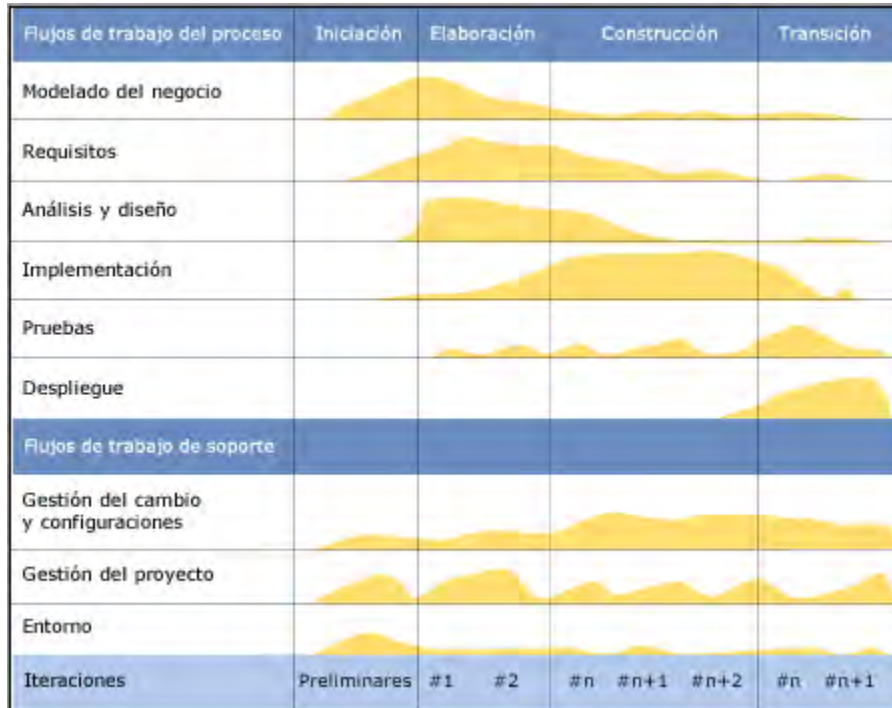
**Figura 3.1.** Arquitectura lógica de tres capas de una aplicación cliente/servidor.

Cuando se menciona el término cliente/servidor es común pensar en bases de datos, sin embargo, la arquitectura cliente/servidor es un modelo pensado en el desarrollo de sistemas de información en que las operaciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. El cliente es el proceso que inicia la operación y solicita los recursos y el servidor es el proceso que responde las solicitudes.

En el capítulo 2 se mencionaron las fases del ciclo de vida del software: análisis, diseño, programación y validación. En el análisis se define el alcance del proyecto y los casos de uso. En el diseño se proyecta un plan a seguir, se definen las características y se establece la arquitectura. La programación permite crear el producto y la validación permite establecer la utilidad de la aplicación para con los usuarios finales.

## Arquitectura de la Aplicación

Estas fases del ciclo de vida definen las diferentes arquitecturas que contempla la construcción de una aplicación. El análisis define la arquitectura conceptual, el diseño define la arquitectura lógica de la aplicación y la programación y la validación define la arquitectura física del producto terminado.



**Figura 3.2.** Estructura del Proceso Unificado.

En esta fase de arquitectura o diseño de la aplicación se establece el comportamiento dinámico del sistema, es decir, como debe reaccionar ante los acontecimientos. El resultado obtenido de esta etapa facilita enormemente la implementación posterior del sistema, pues proporciona la estructura básica del sistema y la manera en que los diferentes componentes actúan y se relacionan entre ellos.

Una buena metodología de diseño consiste en utilizar patrones. Un patrón de diseño ofrece una solución concreta a los posibles problemas que se puedan encontrar en la construcción del sistema, indicando la mejor forma de modelar los objetos. Una vez aplicados, su implementación en lenguajes orientados a objetos no presenta ningún problema adicional.

La elección de la arquitectura interna del sistema es otro de los puntos básicos de la etapa de diseño. La arquitectura más utilizada es en capas, el número de las cuáles depende de diferentes factores, aunque la más utilizada es la arquitectura en 3 capas: Presentación, Dominio y Acceso a datos.

### 3.1 Arquitectura Conceptual

La arquitectura o diseño conceptual se considera como un análisis de actividades, consiste en la solución para el usuario y se expresa con los casos de uso. El diseño conceptual es la solución del proyecto y consiste de las siguientes tareas:

- Identificar los usuarios y sus roles.
- Obtener datos de los usuarios.
- Evaluar la información.
- Documentar los escenarios de uso.
- Validar con los usuarios.

Una forma de obtener estos requerimientos es construir una matriz usuarios-actividades. Para el caso de la aplicación que conecta el dispositivo móvil con el GPS sólo se tiene un usuario, el usuario final. Así mismo, existen 3 actividades explícitas para este usuario: Mostrar ubicación, mostrar ayuda y salir de la aplicación.

|               | Mostrar Ubicación                        | Mostrar Ayuda   | Salir de la Aplicación |
|---------------|--|---|------------------------|
| Usuario Final | Muestra la imagen de la posición actual. | Muestra un pequeño manual de cómo usar la aplicación. | Cierra la aplicación.  |

**Tabla 3.1.** Matriz usuarios-actividades.

#### 3.1.1 Casos de Uso

**Caso de uso:** Mostrar Ubicación.

**Actores:** Usuario final.

**Propósito:** Mostrar mapa en pantalla con la posición actual del usuario.

**Resumen:** Se sincroniza el celular con el GPS vía Bluetooth para obtener las coordenadas de la posición actual. Al final se muestra la imagen del mapa que muestra la posición actual del usuario.

**Curso normal de eventos:** Al presionar el botón “Mostrar Ubicación”, la aplicación en el celular establece una conexión con el GPS mediante el protocolo Bluetooth para obtener las coordenadas correspondientes a la posición actual del usuario. Una vez obtenidas las coordenadas, se procede a mostrar el mapa correspondiente con la posibilidad de interactuar con él (acercar, alejar, mover en diferentes direcciones).

**Curso alternativo:** Al presionar el botón “Mostrar Ubicación”, la aplicación en el celular intenta establecer una conexión con el GPS mediante el protocolo Bluetooth. En caso de que la conexión no se lleve a cabo se envía un mensaje de error. Si la conexión se establece satisfactoriamente se procede a buscar el mapa según las coordenadas. Si el mapa no se puede encontrar se envía un mensaje de error.



**Caso de uso:** Mostrar Ayuda.

**Actores:** Usuario final.

**Propósito:** Mostrar instrucciones de cómo usar la aplicación.

**Resumen:** Se muestra un resumen de las acciones que realiza la aplicación al momento de presionar el botón “Mostrar Ubicación”. Así mismo explica los mensajes de error en caso de que los procesos de sincronización o despliegado de mapas no se lleven a cabo.

**Curso normal de eventos:** Al presionar el botón “Mostrar Ayuda”, la aplicación muestra un formulario con una explicación detallada de qué hace la aplicación al momento de presionar el botón “Mostrar Ubicación” así como posibles errores en caso de no lograr la sincronización con el dispositivo GPS o en caso de no encontrar los mapas en el celular.

**Curso alternativo:** En caso de que el formulario que muestra la ayuda no se pueda cargar, se muestra un mensaje de error.

**Caso de uso:** Salir.

**Actores:** Usuario final.

**Propósito:** Salir de la aplicación.

**Resumen:** La ejecución abandona el proceso actual y termina la aplicación para regresar el control al sistema operativo del celular.

**Curso normal de eventos:** Al presionar el botón “Salir”, termina la aplicación y regresa el control al celular.

**Curso alternativo:** Si la aplicación deja de escuchar los eventos del teclado, es necesario regresar el control al celular por medio el botón colgar que, generalmente, regresa a la pantalla principal del celular o, en último de los casos, apagando el celular.

### 3.1.2 Diagramas de Casos de Uso

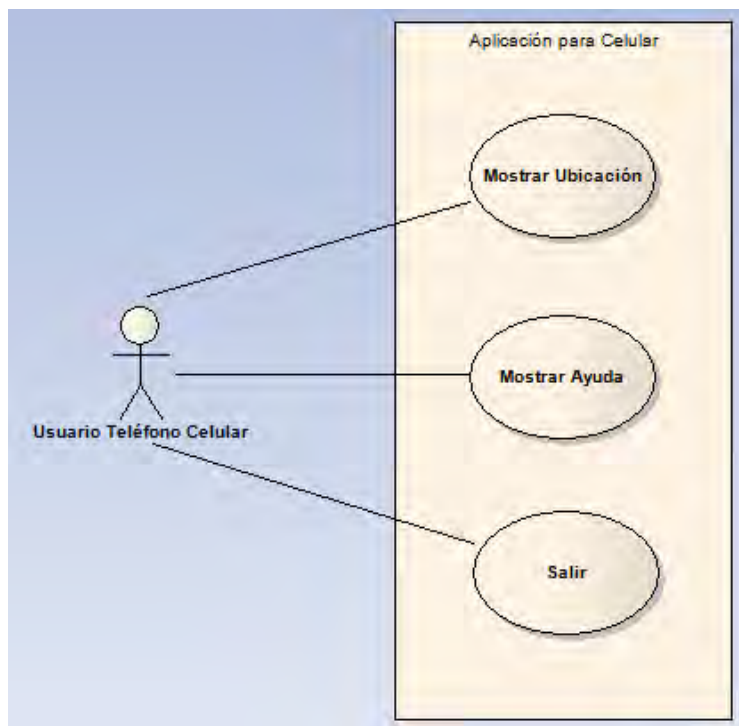


Figura 3.3. Diagrama de Casos de Uso de la aplicación.

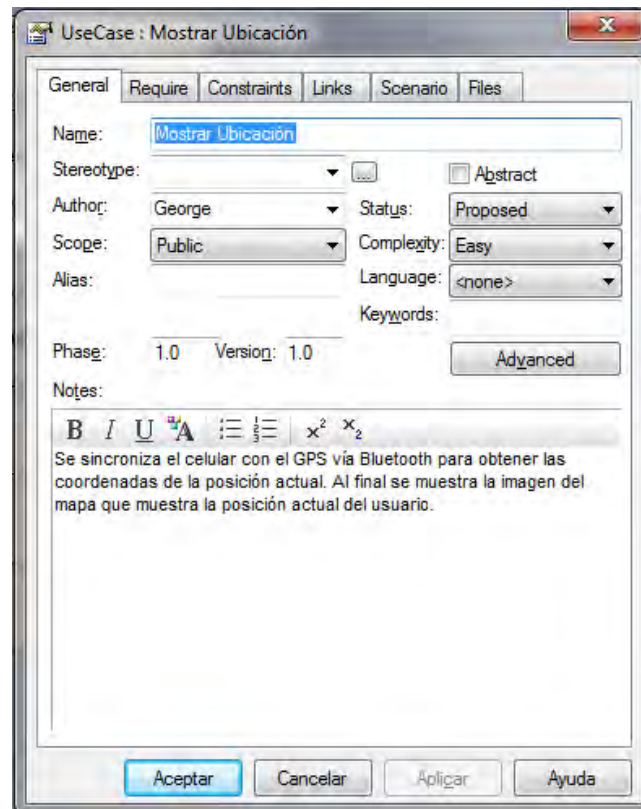


Figura 3.4. Caso de uso de la actividad “Mostrar Ubicación”.

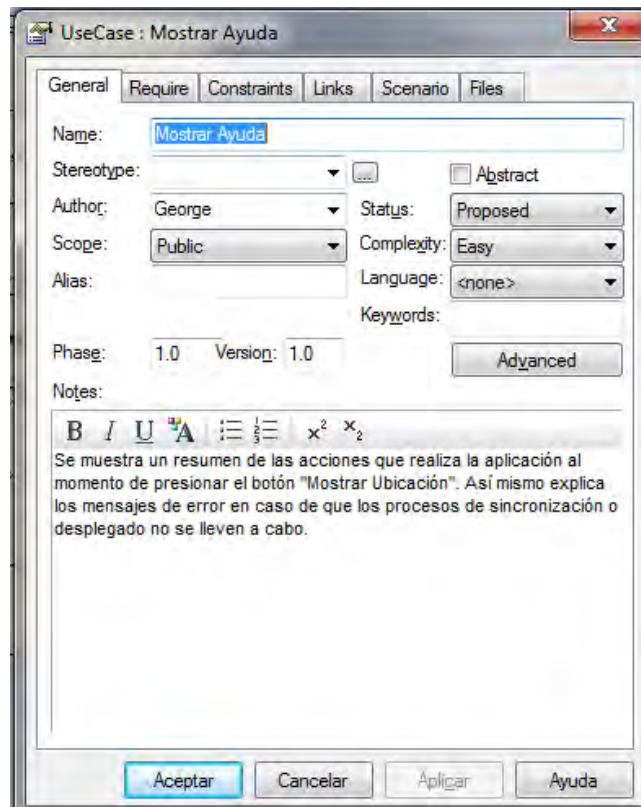


Figura 3.5. Caso de uso de la actividad “Mostrar Ayuda”.

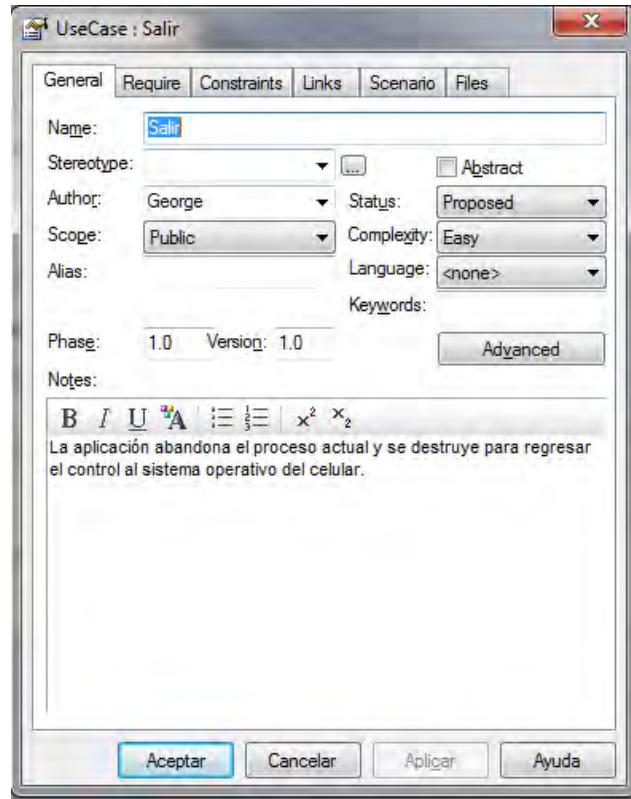


Figura 3.6. Casos de uso de la actividad “Salir”.

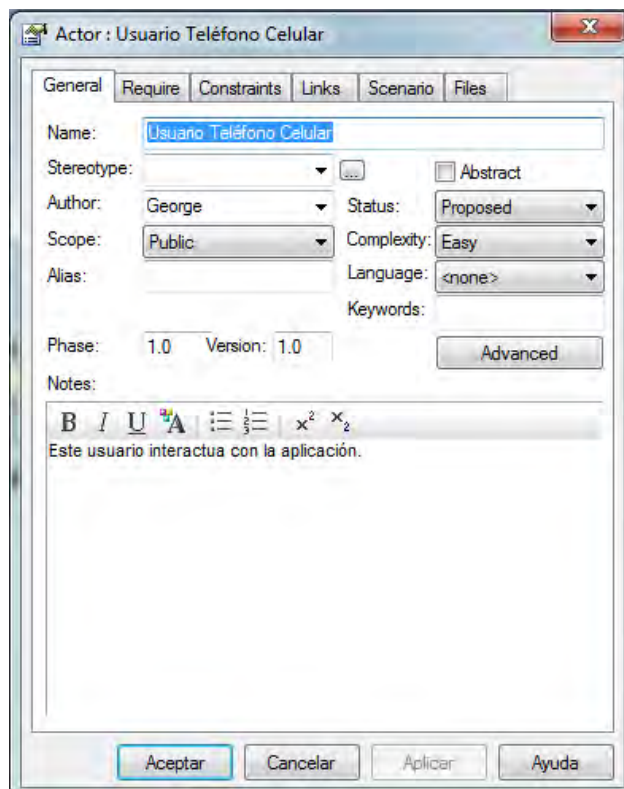


Figura 3.7. Descripción del actor “Usuario Teléfono Celular”.

## 3.2 Arquitectura Lógica

La arquitectura lógica o diseño lógico es una descripción de los requisitos funcionales del sistema a realizar, es decir, define lo que hará el sistema para resolver los problemas identificados en el análisis previo.

El diseño lógico traduce los escenarios de uso creados en el diseño conceptual en un conjunto de objetos de negocio y sus servicios. Se convierte, en parte, en la especificación funcional que se usa en la arquitectura física. Es independiente de la tecnología. Refina, organiza y detalla la solución de negocios y define formalmente las reglas y políticas específicas de negocios.

Un objeto de negocio es la encapsulación de un servicio que abstrae las cualidades esenciales de un tema de interés.

Un servicio es una unidad con capacidad de cómputo. Un servicio debe satisfacer lo siguiente:

- Ser seguro: lo que equivale a un uso correcto y con autorización.
- Ser válido: qué tareas o reglas se pueden aplicar.
- Manejar excepciones: informar al cliente cualquier error.
- Contar con un catálogo de servicios que constituye un repositorio de servicios.

Los objetos de negocio deben verificarse y probarse de tal manera que asegure que los módulos operen como unidades completas de trabajo. Las tareas de verificación incluyen:

- Una verificación independiente:
  - Pre y post condiciones
  - Lógica y funcionalidad individual
- Una verificación dependiente:
  - Verificación de dependencias
  - Que operan como una unidad específica de trabajo

El diseño lógico comprende las siguientes tareas:

- Identificar y definir los objetos de negocio y sus servicios
- Definir las interfaces
- Identificar las dependencias entre objetos
- Validar contra los escenarios de uso
- Revisar y refinar tanto como sea necesario

Una interfaz tiene las siguientes partes:

- Nombre
- Precondiciones, lo que debe estar presente antes de ejecutarse
- Poscondiciones, estado final
- Capacidad o funcionalidad (SQL, pseudocódigo, función matemática)
- Dependencias

## Arquitectura de la Aplicación

La tarea de identificar las dependencias entre objetos permite identificar eventos, sucesos o condiciones que permitan realizar tareas de negocios coordinadamente o por transacciones. Para ello se debe considerar lo siguiente:

- Identificar los eventos disparadores (triggers)
- Determinar cualquier dependencia (existencial o funcional)
- Determinar cualquier problema de consistencia o secuencia
- Identificar cualquier regulación de tiempo crítica
- Considerar posibles problemas (transacciones)
- Identificar y auditar los requerimientos de control
- Determinar lugares y dependencias a través de la ubicación
- Determinar cuando el servicio que controla la transacción es dependiente de los servicios contenidos en otros objetos de negocio

La validación del modelo lógico debe ser tal que éste sea:

- Completo: debe representar todos los escenarios de uso,
- Correcto: el comportamiento lógico debe corresponder con el comportamiento conceptual, y
- Claro: los objetos de negocio y servicios no deben ser ambiguos.

### 3.2.1 Niveles de Servicios

El diseño lógico, conceptualmente, se divide en tres niveles de servicios: servicios de usuario, servicios de negocio y servicios de datos. Esto con el fin de que la aplicación resulte flexible ante los cambios de requerimientos y/o de tecnología, cambiando únicamente la capa o capas necesarias.

#### 3.2.1.1 Servicios de Usuario

Los servicios de usuario (user services) controlan la interacción con la aplicación o sistema. Un servicio de usuario son personas, aplicaciones, otros servicios o la combinación de éstos. Generalmente involucra una interfaz gráfica de usuario (GUI), pero puede ser no visual (mensajes o funciones), maneja todos los aspectos de la interacción con la aplicación. El objetivo central es minimizar el esfuerzo de conocimiento requerido para interpretar la información. Un servicio de usuario incluye un contenido (qué se necesita comunicar al usuario) y una forma (cómo se comunica el contenido) cuando es necesaria la comunicación.

#### 3.2.1.2 Servicios de Negocio

Los servicios de negocio (business services) convierten los datos recibidos de los servicios de datos y de usuario en información (datos + regla de negocio) y pueden usar otros servicios de negocio para completar su tarea.

Las tareas de los servicios de negocio son:

- Dar formato a los datos
- Obtener y mover datos desde y hasta los servicios de datos
- Transformar los datos en información
- Validar los datos inmediatamente en el contexto o en forma diferida una vez terminada la transacción.

### 3.2.1.3 Servicios de Datos

Los servicios de datos (data services) son los servicios de bajo nivel que apoyan los servicios de negocio y son de una amplia gama de categorías como las siguientes:

- Declaración del esquema del servicio.
- Respaldo y recuperación en caso de falla.
- Búsqueda, compilación, optimización y ejecución de solicitudes.
- Procesar modificaciones.
- Permite al acceso concurrente a los datos.
- Validación de datos.
- Permisos a usuarios, grupos y servicios.
- Administración de la conexión.
- Distribución de datos.

El diseño lógico abarca planear el propósito de cada elemento del sistema, sin hacer consideraciones de hardware y software.

- Diseño de salida. Es una descripción de todas las salidas del sistema que incluye: tipos, formato, contenido y frecuencia.
- Diseño de entrada. Se especifican los tipos y formatos de entrada.
- Diseño de procesamiento. Describe los cálculos, comparaciones y manipulación de datos que necesita el sistema.
- Diseño de archivos y bases de datos. En muchos sistemas de información se requieren subsistemas de archivos y bases de datos. Las características de estos subsistemas así como los diagramas de flujo de datos y de entidad relación se especifican también en esta fase.
- Diseño de telecomunicaciones. Es necesario especificar los sistemas de redes y telecomunicaciones que se van a necesitar.
- Diseño de procedimientos. Describir los procedimientos para la ejecución de aplicaciones y la solución de problemas que surjan.
- Diseño de controles y seguridad. Determinar la frecuencia y características necesarias de los sistemas de respaldo así como considerar la prevención y recuperación de un desastre en el dispositivo.
- Diseño de personal y empleos. En caso de que se requiera contratar empleados adicionales.

### 3.2.2 Diagramas de Actividad

A continuación se definen los diagramas de actividad de la aplicación propuesta. El primer diagrama muestra la secuencia de pasos que se plantea siga la opción "Mostrar Ubicación". Al ser presionado el botón, la aplicación intenta conectarse con el dispositivo GPS vía Bluetooth, este proceso es llamado Sincronización. En caso de que la sincronización se lleve a cabo se procede a buscar el mapa, de lo contrario, marca un error y regresa al menú. Durante el proceso de sincronización lo que se obtiene son las coordenadas de la posición en la que se encuentra el dispositivo. En caso de que las coordenadas estén en la base de datos (o en la estructura de datos) se muestra el mapa, de lo contrario, marca un error y regresa al menú.

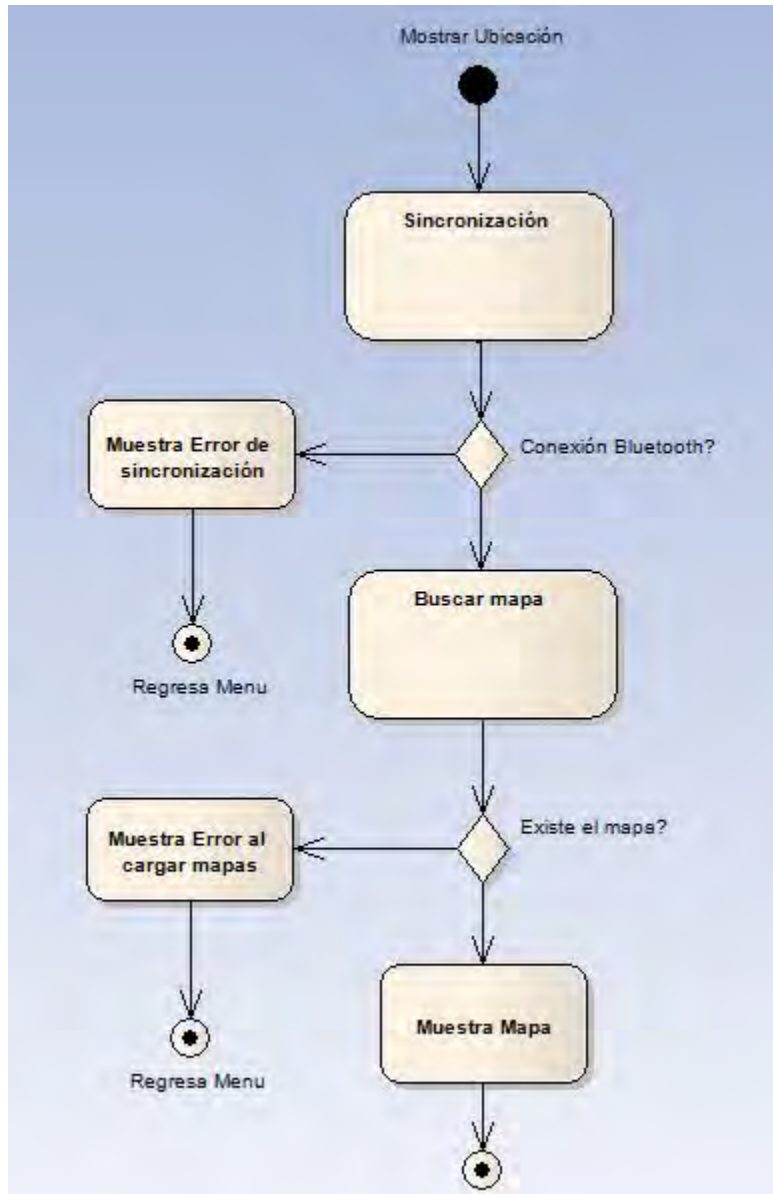


Figura 3.8. Secuencia de actividades del botón “Mostrar Ubicación”.

El siguiente diagrama muestra la actividad del botón “Mostrar Ayuda”. Cuando se presiona este botón se debe desplegar un formulario que explique la actividad que realiza el botón “Mostrar Ubicación”. Esto con el fin de explicar al usuario el proceso que sigue la aplicación. Así mismo se intenta advertir al usuario los mensajes de error que puede generar la aplicación para que esté enterado y, en medida de lo posible, pueda resolverlo. En caso de que el formulario no se encuentre activo o no tenga información, se envía un error y se regresa al menú.

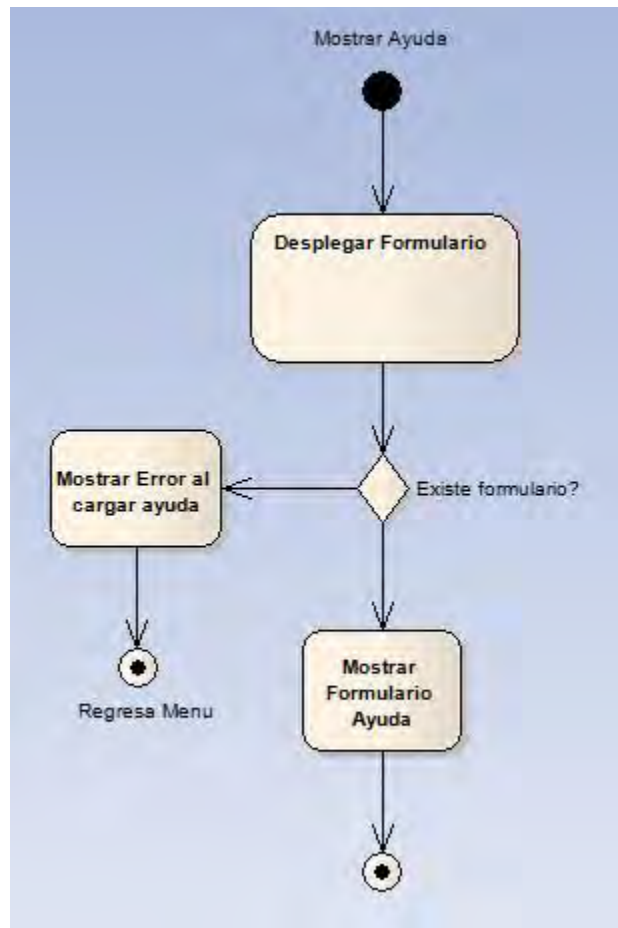


Figura 3.9. Secuencia de actividades del botón "Mostrar Ayuda".

El botón "Salir" finaliza el proceso actual (en caso de que exista alguno) y termina la aplicación. Es importante destacar que este botón debe existir en todas las ventanas de la aplicación.

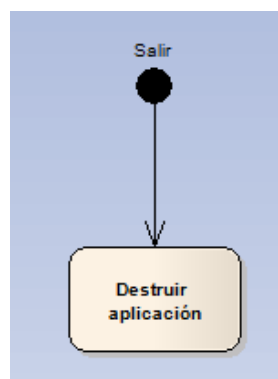


Figura 3.10. Secuencia de actividades del botón "Salir".



### 3.3 Arquitectura Física

La arquitectura física o diseño físico traduce el diseño lógico en una solución que se puede implementar y sea económicamente viable.

El diseño físico de sistemas define la manera en que se cumplirán las tareas del sistema, esto incluye unir todos sus componentes y las funciones que realizará cada uno de éstos.

La arquitectura física debe tomar en cuenta:

- El diseño para la distribución de datos: cantidad de datos que se envían entre componentes así como el medio de transmisión.
- El diseño multitarea: permitir y administrar procesos concurrentes.
- El manejo de errores y prueba de eventos:
  - Validar valores de entrada.
  - Proteger procesos críticos.
  - Proteger datos importantes.
  - Depurar el sistema.
  - Proteger las transacciones de negocios.

En esta fase se especifican las características de los componentes del sistema requeridos para poner en práctica el diseño lógico.

- Diseño de hardware. Debe especificarse todo el equipo de cómputo, lo que incluye dispositivos de entrada, procesamiento y salida, con sus características de rendimiento.
- Diseño de software. Deben especificarse las características de todo el Software. Las especificaciones de diseño lógico, en cuanto a requisitos de salida, entrada y procesamiento de los programas, se toman en cuenta durante aquí.
- Diseño de bases de datos. Es necesario detallar el tipo, estructura y funciones de las bases de datos.
- Diseño de telecomunicaciones. Deben especificarse las características necesarias del software, medios y dispositivos de telecomunicaciones.
- Diseño de personal. Hay que incluir información personal sobre el personal con el fin de obtener un perfil útil para manipular la aplicación.
- Diseño de procedimientos y controles. Detalla la forma en que se ejecuta cada aplicación y las medidas para minimizar las probabilidades de errores, delitos y fraudes.

### 3.3.1 Bocetos de la Aplicación

Diseño de hardware: Dispositivo celular que soporte la tecnología Java y que posea Bluetooth.

Diseño de software: Diseño orientado a objetos (J2ME).

Diseño de base de datos: Se puede usar la base de datos del celular o una estructura de datos.

Diseño de telecomunicaciones: Se debe establecer una comunicación serial entre el celular y el dispositivo GPS para recibir la información.

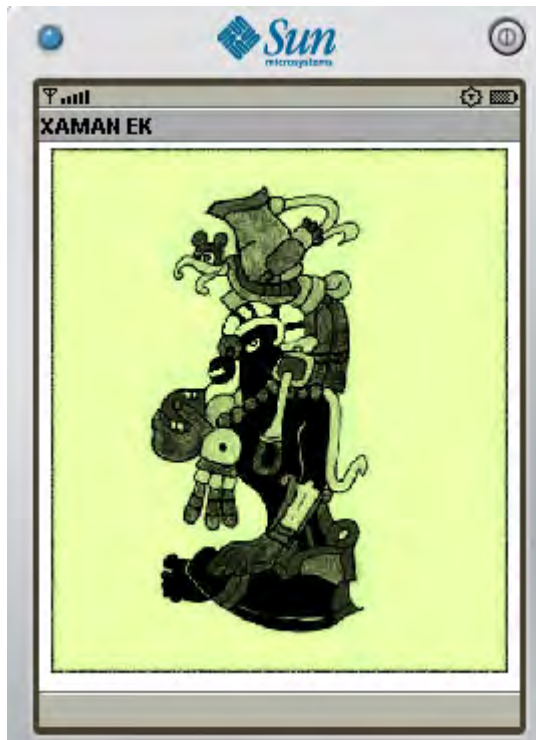


Figura 3.11. Alerta de la aplicación.



Figura 3.12. Menú principal de la aplicación

# Capítulo 4. Implantación

Existen muchas y muy diversas aplicaciones que permiten manipular los datos obtenidos desde un dispositivo GPS tanto en PC's como en PDA's o, incluso, teléfonos celulares. En este capítulo se van mencionar y ejecutar algunos. De igual manera se analizarán ventajas y desventajas de estas aplicaciones para tener parámetros que confirmen la viabilidad de la aplicación que se plantea desarrollar.

## 4.1 Conexión entre el GPS y dispositivos

### 4.1.1 Google Earth

La primera aplicación que se analizará es de Google y permite el uso de dispositivos GPS con su conocida aplicación Google Earth. Este software permite importar desde el GPS puntos de ruta y datos de seguimiento. Sin embargo, estas funciones sólo están disponibles para usuarios de Google Earth Plus, Google Earth Pro y Google Earth EC, pero no para la versión gratuita.

#### 4.1.1.1 Ventajas

Actualmente, Google Earth es compatible con la mayoría de los dispositivos GPS de los fabricantes Garmin y Megellan, y se plantea el uso de un cable USB para conectar el dispositivo GPS con la aplicación en la PC. Para los dispositivos que no son de estas marcas, se plantea la posibilidad de probar importar datos GPS como archivo .gpx o .loc y después intentar abrirlo en Google Earth (Archivo > Abrir).



Figura 4.1. Google Earth Pro, para usos profesionales y comerciales.

De igual manera, existe la posibilidad de visualizar la información que envía el GPS en tiempo real, así como tener una visualización de la línea de tiempo de diferentes datos (o seguimientos) que tenga la aplicación.

Si el software es instalado en un equipo portátil, es posible ver la información del GPS reflejada en la aplicación en tiempo real. Por ejemplo, es posible hacer un seguimiento en vivo de un viaje en coche. Para ello se conecta el dispositivo GPS al equipo portátil, en el cuadro de diálogo "GPS" se selecciona la pestaña "Tiempo real" y se elige las opciones adecuadas: Protocolo NMEA, los puntos

## Implantación

de importación, intervalo de sondeo de la aplicación (en segundos), entre otras. Al final se presiona “Iniciar” para comenzar el seguimiento.

De igual manera, es posible visualizar una línea de tiempo con los datos descargados a la aplicación, es decir, si se posee información sobre periodos de tiempo, se puede ver ésta de forma secuencial.

### 4.1.1.2 Desventajas

La mayor desventaja de esta aplicación es que estas características no están disponibles para la versión gratuita y la versión Google Maps Pro (que es la que se documentó) vende su licencia en 400 dólares anuales.

### 4.1.2 OziExplorer

OziExplorer es un programa interactivo que usa mapas de imágenes convencionales y algunos en formatos vectoriales, que permite planificar viajes, realizar funciones de Mapa Móvil, permite crear y añadir puntos de referencia (Waypoints), rutas y pistas (tracks). Además permite el intercambio de información entre receptores GPS y una PC.

#### 4.1.2.1 Ventajas

Este programa soporta receptores GPS de las marcas Magellan, Garmin, Lowrance/Eagle, Brunton/Silva y MLR. Puede realizar las funciones de Mapa Móvil con la mayoría de los receptores que emitan sentencias NMEA.

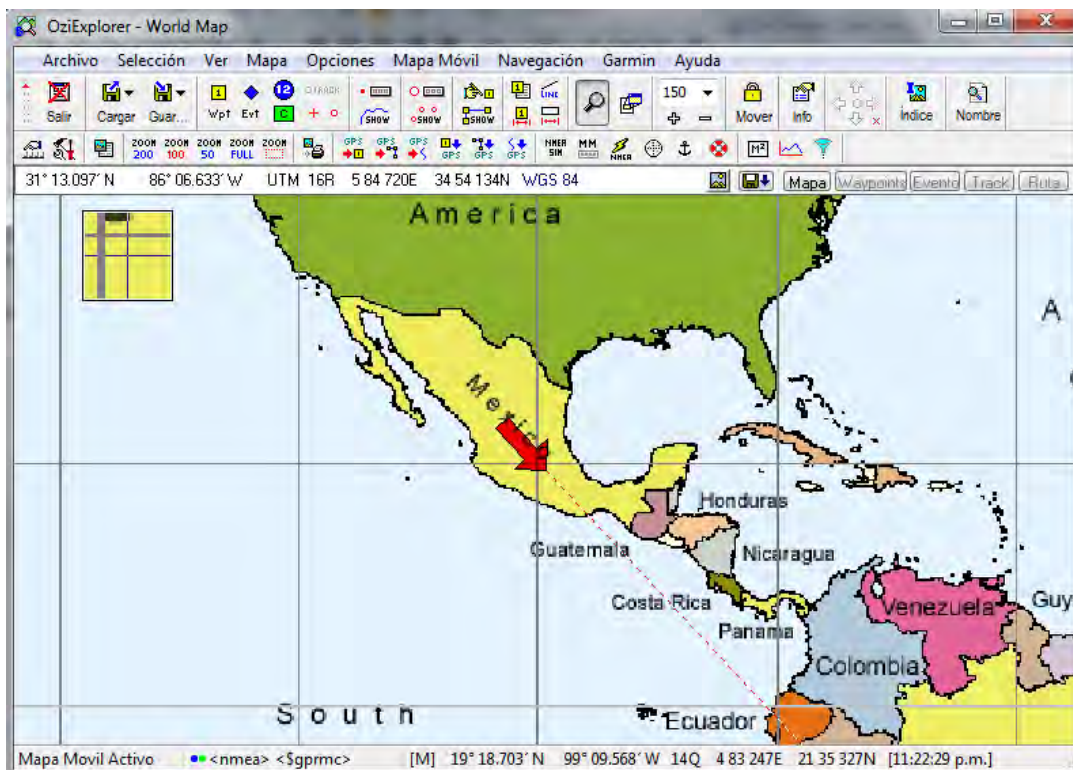


Figura 4.2. Conexión establecida entre OziExplorer y GPS.

Permite crear elementos permanentes en el mapa como lo son símbolos o comentarios. Es posible imprimir mapas, listas de referencias, rutas, pistas, etc. También se pueden crear perfiles de altura y velocidad de las pistas y tiene asociados potentes motores de búsqueda de mapas. Es compatible con el sistema operativo Windows (95, 98, ME, NT4, 2000, XP, Vista y 7).

#### 4.1.2.2 Desventajas

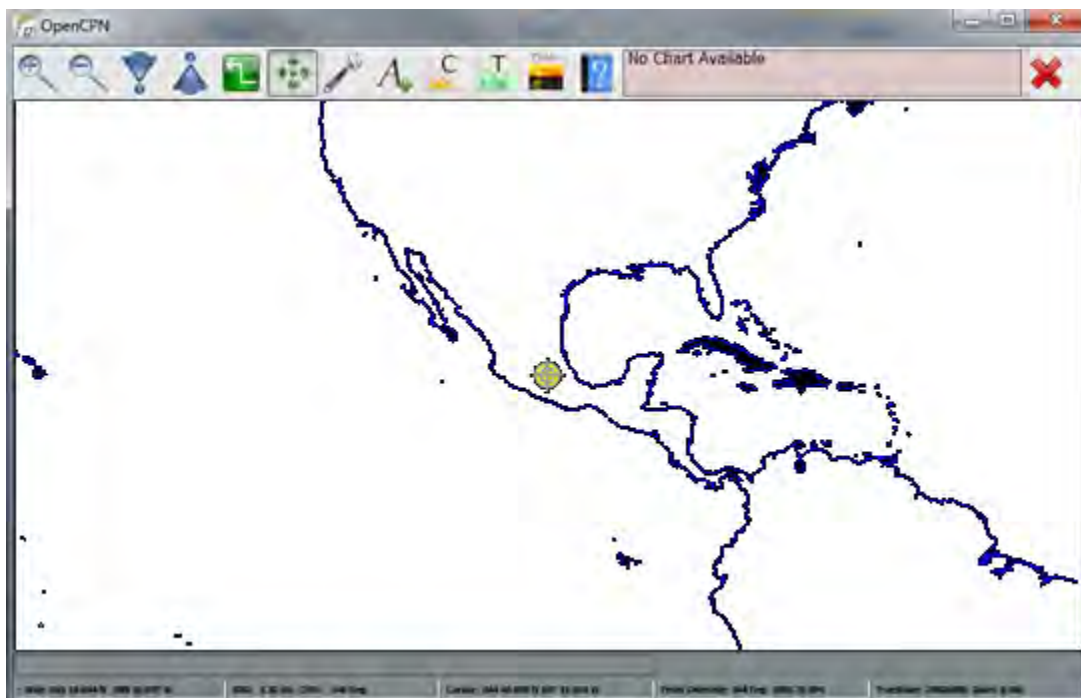
Para desbloquear todas las características mencionadas anteriormente es necesario registrar el producto y éste tiene un costo de 69 euros. Cabe resaltar que el programa sin registrar permite comprobar la comunicación con el GPS para corroborar que el software sea compatible con el GPS que posee el usuario.

#### 4.1.3 OpenCPN

Como se había mencionado en el capítulo 2, OpenCPN (Chart Plotter Navigator) es un programa para navegación y cartografía náutica. Está diseñado para ser utilizado en el puesto de mando de un barco, permitiendo al operador seguir fácilmente su posición sobre imágenes cartográficas. Además, OpenCPN puede mostrar las predicciones de mareas y corrientes.

##### 4.1.3.1 Ventajas

OpenCPN es una aplicación de código abierto, por lo que, entre otras cosas, se puede descargar y usar sin problemas. Además es un software conciso ya que no contiene procesos superfluos que consuman recursos haciendo la aplicación ligera y rápida tanto al iniciar como al interactuar con ella.



**Figura 4.3.** OpenCPN con la posición obtenida de GPS.

## Implantación

Es una aplicación que se ajusta a las especificaciones internacionales S52 de la IMO (Organización Marítima Internacional) para cartas de navegación electrónicas. Estas especificaciones son un estándar a nivel mundial que permite la portabilidad de datos y tratan de reducir el desorden e información extraña.

### 4.1.3.2 Desventajas

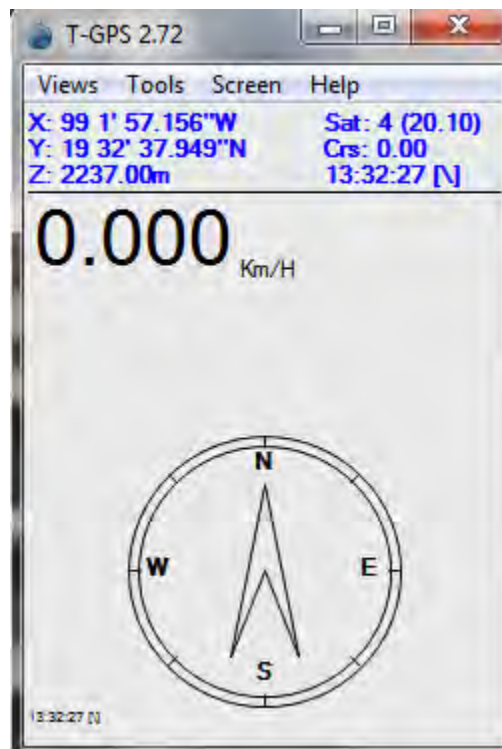
Actualmente, OpenCPN está en una fase de desarrollo por lo que es propenso a presentar problemas de programación y/o fallas.

### 4.1.4 Turbo GPS

Es una aplicación muy flexible tanto para PC como para Pocket PC. Es un gestor flexible y todo en uno para el GPS y está disponible para arquitecturas x86, x64, Pocket PC 2003 y Windows Mobile 5 y 6.

#### 4.1.4.1 Ventajas

Turbo GPS es una aplicación gratuita. Al conectar con el dispositivo GPS permite tener diferentes tipos de información, como lo son coordenadas, orientación geográfica (brújula), mapas en 2D y 3D, ubicación de satélites, permite crear pistas y rutas. Posee un simulador de vuelo y un simulador de manejo. Permite medir velocidad (en km/h, Mph, knot y m/s) y/o aceleración.



**Figura 4.4.** Pantalla principal de T-GPS una vez establecida la conexión con el GPS

Además, proporciona información sobre la posición de la luna y el sol. Maneja estadísticas de todas sus características. Posee una opción llamada "Sport" en la que se mide velocidad, distancia, calorías y tiempo, entre otras cosas. La conexión con el dispositivo es muy sencilla e intuitiva. Es una aplicación que se actualiza automáticamente.

#### 4.1.4.2 Desventajas

A pesar de ser una aplicación muy completa, sólo está disponible para PC's (x86, x64, Pocket PC 2003 y Windows Mobile 5 y 6), es decir, arquitecturas físicas robustas, por lo que no se puede ejecutar desde un teléfono celular convencional.

T-GPS posee una opción para mostrar la ubicación actual desplegada en Google (Google live position), esta característica se puede realizar conectándose a internet, lo que en un celular implicaría un costo extra.

#### 4.1.5 GPSToday

GPSToday es una aplicación que transforma un dispositivo con sistema operativo Windows Mobile en una poderosa herramienta de localización.

##### 4.1.5.1 Ventajas

GPSToday permite localizar direcciones con informe de tránsito y guarda el historial de localizaciones en línea para ser reutilizado o compartido. Es fácil compartir puntos de interés, imágenes y mensajes. Posee lo que llaman GeoAlertas, que informan cuando usuarios específicos están cerca o se alejan. Permite ver actualización de clima.



Figura 4.5. Dirección mostrada en el GPSToday.

## Implantación

Esta aplicación posee un algoritmo inteligente que mantiene y muestra, constantemente, la posición exacta del usuario sin consumir demasiada batería. Es una aplicación gratuita.

### 4.1.5.2 Desventajas

La aplicación sólo se puede ejecutar en un dispositivo que posea el sistema operativo Windows Mobile. Generalmente, los dispositivos que poseen dicho sistema operativo son las PDA's, aunque hay dispositivos de celulares que ya lo soportan.

Otro aspecto necesario para explotar las posibilidades de este software es la conexión a internet. Si el dispositivo celular no posee conexión inalámbrica, disfrutar de los beneficios que proporciona GPSToday puede resultar costoso, debido a las cuotas de las compañías de telefonía celular.

### 4.1.6 Aplicaciones para Celular

Nokia es una empresa que ha avanzado mucho en cuanto a mapas y localización se refiere. Tiene varias aplicaciones entre las que mencionaremos dos: Nokia Maps y Nokia Sports Tracker.

#### 4.1.6.1 Nokia Maps

Nokia Maps es una aplicación gratuita que permite a los dispositivos móviles descargar mapas para localizar lugares de interés o, simplemente, la ruta hacia un sitio. De igual manera, permite compartir rutas de lugares específicos. Además, ofrece un sistema de navegación completo y guías de ciudades con información muy útil. Esta aplicación permite conectar un GPS vía Bluetooth. Sin embargo, para obtener toda la información (acercamientos) hay que conectarse a internet, para ello se necesita o una conexión a internet desde el teléfono (en caso de que éste posea tarjeta inalámbrica) o usar la red de telefonía móvil. Además, hay que poseer un teléfono Nokia, porque las aplicaciones están diseñadas para su sistema operativo.

#### 4.1.6.2 Nokia Sport Tracker

Nokia Sport Tracker es una herramienta de monitorización GPS que proporciona información como posición, velocidad, altura, distancia y tiempo que se guardan automáticamente en el teléfono celular, así como las rutas asociadas. La desventaja es que sólo se puede instalar en dispositivos Nokia habilitados con GPS.

#### 4.1.6.3 GPS Track

GPS Track es una aplicación gratuita diseñada para conectar cualquier dispositivo celular que soporte J2ME y el API para Bluetooth de Java, con un dispositivo GPS con Bluetooth. Una vez establecida la conexión, GPS Track graba la ruta del viaje realizado para, posteriormente, transmitir las por cualquier medio y poder leer la información desde Google Maps o Google Earth. El problema con esta aplicación es que sólo graba las rutas, no es posible ver la ubicación desplegada en un mapa en tiempo real.



A continuación se presenta una tabla con las aplicaciones mencionadas en este capítulo para tener un panorama más amplio de ellas.

|   | <b>Fabricante</b>    | <b>Software</b>                                       | <b>Licencia</b> | <b>Dispositivos GPS que soportan</b>                  | <b>Conexión entre SW y GPS</b> |
|---|----------------------|---|-----------------|---|--------------------------------|
| 1 | GPS con Google Earth | Google Earth Plus, Google Earth Pro y Google Earth EC | US 400          | Garmin y Magellan                                     | Puerto Serial y USB            |
| 2 | OziExplorer          | OziExplorer, OziExplorer3D y OziExplorerCE            | € 69            | Magellan, Garmin, Lowrance/Eagle, Brunton/Silva y MLR | Puerto Serial, USB y Bluetooth |
| 3 | OpenCPN              | OpenCPN   | Libre           | Varios  | Puerto Serial, USB y Bluetooth |
| 4 | Turbo IRC            | Turbo GPS   | Gratuita        | Varios  | Puerto Serial, USB y Bluetooth |
| 5 | Geo Terrestrial      | GPSToday  | Gratuita        | Ninguno   | Ninguna                        |
| 6 | Nokia                | Nokia Maps  | Gratuita        | Ninguno   | Ninguna                        |
| 7 | Nokia                | Nokia Sport Tracker                                   | Gratuita        | GPS integrado   | Interna                        |
| 8 | Qcontinuum Freeware  | GPS Tracker   | Libre           | Varios  | Bluetooth                      |

**Tabla 4.1.** Tabla comparativa de aplicaciones que leen datos de dispositivos GPS

## 4.2 Ejecución de la aplicación

En este tema se van a realizar pruebas con algunas de las aplicaciones mencionadas en el tema anterior, con el fin de entender el funcionamiento básico de la aplicación que se quiere realizar y la ventaja de realizarlo para un dispositivo móvil.

Antes de poder ejecutar alguna aplicación vista en el tema anterior, es necesario saber el tipo de conexión que acepta (por cable, Bluetooth o ambas) y verificar que ésta sea exitosa. Para ello se siguen los pasos vistos en el Capítulo 2, tema 2.4 y, una vez establecida la comunicación, ya es posible realizar las instrucciones que se muestran a continuación para realizar la conexión.

### 4.2.1 Ejecución de OziExplorer



Figura 4.6. Ejecución de OziExplorer

Si la aplicación no está registrada aparecerá una ventana que para ingresar un número de serie y un código:



Figura 4.7. Registro de OziExplorer.

Se presiona el botón “Continua” para abrir la aplicación, estarán deshabilitadas algunas funciones, pero permite realizar la conexión con el GPS.

Si es la primera vez que se abre la aplicación, muestra una ventana emergente con las Condiciones de Uso. Se presiona el botón “OK” y se abre otra ventana emergente con Ayuda sobre el software, presionamos “OK” otra vez para abrir la aplicación.

En el menú “Archivo” se selecciona “Cargar Archivos” y después se presiona la opción “Cargar Archivo de Mapa”, se abre un explorador de archivos para elegir el mapa que se va a cargar.

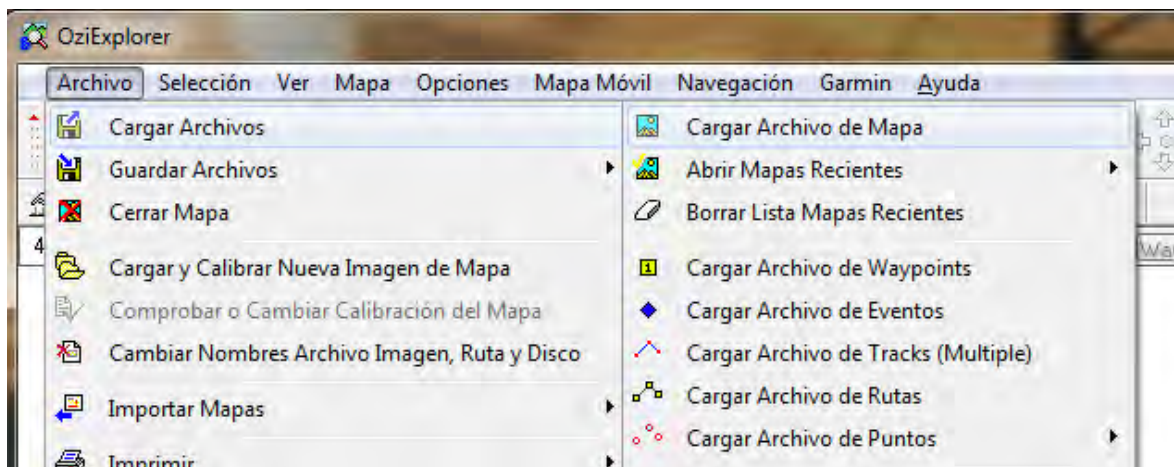


Figura 4.8. Cargar mapa en OziExplorer.

OziExplorer trae dos mapas por default para la versión de prueba, se selecciona el que dice mapa “World.map” y se presiona el botón “abrir”. Se despliega un mapamundi en la aplicación.

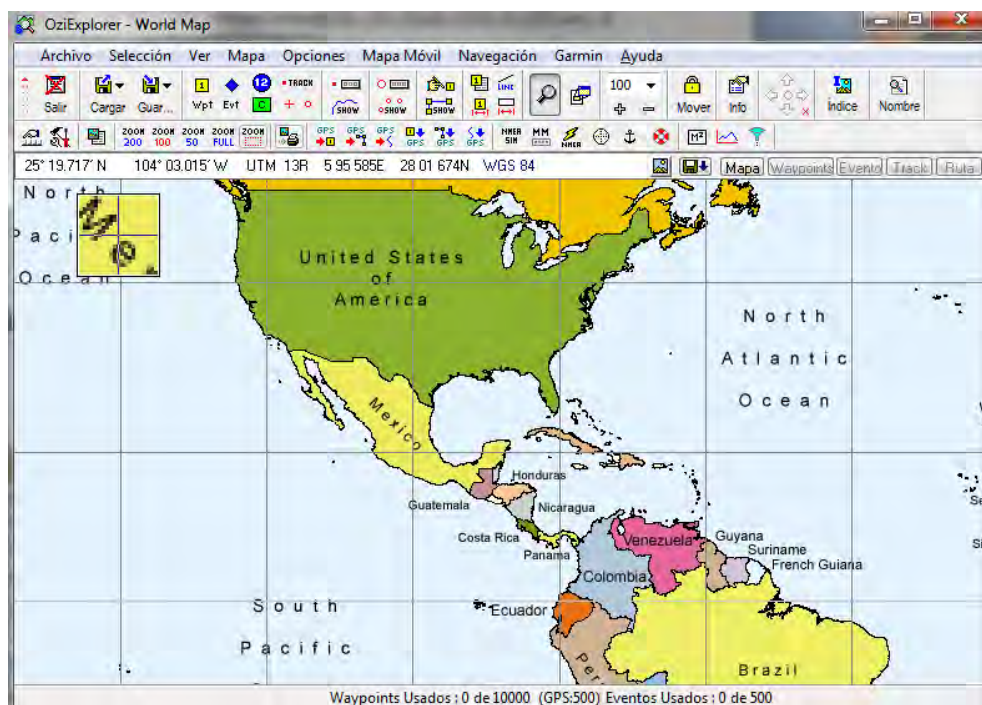
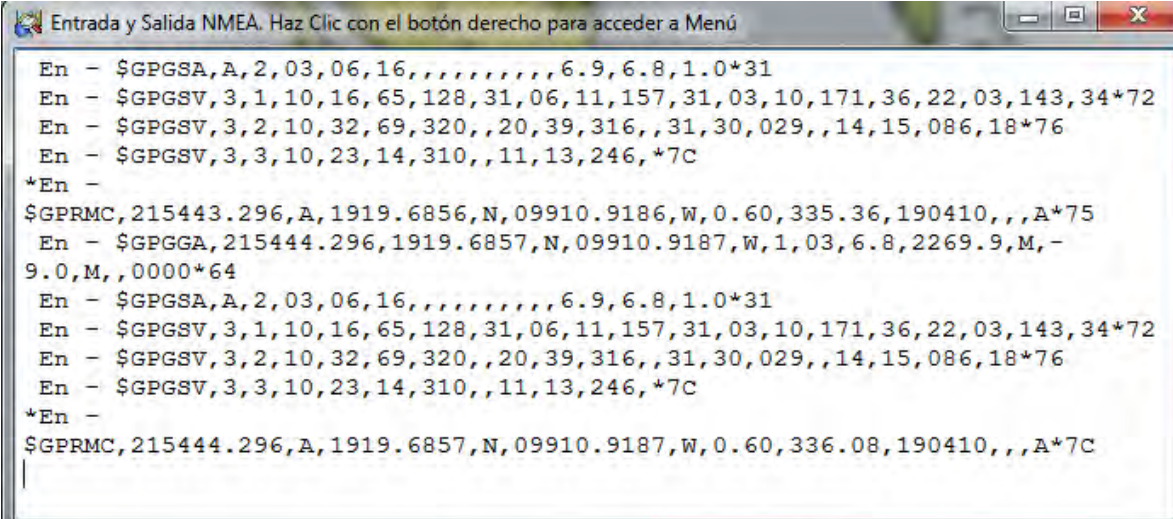


Figura 4.9. Mapa de demostración en OziExplorer

## Implantación

Una vez abierto el mapa se selecciona el menú “Mapa Móvil” y se elige la opción “Iniciar Comunicación NMEA con el GPS”. Si no marca ningún error significa que la comunicación fue exitosa y, en el menú “Mapa Móvil”, se desactiva la opción “Iniciar Comunicación NMEA con el GPS” y se activa la opción “Detener Comunicación NMEA con el GPS”. Si se presiona, en el mismo menú, la opción “Mostrar Entrada y Salida de información NMEA”, se puede observar la información enviada por el GPS.



```
Entrada y Salida NMEA. Haz Clic con el botón derecho para acceder a Menú
En - $GPGSA,A,2,03,06,16,,,,,,,,,6.9,6.8,1.0*31
En - $GPGSV,3,1,10,16,65,128,31,06,11,157,31,03,10,171,36,22,03,143,34*72
En - $GPGSV,3,2,10,32,69,320,,20,39,316,,31,30,029,,14,15,086,18*76
En - $GPGSV,3,3,10,23,14,310,,11,13,246,*7C
*En -
$GPRMC,215443.296,A,1919.6856,N,09910.9186,W,0.60,335.36,190410,,,A*75
En - $GPGGA,215444.296,1919.6857,N,09910.9187,W,1,03,6.8,2269.9,M,-
9.0,M,,0000*64
En - $GPGSA,A,2,03,06,16,,,,,,,,,6.9,6.8,1.0*31
En - $GPGSV,3,1,10,16,65,128,31,06,11,157,31,03,10,171,36,22,03,143,34*72
En - $GPGSV,3,2,10,32,69,320,,20,39,316,,31,30,029,,14,15,086,18*76
En - $GPGSV,3,3,10,23,14,310,,11,13,246,*7C
*En -
$GPRMC,215444.296,A,1919.6857,N,09910.9187,W,0.60,336.08,190410,,,A*7C
```

Figura 4.10. Lectura de datos NMEA en OziExplorer.

De igual manera, ahora que la conexión está establecida, se puede observar que OziExplorer nos indica la posición en la que estamos.

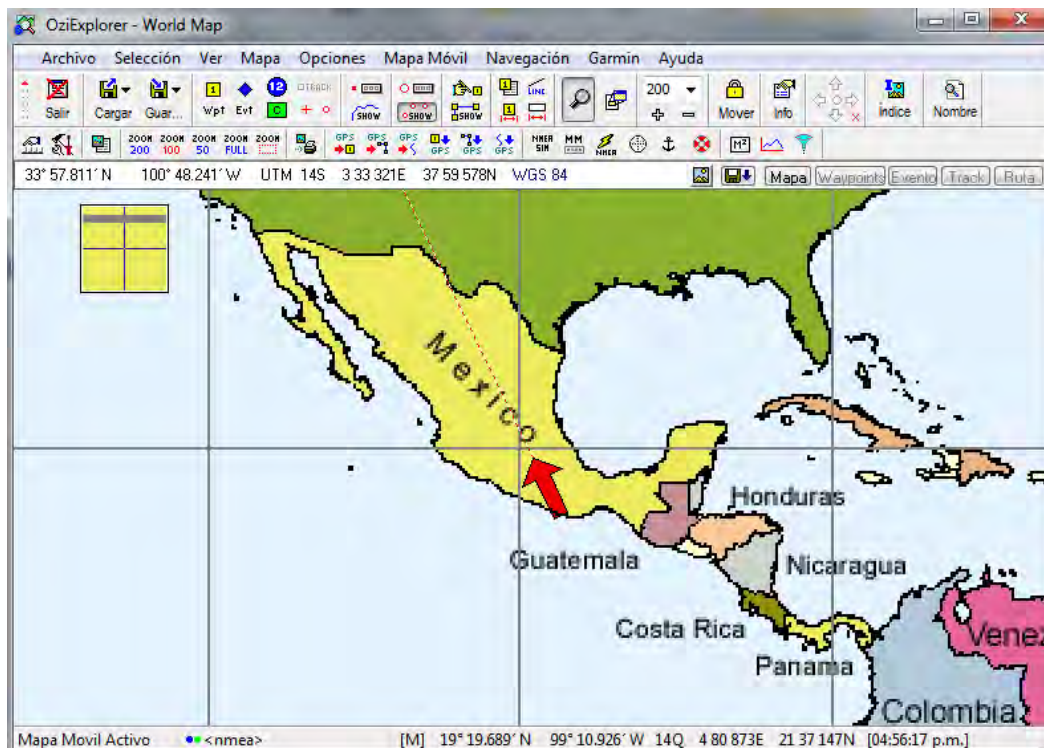


Figura 4.11. Ubicación de la posición obtenida de datos NMEA en OziExplorer.

Lamentablemente, no es mucho lo que se puede hacer con esta versión de OziExplorer ya que es una versión de prueba, pero cumplió con la función de conectar con el GPS a pesar de no estar registrado.

#### 4.2.2 Ejecución de OpenCPN

Si es la primera vez que se abre OpenCPN, el programa manda al menú Herramientas (ToolBox). En este menú, se selecciona la pestaña que dice “GPS” y se configura el puerto por el cual se está escuchando el dispositivo GPS. Si se quiere verificar que la conexión se lleva a cabo de manera exitosa, se selecciona la casilla que dice “Show GPS/NMEA data stream window” para mostrar los datos NMEA leídos del GPS.

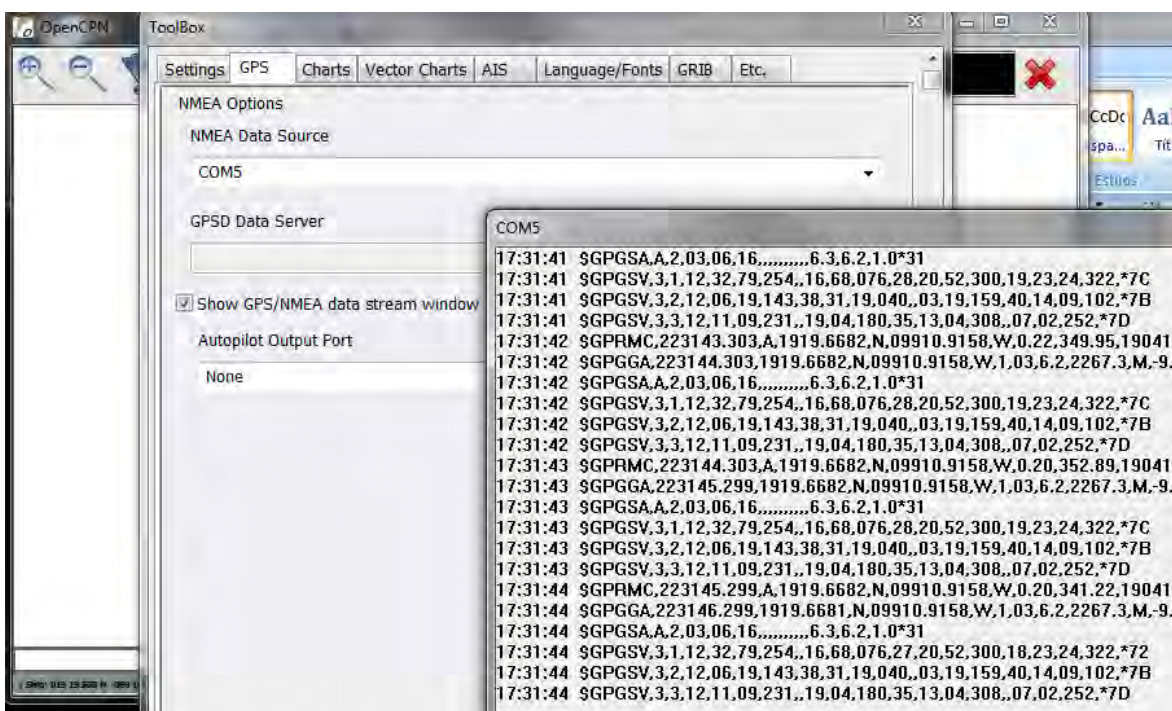
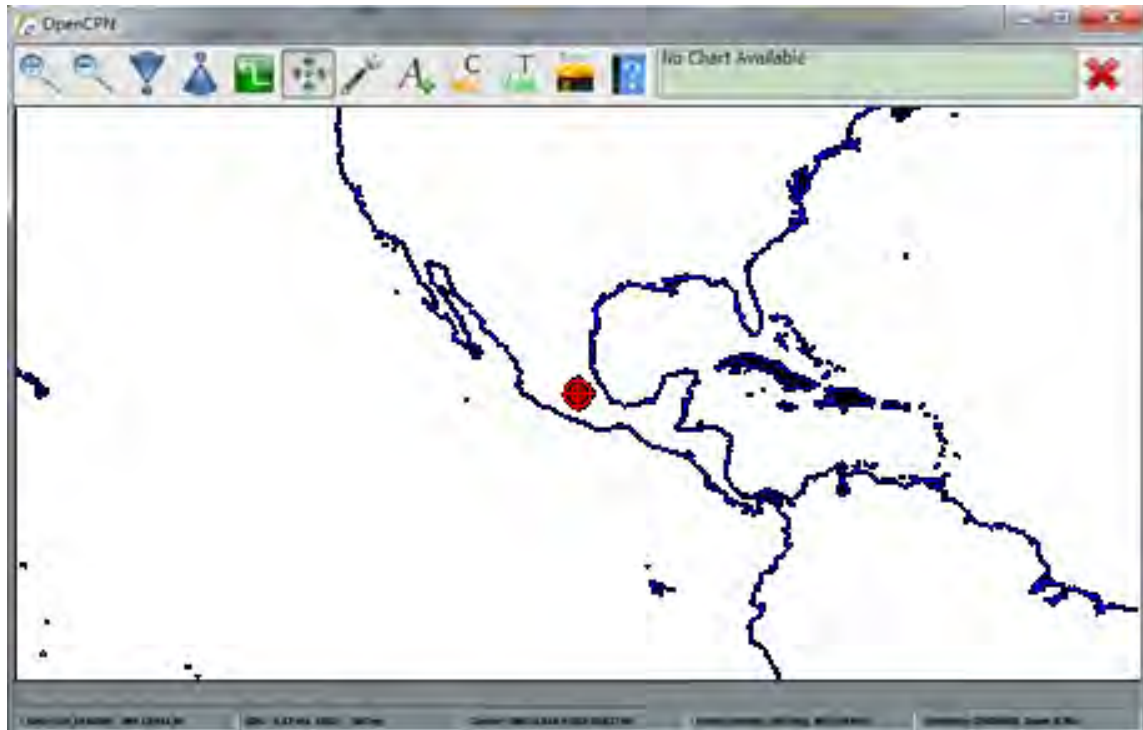


Figura 4.12. Configurar OpenCPN.

Si la conexión se llevó a cabo de manera exitosa, en la pantalla principal de Open CPN se mostrará un mapa de América muestra la ubicación actual.

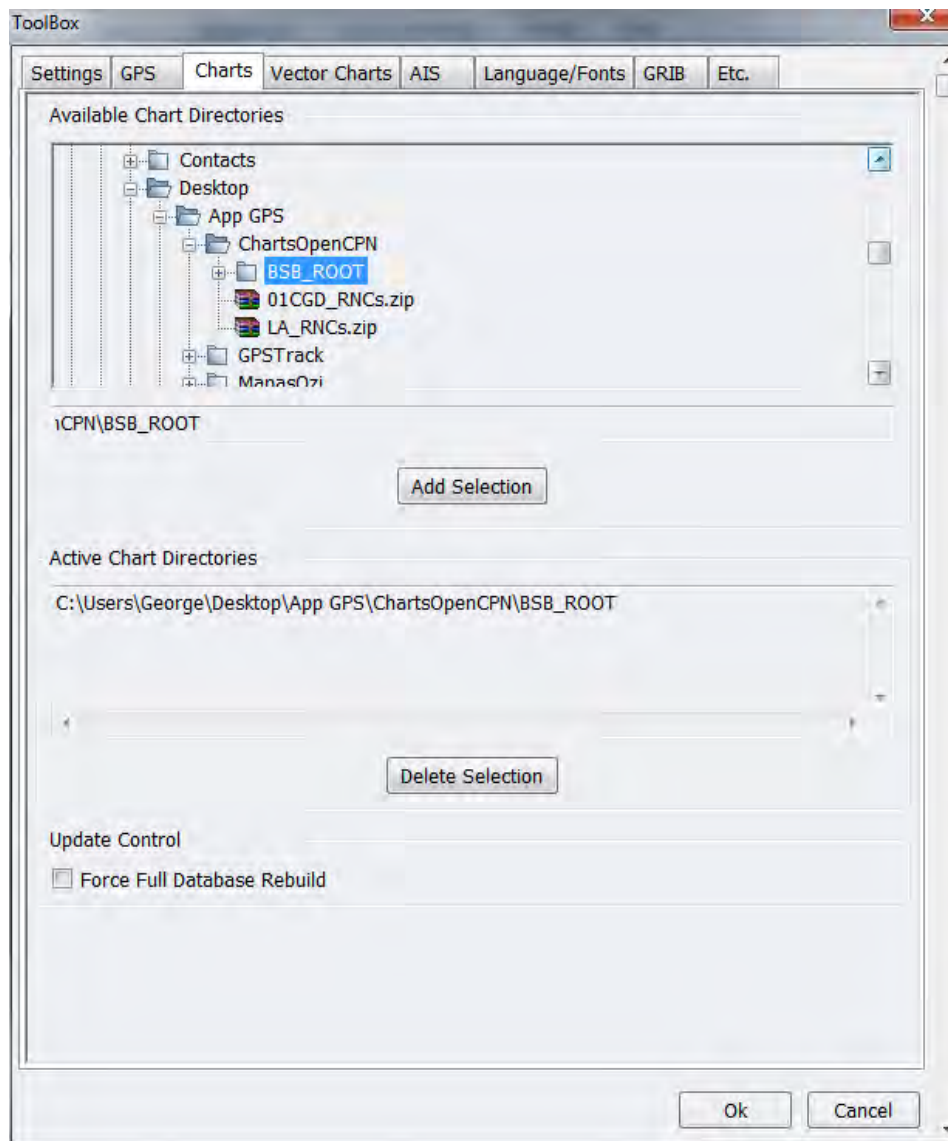
## Implantación



**Figura 4.13.** Ubicación obtenida del GPS en OpenCPN.

OpenCPN, como ya se ha mencionado, es una aplicación dedicada a la navegación y a la cartografía náutica, por ello posee cartas de navegación (Chart) de distintas partes del mundo. Estas cartas de navegación se pueden crear o se pueden descargar de la red.

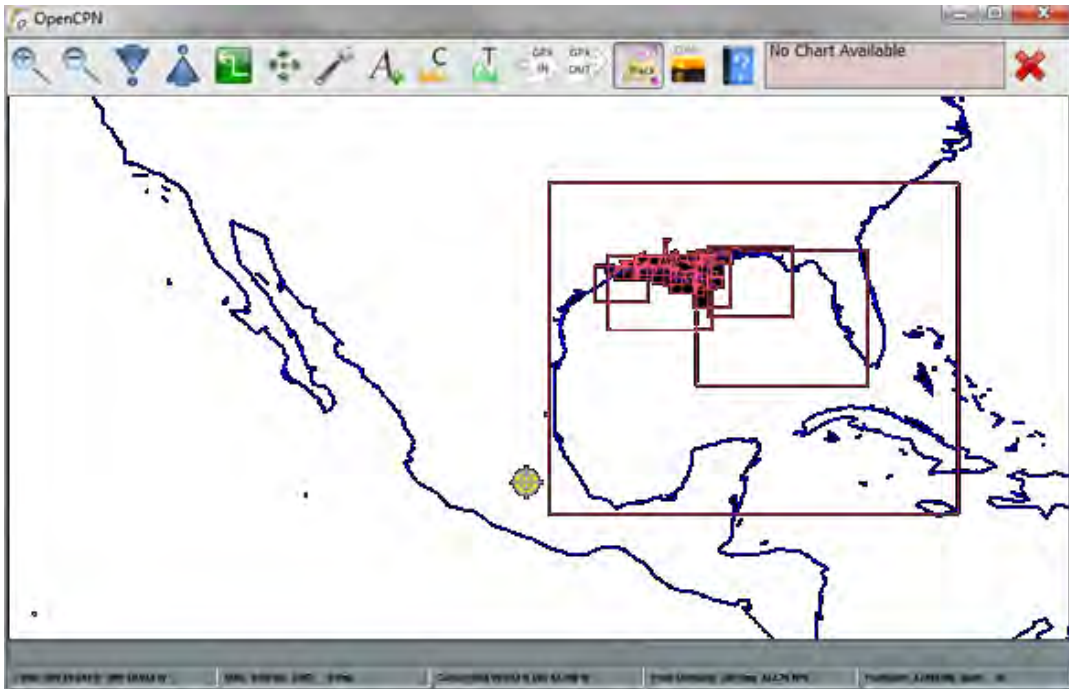
Para cargar las cartas de navegación se presiona el menú "Herramientas" (Tools) y en la pestaña "Cartas de Navegación" (Charts) se elige el directorio donde se encuentra las cartas que se van a cargar, se oprime el botón "Agregar Selección" (Add Selection) y se presiona "OK".



**Figura 4.14.** Cargando Cartas de Navegación en OpenCPN.

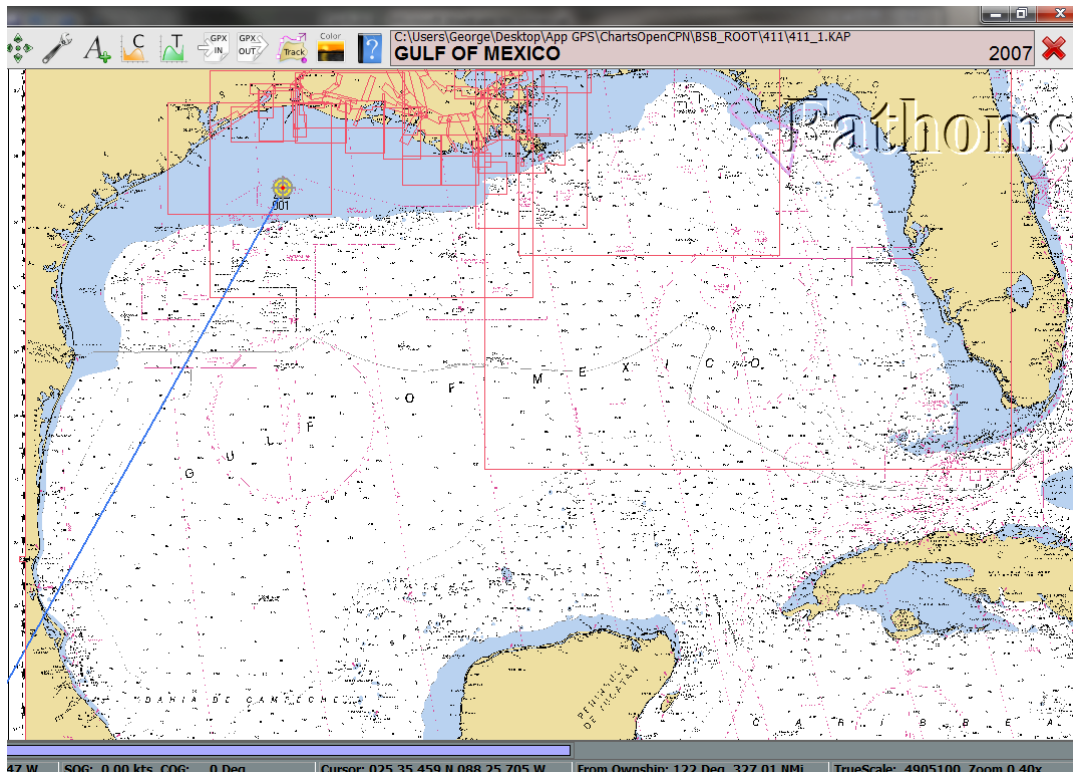
En la figura 4.15, se puede observar que la aplicación ha dibujado unos cuadrados en el mapa, eso significa que las cartas cargadas pertenecen a esa zona geográfica. En este caso, se descargaron cartas de navegación de la costa Este de Estados Unidos.

## Implantación



**Figura 4.15.** Cartas de navegación cargadas y desplegadas en el mapa en OpenCPN.

Las cartas de navegación proporcionan más información sobre la zona en la que se está “navegando”.



**Figura 4.16.** Acercamiento de las Cartas de Navegación cargadas en OpenCPN.



Además, se pueden crear rutas, pistas y/o ver información más detallada de la zona, hacer mediciones, revisar rutas ya hechas, entre otras cosas, únicamente apretando el botón secundario del mouse y seleccionando la opción deseada. En la figura 4.16 se observa un círculo amarillo en el Golfo de México y una línea que se pierde en la República Mexicana, ese es un ejemplo de una ruta trazada de nuestra posición actual (Figura 4.13) a una posición arbitraria en el Golfo de México.

#### 4.2.3 Ejecución de Turbo GPS

Una vez abierta la aplicación, en el menú “Herramientas” (Tools) se selecciona la opción “GPS” y se presiona “Configuraciones del GPS” (GPS Setup). En esa pantalla se puede indicar el puerto COM por el que se va a escuchar el GPS y la velocidad de transferencia (Baudios), entre otras cosas.

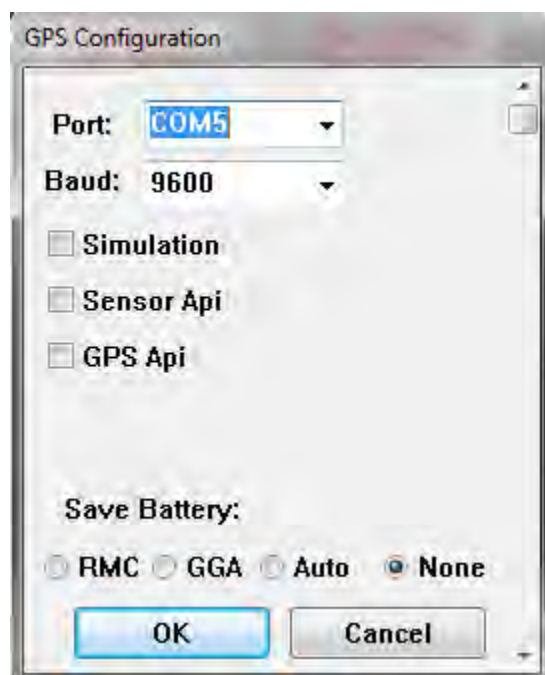


Figura 4.14. Configuración de parámetros en Turbo GPS

Una vez configurados los parámetros de conexión, se presiona el botón “OK”, se vuelve a ingresar al menú “Herramientas” (Tools), se selecciona la opción “GPS” y se presiona la opción “Conexión” (Connect). Si la conexión fue exitosa se escucha un sonido (como un beep), de lo contrario aparece un mensaje de error. Para verificar que la aplicación se está comunicando con el dispositivo GPS, hay que dirigirse a las “Vistas” de la aplicación (Views) y se selecciona la “Bitácora de la Conexión” (Raw Log), ahí se pueden observar las tramas NMEA que reciben del GPS.

## Implantación

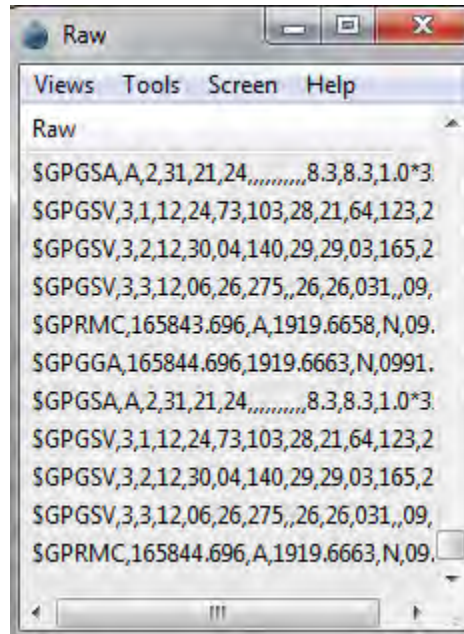


Figura 4.15. Información recibida por Turbo GPS.

Cómo ya se había mencionado, esta aplicación contiene diversas ventanas e información, a continuación se muestran algunas.

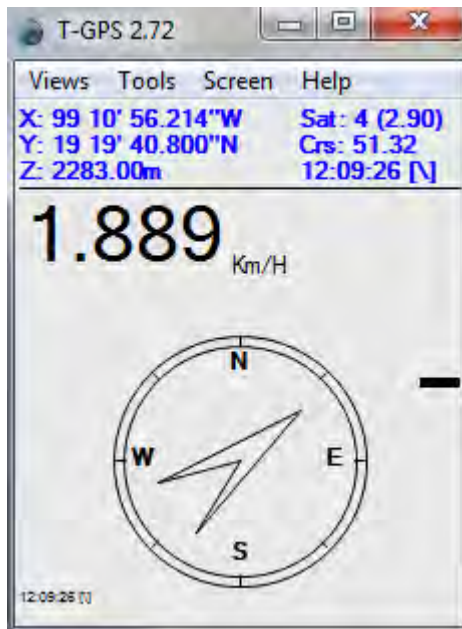


Figura 4.16. Información del GPS

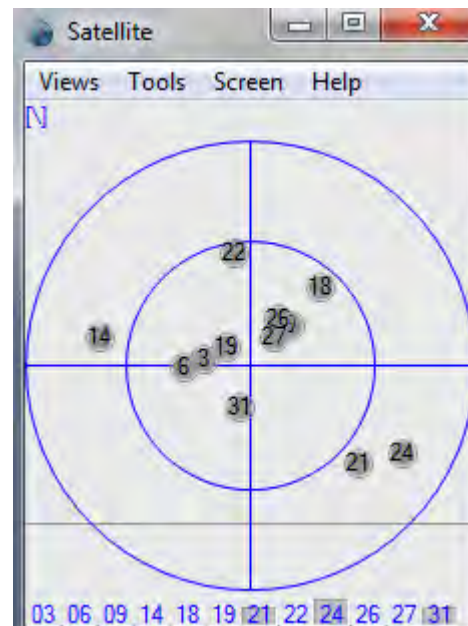


Figura 4.17. Vista de Satélites

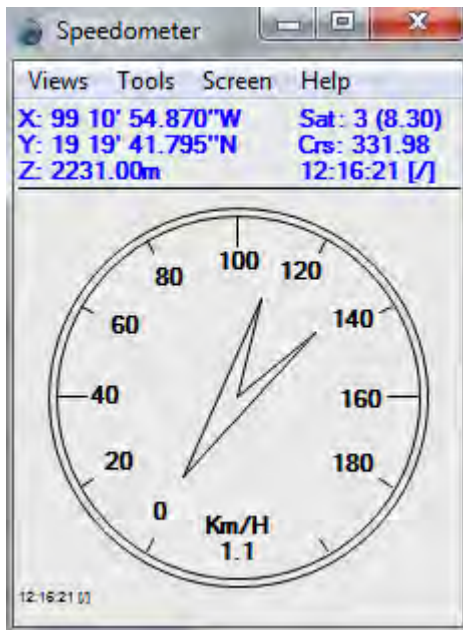


Figura 4.18. Velocímetro

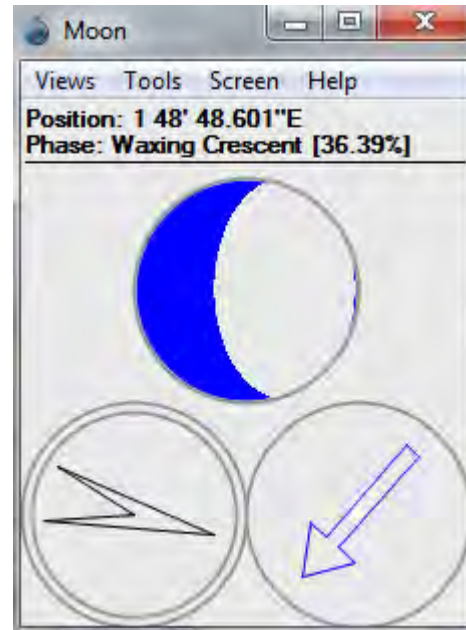


Figura 4.19. Posición de la Luna

Si se posee conexión a internet, esta aplicación permite conectarse con Google Live para mostrar la posición actual en un mapa, para ello se selecciona el menú “Herramientas” (Tools), se selecciona el submenú “Características” (Features) y se presiona Google Live Position. Se abre otra ventana donde aparece un mapamundi y está señalada la ubicación actual.

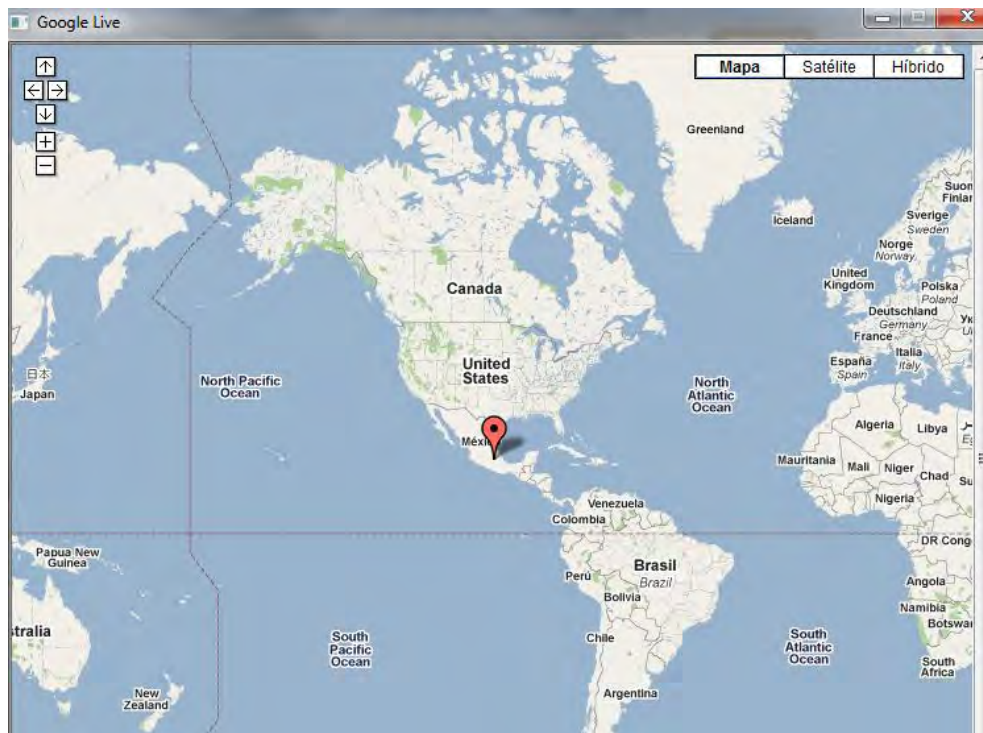
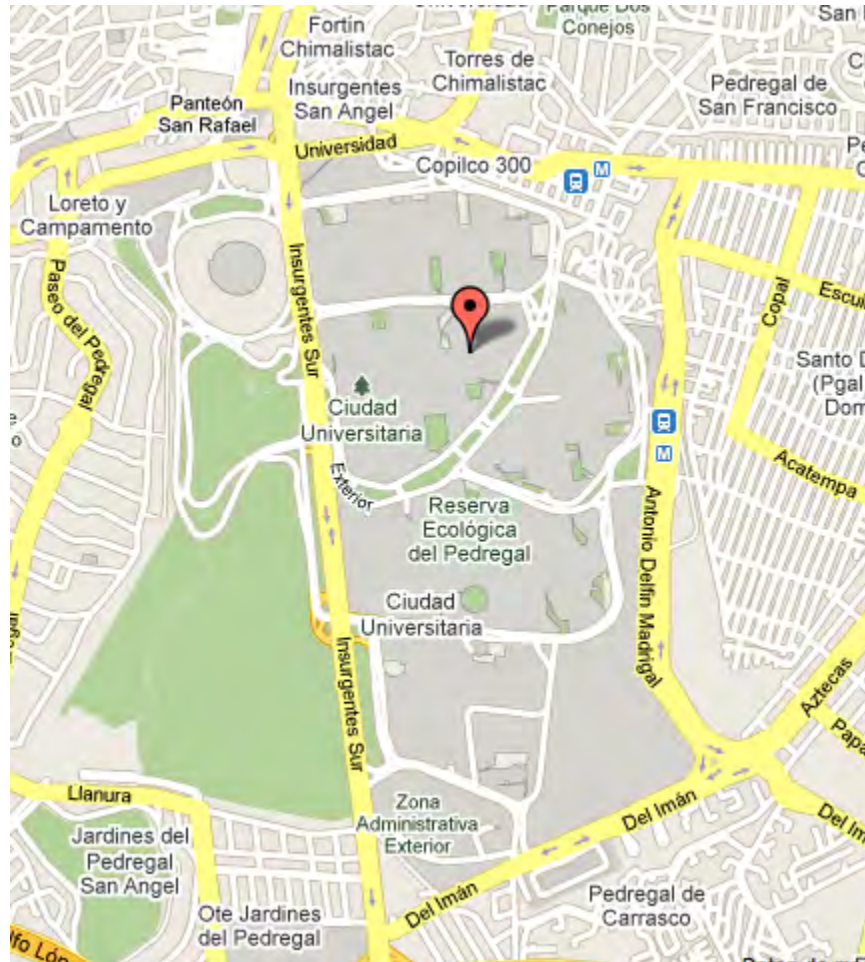


Figura 4.20. Posición mostrada en Google Live.

## Implantación



**Figura 4.21.** Acercamiento de la posición mostrada en Google Live

### 4.2.4 Ejecución de Nokia Maps

Para ejecutar la aplicación Nokia Maps es necesario abrir el “Menú” del celular, seleccionar “Aplicaciones” y presionar el ícono “Mapas”. Aquí se da la opción de “Activar el uso de red”, si se activa es posible visualizar hasta la calle en la posición actual, sin embargo, esto representa un costo debido a que la consulta la hace por la red del teléfono móvil. Si no se activa al iniciar el programa (se puede realizar después) sólo muestra la ubicación en un mapamundi.

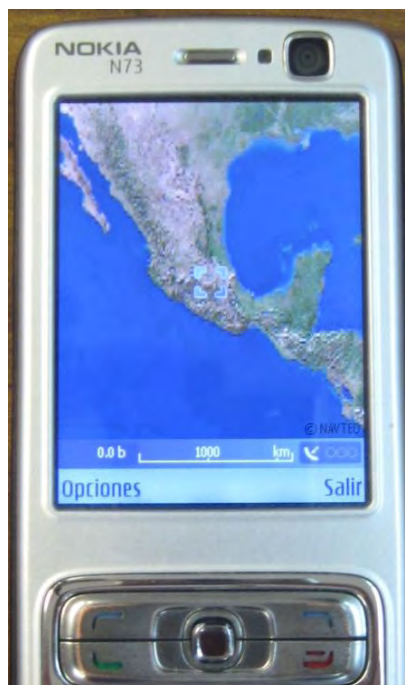


**Figura 4.22.** Ícono “Maps”



**Figura 4.23.** Aplicación “Nokia Maps”

Una vez abierta la aplicación, se selecciona el menú “Opciones” y se presiona “Buscar Lugar”. Este submenú permite elegir entre varias opciones, se escoge “Posición GPS [0]”. En ese momento Nokia Maps se intenta conectar al GPS, si el Bluetooth del dispositivo no está activado, nos da la posibilidad de hacerlo. Una vez que Nokia Maps haya establecido conexión con el dispositivo GPS muestra la posición en la pantalla.



**Figura 4.24.** Posición desplegada por Nokia Maps

## Implantación

Sin embargo, cuando se intenta acercar la imagen para ver algo más que la imagen de la República Mexicana, Nokia Maps no muestra nada, ya que sólo posee imágenes globales cargadas. Para ver estos detalles es necesario activar la red del teléfono teniendo en cuenta los costos que la compañía telefónica imponga (ya sea por minuto, por kilobyte, etc.).

# Capítulo 5. Pruebas y resultados

Con base en todo lo visto en los capítulos anteriores, se va a crear un análisis sencillo para realizar una aplicación que se comuniquen con un dispositivo GPS. Lo primero que hay que analizar es la manera en que el usuario va a interactuar con su teléfono celular, es decir, el ambiente que va a utilizar el dispositivo móvil. Después hay que visualizar la conexión entre el dispositivo móvil y el GPS, aquí es donde se justifica el uso del lenguaje de programación Java, ya que posee métodos que ayudarán a realizar esta tarea de conexión. Por último, hay que conocer la manera en que se manejarán las coordenadas recibidas y los mapas.

La idea es descargar sobre la aplicación el número de mapas que sean útiles, es decir, si el usuario va a ir a acampar, por ejemplo, se sabe qué zona se va a visitar, se puede verificar el radio de acción en el que se desenvolverá y entonces descargar los mapas de la zona que se va a visitar para saber la ruta a seguir o poder regresar a una zona urbana en caso de algún inconveniente. Entonces, ¿cuál es la finalidad de la aplicación?, que sea útil en situaciones necesarias y, de alguna manera, predeterminadas. Si se quisiera consultar todos los rincones del mundo pues conviene más utilizar un navegador o alguna aplicación como Google Earth. Empero, lo que se planea es realizar una aplicación que pueda fungir como guía o apoyo, no sólo para diversión, por sus características y debido al dispositivo al que está enfocado, el celular.

Es importante tener en cuenta que se pretende realizar una aplicación libre, esto con el fin de que el código se enriquezca con aportaciones voluntarias para así formar una aplicación robusta, estable y útil a la comunidad. La mayor ventaja de realizar una aplicación GNU (General Public License) es poder mejorar constantemente el software compartiendo conocimiento e invitando a la sociedad a mejorar las propuestas hechas en la aplicación, para así aprender y/o retroalimentarse todos de todos.

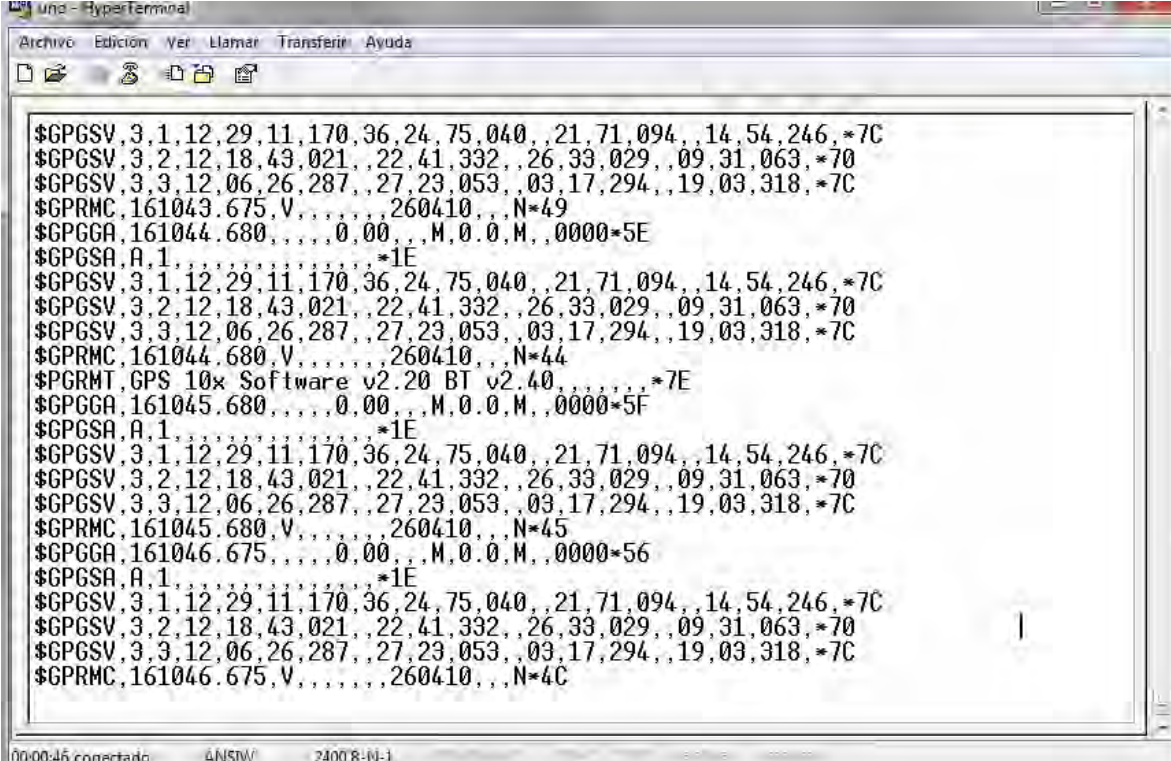
## 5.1 Interacción con la aplicación y el dispositivo GPS

Lo primero a realizar es un bosquejo de la secuencia de pantallas de la aplicación para contar con una idea de cómo empezar a abordar la solución al problema. Aquí es donde entra el esquema de una arquitectura Cliente/Servidor y, de igual manera, es aquí donde toma importancia la arquitectura del programa hecha previamente.

De inicio, se plantea que la aplicación obtenga la posición desde un dispositivo GPS y que muestre en un mapa la ubicación actual. Entonces, la primera opción va a ser “Mostrar Ubicación”. De igual manera debe existir una pantalla informativa para que el usuario esté enterado de lo que está haciendo internamente la aplicación. Esta pantalla será “Ayuda” y contendrá información relevante para que el usuario entienda el funcionamiento de la aplicación de manera clara, es decir, de una manera amigable y útil. La última opción debe cerrar la aplicación y se llamará “Salir”.

## Pruebas y resultados

El botón “Mostrar Ubicación” es el principal en la aplicación. Cuando se presiona este botón se siguen varios procesos. El primero es localizar los dispositivos Bluetooth cercanos. Si no se encuentra ningún dispositivo se manda un mensaje a pantalla. Si se encuentra algún dispositivo, el usuario seleccionará el correspondiente al dispositivo GPS. Una vez seleccionado el dispositivo, la aplicación debe “escuchar” lo que éste envía para poder obtener las coordenadas. Estas coordenadas las envía el dispositivo GPS como se muestra en la figura 5.1.



```
uno - Hyperterminal
Archivo Edición Ver Llamar Transferir Ayuda
$GPGSV,3,1,12,29,11,170,36,24,75,040,,21,71,094,,14,54,246,*7C
$GPGSV,3,2,12,18,43,021,,22,41,332,,26,33,029,,09,31,063,*70
$GPGSV,3,3,12,06,26,287,,27,23,053,,03,17,294,,19,03,318,*7C
$GPRMC,161043.675,V,,,,,260410,,N*49
$GPGGA,161044.680,,,,,0,00,,M,0.0,M,0000*5E
$GPGSA,A,1,,,,,,*1E
$GPGSV,3,1,12,29,11,170,36,24,75,040,,21,71,094,,14,54,246,*7C
$GPGSV,3,2,12,18,43,021,,22,41,332,,26,33,029,,09,31,063,*70
$GPGSV,3,3,12,06,26,287,,27,23,053,,03,17,294,,19,03,318,*7C
$GPRMC,161044.680,V,,,,,260410,,N*44
$PGRMT,GPS 10x Software v2.20 BT v2.40,,,,,*7E
$GPGGA,161045.680,,,,,0,00,,M,0.0,M,0000*5F
$GPGSA,A,1,,,,,,*1E
$GPGSV,3,1,12,29,11,170,36,24,75,040,,21,71,094,,14,54,246,*7C
$GPGSV,3,2,12,18,43,021,,22,41,332,,26,33,029,,09,31,063,*70
$GPGSV,3,3,12,06,26,287,,27,23,053,,03,17,294,,19,03,318,*7C
$GPRMC,161045.680,V,,,,,260410,,N*45
$GPGGA,161046.675,,,,,0,00,,M,0.0,M,0000*56
$GPGSA,A,1,,,,,,*1E
$GPGSV,3,1,12,29,11,170,36,24,75,040,,21,71,094,,14,54,246,*7C
$GPGSV,3,2,12,18,43,021,,22,41,332,,26,33,029,,09,31,063,*70
$GPGSV,3,3,12,06,26,287,,27,23,053,,03,17,294,,19,03,318,*7C
$GPRMC,161046.675,V,,,,,260410,,N*4C
00:00:46 conectado 4NSIW 24008-N-1
```

Figura 5.1. Tramas enviadas por el GPS.

### 5.1.1 Análisis de tramas NMEA

El dispositivo GPS transmite la información en código ASCII. Cada trama comienza con el carácter “\$”, un identificador de dos dígitos llamado identificador de locutor “talker ID” (para los datos transmitidos desde un GPS el identificador es “GP”), un código identificador de 3 dígitos llamado identificador de sentencia “sentence ID” y, posteriormente, una serie de datos separados por comas que finalizan, generalmente, con el símbolo “\*” y un código de 2 caracteres. Cada sentencia puede contener hasta un máximo de 82 caracteres (incluyendo inicial y final). Si algunos de los datos de la serie que van separados por comas no están disponibles, se dejan vacíos.

Los dispositivos GPS generan sentencias NMEA referidas al Datum WGS-84, es decir, un sistema de coordenadas cartográficas mundial. Las sentencias NMEA típicas son GGA, GLL, GSA, GSV, RMC y VTG.



El dispositivo GPS envía información periódicamente (cada segundo). Una vez establecida la comunicación con el GPS se pueden capturar las tramas de información y analizarlas para obtener las coordenadas, para ello es importante leer la información que envía el GPS y saber los datos que ésta contienen.

```
$GPGGA,161537.678,1919.6727,N,09910.9121,W,1,03,3.7,2273.4,M,-9.0,M,,0000*6D
$GPGSA,A,2,29,21,31,,,,,,,,,3.8,3.7,1.0*37
$GPGSV,3,1,12,21,70,102,36,29,09,169,36,31,05,196,29,24,76,050,19*7B
$GPGSV,3,2,12,14,55,250,19,18,43,025,,22,42,335,,26,30,030,*7F
$GPGSV,3,3,12,09,30,060,,06,26,284,,27,22,051,,03,17,291,*74
$GPRMC,161537.678,A,1919.6727,N,09910.9121,W,0.31,89.78,260410,,,A*41
```

Es posible obtener las coordenadas de la posición actual de dos sentencias NMEA: GGA y RMC. Se va a analizar la trama RMC. Para ello hay que separar la trama y analizar.

RMC: Datos de Tránsito/GPS recomendados (Recommended minimum specific GPS/Transit data)  
 Formato: \$GPRMC,hhmmss.ss,A,lll.ll,a,yyyyy.yy,a,x.x,x.x,ddmmyy,x.x,a\*hh

Donde:

- hhmmss.ss: Hora UTC en la que fue tomado el dato (horas, minutos, segundos)
- A: Advertencia de navegación recibida (A = correcto, V = advertencia).
- lll.ll: Latitud (grados y minutos).
- a: Tipo de latitud (N = norte, S = sur).
- yyyyy.yy: Longitud (grados y minutos).
- a: Tipo de longitud (E = este, W = oeste).
- x.x: velocidad sobre la tierra en knots.
- x.x: Pista o curso cumplido en grados.
- ddmmyy: Fecha UT en la que fue tomado el dato (día, mes, año).
- x.x: Variación magnética oriental en grados
- a: Orientación de la variación magnética (E = este, W = oeste).
- \*hh: Fin de trama.

Ahora se analizará una trama recibida para comprender mejor la información enviada por el GPS.

```
$GPRMC,161537.678,A,1919.6727,N,09910.9121,W,0.31,89.78,260410,,,A*41
      1      2      3      4      5      6      7      8      9      10
```

|    |            |   |
|----|------------|---|
| 1  | 161537.678 | Arreglo de tiempo 16:15:37 UTC.                               |
| 2  | A          | Advertencia de navegación: A = correcto.                      |
| 3  | 1919.6727  | Latitud 19 grados, 19.6727 minutos.                           |
| 4  | N          | Norte.  |
| 5  | 09910.9121 | Longitud 99 grados, 10.9121 minutos.                          |
| 6  | W          | Oeste.  |
| 7  | 0.31       | Velocidad sobre la tierra 0.31 [knots] = 0.15947777764 [m/s]. |
| 8  | 89.78      | Curso cumplido en 89.78 grados.                               |
| 9  | 260410     | Fecha de captura de datos 26 de abril de 2010.                |
| 10 | A          | No hay captura de datos de la variación magnética.            |

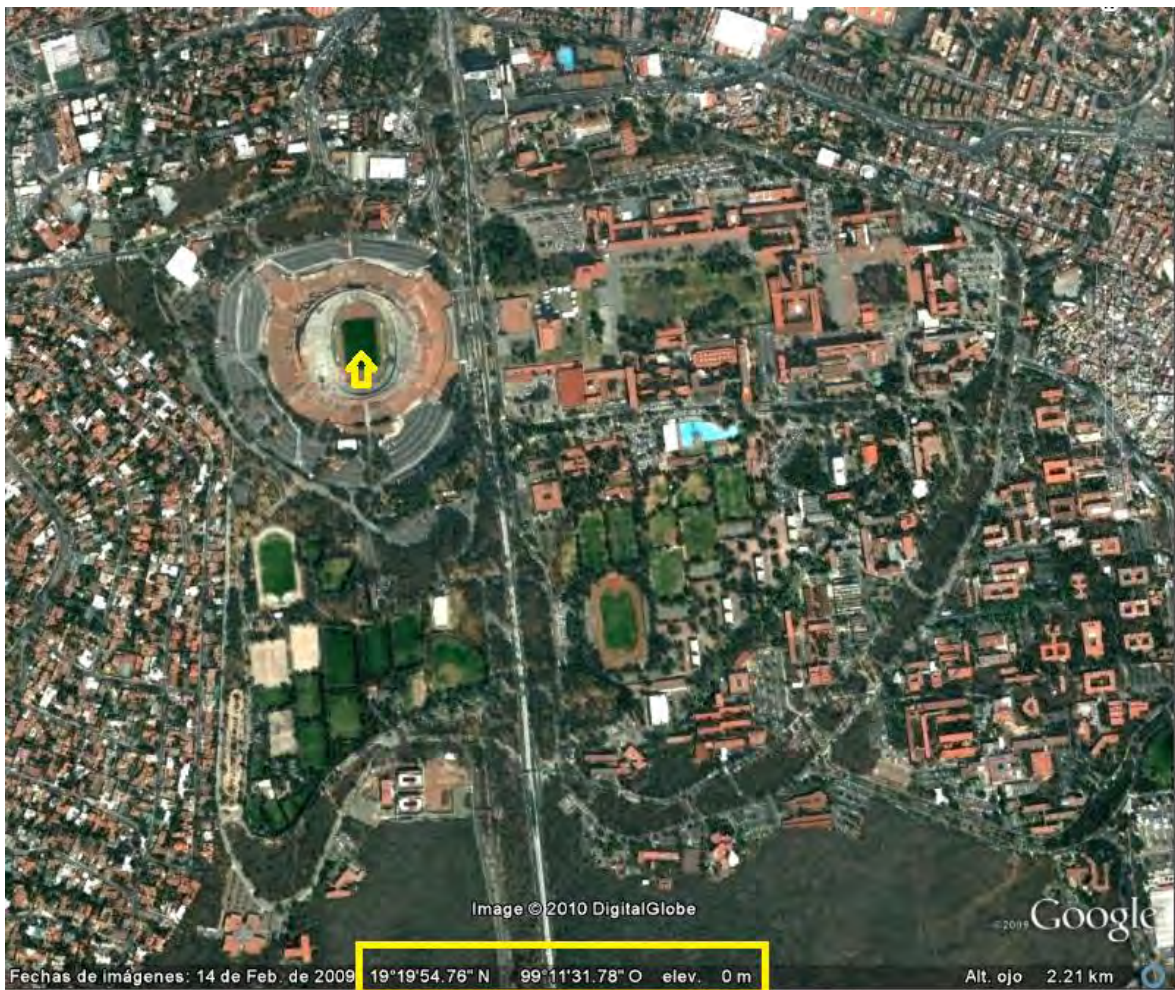
## Pruebas y resultados

De los datos obtenidos, los importantes (para esta aplicación) son los que proporcionan la ubicación del punto en cuestión, es decir, los puntos 3, 4, 5 y 6 que son la latitud, longitud y el tipo de ambas mediciones.

Entonces, una vez establecida la comunicación con el dispositivo GPS, se procede a capturar los datos que éste envíe para obtener de ahí la posición. Una vez conocidas las coordenadas se procede a buscar el mapa a mostrar. Para ello es necesario tener una relación mapa-coordenadas en el celular.

### 5.1.2 Relación mapa-coordenadas

Existe una relación entre coordenadas y mapas, por lo que es necesario contar con los mapas correspondientes. Se puede utilizar cualquier aplicación cartográfica con el fin de obtener las imágenes y las coordenadas que vamos a ocupar. Para las pruebas se ocupan imágenes de Google Earth. En la figura 5.2 se puede observar que el lugar seleccionado es el estadio olímpico México 68 y, en la parte inferior de la pantalla, se muestran las coordenadas correspondientes a la ubicación seleccionada.



**Figura 5.2.** Posición y coordenadas proporcionadas por Google Earth.

Para la aplicación se limita el área a ocupar, en este caso se obtienen las imágenes de Ciudad Universitaria. Se debe limitar la imagen a ocupar, para ello se obtienen las coordenadas de cada extremo de nuestro mapa:

IS (Izquierda Superior): Latitud  $19^{\circ}20'8''N$ , Longitud  $99^{\circ}11'48''O$

DS (Derecha Superior): Latitud  $19^{\circ}20'8''N$ , Longitud  $99^{\circ}10'50''O$

II (Izquierda Inferior): Latitud  $19^{\circ}19'22''N$ , Longitud  $99^{\circ}11'48''O$

DI (Derecha Inferior): Latitud  $19^{\circ}19'22''N$ , Longitud  $99^{\circ}10'50''O$

En la figura 5.3, se tienen marcadas las coordenadas de los 4 puntos que van a limitar el mapa. Sin embargo, si se observa detenidamente, es redundante tener las coordenadas de cada punto, así que sólo se ocupan dos puntos, puntos extremos de la imagen. Se van a utilizar las coordenadas de la esquina superior izquierda y la esquina inferior derecha. Siempre que se inserte una nueva imagen para la aplicación hay que dar estas dos coordenadas.



**Figura 5.3.** Límites del área a mostrar.

De la misma manera que se obtuvo el mapa y sus coordenadas la cantidad de información depende de la cantidad de acercamientos que se desea mostrar en la aplicación.

Una vez obtenidos los mapas hay que guardarlos en el celular para poder referenciarlos posteriormente. Ya se ha comentado que existen dos opciones para guardar las imágenes: se puede utilizar la base de datos del celular o utilizar una estructura de datos programada.

El celular utiliza almacenamiento persistente y se maneja con las clases del paquete rms (record management system). Rms define una forma básica de almacenamiento en la cual se manejan los registros a modo de cadenas de bytes. El problema con este tipo de almacenamiento es que para grandes cantidades de datos diferentes hay que proveer clases que manejen almacenamiento en bytes. Además, dependiendo de la KVM (entorno de ejecución de la plataforma J2ME) y del dispositivo, su capacidad de almacenamiento puede variar entre 32 kilobytes y 5 megabytes.

Otra opción es utilizar una estructura de datos tipo árbol para poder organizar la información de forma jerárquica y poder acceder a los datos de una manera más rápida.

Para ambos sistemas de almacenamiento el código se incrementa debido a que, o se implementa un código que convierta una imagen a bytes, o se implementa un código para crear la estructura de datos a ocupar. De igual manera, se depende de la memoria del dispositivo. En cualquier caso, es necesario guardar la imagen y dos coordenadas (superior izquierda e inferior derecha).

## 5.2 Posición Geográfica desplegada en la pantalla del celular

Una vez descrito el funcionamiento interno de la aplicación, es necesario imaginar un entorno o presentación. Hay dos posibilidades, o se crea un formulario y se agregan botones estáticos para formar un menú o crear una Interfaz Gráfica de Usuario (GUI).

En este caso se va a generar una interfaz gráfica. En Java es posible implementar interfaces para juegos, aprovechando la clase GameCanvas que permite manejar diferentes Sprites (cualquier elemento gráfico) y un manejador de capas (Layer Manager) para manejar los Sprites. Otra ventaja de esta clase es que permite controlar eventos del teclado y actualizar (redibujar) la pantalla en cualquier momento.

Hay que tener en cuenta la referencia que maneja la pantalla del celular así como los métodos getHeight() y getWidth() para obtener el ancho y largo de nuestra pantalla.

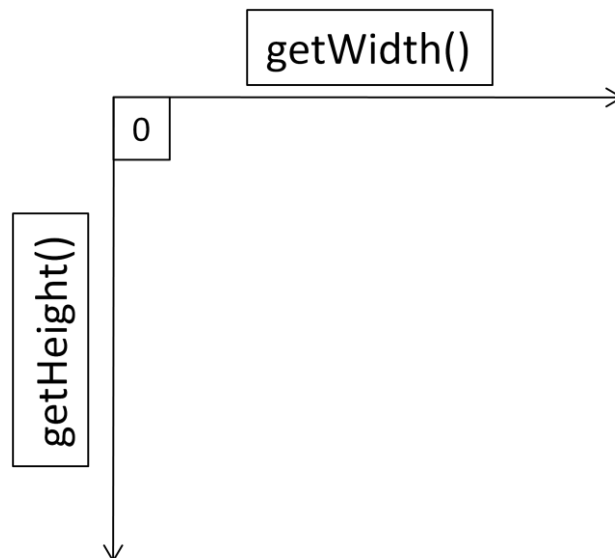


Figura 5.4. Referencia de la pantalla del celular.

Con esto se pueden ubicar los elementos gráficos en cualquier parte de la pantalla independientemente del dispositivo que se utilice. Esto genera una aplicación dinámica.

Cada vez que se ejecuta la aplicación es común mostrar una imagen o alguna leyenda relativa a la aplicación. Este es un recurso recurrente en aplicaciones para dispositivos móviles.

En este caso se utilizará una alerta con una duración determinada que mostrará el nombre de la aplicación y una imagen (si la alerta del dispositivo soporta mostrar imágenes) y, después de un tiempo, mostrará el menú principal.

### 5.2.1 Botón “Ubicación”

En la figura 5.6 se observa la pantalla principal de la aplicación y la secuencia de la misma. La primera opción permite conectarse con el dispositivo GPS y, a la postre, mostrar la ubicación en un mapa.



Figura 5.5. Alerta de la aplicación.



Figura 5.6. Menú principal de la aplicación

La aplicación realiza una búsqueda de dispositivos Bluetooth cercanos, por lo que es necesario tener activado el dispositivo Bluetooth del celular, en caso de que no esté activado se envía una alerta para que el usuario autorice que la aplicación lo active. Una vez terminado el proceso de búsqueda, se selecciona el dispositivo indicado para obtener los datos de la posición.



Figura 5.7. La aplicación activa el Bluetooth.

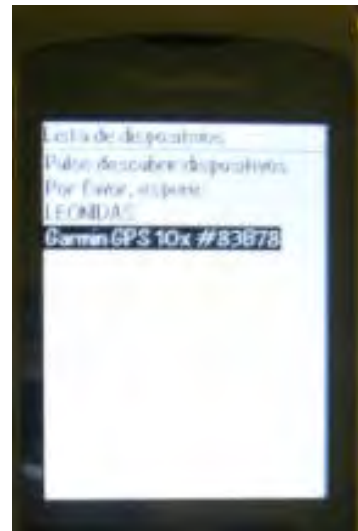


Figura 5.8. Se selecciona el GPS.

Una vez que se obtienen los datos, se buscan las coordenadas en el sistema de almacenamiento elegido, en caso de que exista una correspondencia, se muestra la imagen, de lo contrario se envía el mensaje de error.



Figura 5.9. Buscando mapa a cargar.



Figura 5.10. Mapa mostrado en pantalla.

### 5.2.2 Botón “Ayuda”

Este botón emite una explicación detallada del comportamiento del software. También alerta al usuario sobre posibles mensajes de error para que éste sea capaz de solucionarlos. Esta explicación debe ser lo más detallada posible sin llegar a ser redundante y/o tediosa.

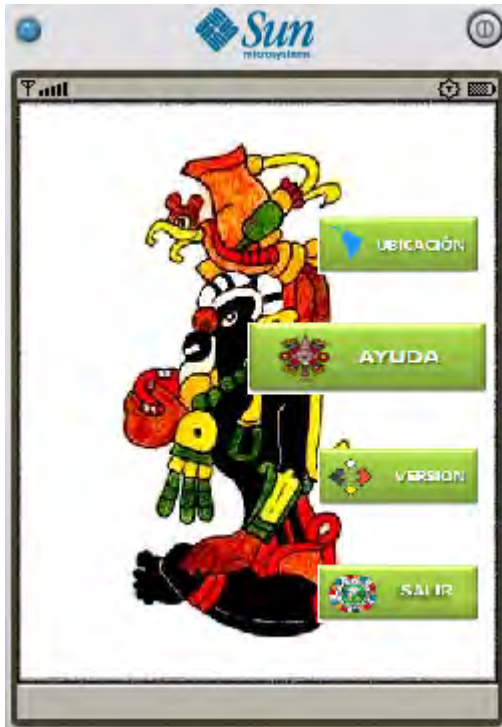


Figura 5.11. Botón “Ayuda”.



Figura 5.12. Información del botón “Ayuda”.

### 5.2.2 Botón “Versión”

Aquí se inserta una opción que no se había contemplado anteriormente y que es exclusivamente informativa sobre la aplicación. Al ser una aplicación libre se esperan contribuciones hacia la misma, por lo que se propone un botón que explique la versión que se está ejecutando (dependiendo del número de arreglos/contribuciones que se aporten al software) así como el nombre de las personas que contribuyan a la causa.



Figura 5.13. Botón "Versión".



Figura 5.14. Información del botón "Versión".

#### 5.2.4 Botón "Salir"

La última opción, el botón "Salir", termina la aplicación para regresar el control al dispositivo celular.



Figura 5.15. El botón "Salir" termina la aplicación.



### 5.2.5 Resultados

Las ventanas que muestran la ayuda y la versión del dispositivo son elementos básicos que se generaron a partir de un formulario y se les agregó una imagen y botones, por ello se pudo eliminar los mensajes de error como se había planteado en los diagramas de actividades del diseño lógico de la aplicación, en el capítulo 3. Los mensajes de error que se manejan se presentan para el caso de que las imágenes no se puedan cargar, la cual genera Java automáticamente para cualquier imagen en cualquier momento, cuando no se encuentran dispositivos Bluetooth cercanos o cuando no se puede establecer la comunicación con el dispositivo seleccionado.

El botón “Salir” se desarrolló de manera eficiente, es decir, cuando el usuario oprime esta opción, se termina la aplicación y regresa el control al celular.

En el menú principal y con el botón “Ubicación” se obtuvieron algunos tropiezos. Para generar el menú principal fue necesario hacer uso de elementos gráficos. Para ello se heredó de la clase GameCanvas, y así poder redibujar la pantalla en cualquier momento y poder leer las teclas presionadas por el usuario.

La ubicación de los “botones”, que en realidad son imágenes, se pudo lograr gracias al método setRefPixelPosition que pertenece a la clase Sprite. Este recurso permite crear un menú dinámico, es decir, la posición de los botones, el tamaño de los botones y el tamaño de la imagen de fondo dependen completamente del tamaño de la pantalla del celular. Esto se logró con una clase que redimensiona las imágenes.

Al momento de oprimir el botón “Ubicación”, la aplicación llama a la clase MostrarUbicacion. Esta clase, a su vez, manda llamar a la clase ConexionBluetooth, que es la que permite buscar dispositivos GPS en el alcance. Un posible problema es que el usuario no tenga encendido el Bluetooth del celular, así que la aplicación tiene que verificar que estuviera encendido y, de no estarlo, dar la opción de encenderlo.

La localización no es sencilla ni inmediata, así que hay que tener paciencia y, en ocasiones, hay que ejecutarla más de una vez para poder ubicar el dispositivo GPS. Una vez que se obtiene la lista de dispositivos Bluetooth dentro del radio de alcance, hay que seleccionar el adecuado para poder realizar la conexión serial. Esta conexión es otro paso que toma su tiempo, sin embargo, es un paso muy importante, ya que, una vez establecida la comunicación serial, se puede capturar la información enviada por el GPS y así obtener los datos buscados (las coordenadas).

El último paso, mostrar el mapa, parece muy sencillo y trivial, empero, conlleva mucha dedicación y trabajo previo, esto debido a que primero hay que obtener los mapas necesarios de alguna aplicación, en este caso se utilizaron imágenes de Google Earth, para después insertarlos en nuestro sistema de almacenamiento de una forma lógica y ordenada, esto es, se debe insertar la longitud y latitud correspondiente a los dos puntos de la imagen, es decir 4 coordenadas, cada una incluye grados, minutos, segundos y orientación cardinal.

## Pruebas y resultados

Entonces, es necesario determinar la manera en que se van a separar estos valores (con coma, con punto, con guion, etc.) para poder extraer los números y compararlos con los recibidos del GPS. Una vez establecido el orden de las coordenadas y el separador de valores, es necesario establecer un formato para el nombre del mapa, debido a que es posible estar en las coordenadas correctas y no tener el nombre adecuado para invocar al mapa, sobre todo cuando dejamos en claro que el usuario es el encargado de descargar la cantidad de mapas que requiera y éste puede insertar los mapas con el nombre que se le ocurra. Para ello se propone utilizar los grados, minutos y segundos de latitud y longitud inicial y final, es decir GGMMSSGGGMMSS-GGMMSSGGGMMSS.png, donde, leyendo de izquierda a derecha, las primeras coordenadas (GGMMSS) corresponden a la Latitud Superior Izquierda, las siguientes coordenadas (GGGMMSS) corresponden a la Longitud Superior Izquierda, las siguientes coordenadas después del guión (GGMMSS) pertenecen a la Latitud Inferior Derecha y las últimas coordenadas (GGGMMSS) pertenecen a la Longitud Inferior Derecha, con las imágenes guardadas en formato png. De esta manera ya se sabe el nombre del mapa que se debe cargar en pantalla.

Por último, hay indicar la posición en la que se encuentra el usuario. Para ello se deben tener en cuenta las coordenadas del mapa, se obtiene la distancia entre puntos, tanto en latitud como en longitud (recordando que son minutos y segundos y se miden de 0 a 60, por lo que la resta no es directa) y se obtiene el ancho de la imagen. Se realiza una regla de tres entre estos datos y se obtiene la ubicación de las coordenadas recibidas. Una vez establecida la posición en el mapa, se utiliza un elemento gráfico para cargar el mapa y otro para indicar la posición. Como el mapa, normalmente, superará el tamaño de la pantalla del celular, se muestra el cuadrante donde se encuentran las coordenadas recibidas.

# Conclusiones

En la actualidad, las Tecnologías de la Información (TI) son herramientas muy útiles y, en algunos casos básicas, para el desarrollo de actividades en diversos ámbitos de la vida. Estas herramientas tecnológicas han hecho más eficiente el desempeño de actividades y han simplificado esfuerzos y tiempo, y, a su vez, han facilitado la vida del ser humano.

De igual manera, las TI crecen constantemente y es importante estar al pendiente de los cambios que puedan presentar no sólo para aprovecharlos sino para crecer con ellos, es decir, aprovechar las mejoras de TI para poder desarrollar nuevas tecnologías que ayuden al desarrollo de actividades.

Es, con base en lo anterior, con lo que surge la idea de crear una aplicación que ayude a ubicar la posición de un usuario en un mapa que se despliegue en su celular. De igual manera, esta idea se propició debido a los recursos tecnológicos disponibles, como el lenguaje de programación orientado a objetos Java, en especial J2ME que es el que permite crear programas para celulares, ya que la mayoría de teléfonos celulares en la actualidad poseen una máquina virtual para ejecutar código J2ME; y a los dispositivos GPS que proporcionan información cartográfica sobre la posición en la que se encuentran.

Así como el GPS aprovecha los satélites en órbita y el estándar NMEA 0183, y este estándar a su vez aprovecha el Sistema Geodésico Mundial WGS-84, esta propuesta planea aprovechar el lenguaje Java y el dispositivo GPS para intentar crear una aplicación útil a la sociedad.

Sin embargo, una cosa es desarrollar nuevas formas de TI y otra redundar en tecnologías ya existentes. Por ello, antes de realizar la aplicación hubo que investigar aplicaciones en el mercado que ubicaran la posición mediante un GPS.

Investigar sobre aplicaciones similares en el mercado fue el primer paso para seguir con el proyecto. Al principio no se sabía que tan redundante podría ser la aplicación propuesta, es decir, cuantas otras aplicaciones con características similares podían existir ya en el mercado. Empero, la búsqueda ayudó a determinar que la mayoría de las aplicaciones existentes estaban disponibles para PC, y, de ellas, las gratuitas o libres estaban desarrolladas para mostrar mapas cartográficos.

De las aplicaciones disponibles para celular, Nokia Maps es la que más se acerca a las características propuestas, sin embargo, tiene el problema de que para mostrar los mapas a detalle necesita conectarse a la red móvil de donde emanan dos problemas: primero, el gasto que genera conectarse a la red de la telefonía móvil y descargar los mapas; segundo, si no hay señal en el área, no va a ser posible el enlace a la red de telefonía y, por ende, la descarga de mapas sería imposible y la aplicación no sería capaz de ayudar al usuario en caso de algún desajuste.

Realizar lo anterior fue un paso clave para saber de qué punto partir, ya que, como se mencionó, no es lógico realizar una aplicación que ya exista, la idea es mejorar lo existente y/o usarlo para crear cosas nuevas. Con esta filosofía y con la de compartir e intercambiar conocimientos es como se empieza a trabajar en la aplicación.

## Conclusiones

Ya que se confirmó que no existe una aplicación igual a la que se plantea implementar, el siguiente paso fue realizar el análisis de la aplicación, es decir, las opciones que iba a tener la aplicación así como la secuencia que iban a seguir cada una de las opciones, pensando siempre en la simplicidad para el usuario final. La ventaja es que siempre se tuvo en mente la manera en que el usuario iba a interactuar con la aplicación, intentando ser muy intuitiva para abarcar un número más amplio de usuarios finales.

Una vez realizado el análisis se procedió a la etapa del diseño, donde se hizo un bosquejo de la manera en que el usuario iba a interactuar con la aplicación y, al mismo tiempo, las acciones que, internamente, llevaría a cabo el programa. Aquí, y de manera simultánea, se definió la arquitectura cliente-servidor de la aplicación. Esto es, se identificó el alcance de cada capa de la aplicación y sus procesos. El diseño aclaró la manera de realizar la aplicación, al tiempo que se detectó la incertidumbre por temas no contemplados, como la conexión Bluetooth que complicaba de manera sustancial el desarrollo.

La culminación de las dos fases anteriores (análisis y diseño) es la programación, es aquí donde la estructura determinada en las fases previas toma forma y da como resultado la aplicación deseada. El problema es que la práctica siempre es diferente a la teoría y, en esta ocasión, no fue la excepción. Un ejemplo de esto se puede observar en los casos de uso propuestos en el capítulo 3. En el caso de uso “Mostrar Ayuda” se menciona como curso alternativo de la aplicación: “En caso de que el formulario que muestra la ayuda no se pueda cargar, se muestra un mensaje de error *“Error al cargar la ayuda”* y se regresa al menú principal”. Sin embargo, una vez que se programó ese botón y, en específico, ese formulario, se pudo observar que la única manera para que el formulario Ayuda no se pudiera crear es que la clase Ayuda no exista y, si esto ocurriera, el error lo manejaría la aplicación principal y ésta no se podría ejecutar.

Para crear el menú gráfico se tuvo que investigar la manera en que se manejan elementos gráficos en J2ME, para ello fue necesario familiarizarse con el concepto de Sprites que sirven para cargar imágenes y posicionarlas. Sin embargo, se requiere otra instancia que permita manipular todos los elementos gráficos con los que se cuenta, es decir, controlar cuando deben aparecer, en qué orden, etc. La instancia que permite el manejo de elementos gráficos es el manejador de capas (Layer Manager). Gracias a esta clase es posible cargar los sprites (que poseen una imagen y posición establecida con respecto a la pantalla) y ubicarlos en la pantalla del celular en la posición y orden deseado, permitiendo así construir una aplicación gráfica.

Con las imágenes cargadas y controladas, se muestran los Sprites adecuados dependiendo la selección del usuario. Es aquí donde cobra importancia controlar eventos del teclado mediante el método `getKeyStates()`. Entonces, cuando el usuario presiona el botón del teclado 2 (sube) y 8 (baja) o, si el teclado tiene teclas de desplazamiento, con tecla arriba (sube) o tecla abajo (baja), se captura el evento del teclado y se le ordena al manejador de capas qué imágenes mostrar. Y es así como se puede interactuar con el menú gráfico.

En el proceso de creación de la aplicación surgió la idea de insertar otro botón que no se había pensado y que no tenía nada que ver con el funcionamiento del programa pero que era importante, no tanto para el usuario final sino para algún desarrollador interesado en contribuir con la mejora de la aplicación, ya que, como se ha mencionado antes, se plantea que el software sea libre para que pueda ser mejorado. En ese sentido, se insertó el botón “Versión” intentando que ahí se ingresen el nombre de cada contribuyente así como la versión que generó, dependiendo el número de arreglos hechos a la aplicación.

La creación de los botones “Ayuda”, “Versión” y “Salir” resultaron con base en lo esperado, sin embargo, el botón “Ubicación” resultó con mejoras sustanciales. Al presionar este botón, la aplicación tiene que conectarse al dispositivo GPS vía Bluetooth, para ello es necesario rastrear los dispositivos GPS que estén al alcance del celular. En un principio se había pensado limitar la aplicación para que sólo se pudiera conectar con el GPS con el que se realizaron las pruebas, que es Garmin 10X. Sin embargo, el permitir conexiones con diferentes dispositivos GPS hace más útil la aplicación. Una vez descubiertos los dispositivos dentro del alcance hay que emparejarse con el GPS.

Los datos del dispositivo GPS se obtienen mediante una conexión serial. Desde la PC se puede utilizar la hyperterminal (en Windows) o el comando rfcmm (en Linux). J2ME utiliza una conexión btsp (Perfil de puerto serial por Bluetooth), de esta manera se puede capturar la información enviada por el GPS para obtener la trama a analizar y, posteriormente, las coordenadas.

Analizar las tramas enviadas por el GPS es una tarea interesante, ya que se puede obtener información extra que también puede ser útil y/o generar otra función a nuestra aplicación. El GPS envía diferentes tramas de donde se pueden obtener los datos buscados (las coordenadas) sin embargo, como ya vimos en el capítulo 5, se utilizó la trama RMC.

El proyecto resultó más ambicioso que lo contemplado en un principio, y surgieron imprevistos al mostrar los mapas en pantalla, ya que se requería un sistema de almacenamiento donde insertar las coordenadas y las imágenes. Se estudió la posibilidad de utilizar la base de datos del celular o una estructura de datos. Se optó por la segunda opción ya que el usuario final va a ingresar las coordenadas y tiene que ser un proceso sencillo, simplemente seguir un patrón de un archivo y el programa cargará de ahí la información. De igual manera, las imágenes se tienen que guardar en una carpeta específica (Maps) con el formato descrito al final del capítulo 5. Lo ideal sería realizar una aplicación para PC que permita cargar coordenadas y mapas automáticamente en la aplicación, así que esta es una mejora que podría tener el programa a futuro.

Realizar una aplicación innovadora siempre es emocionante, para ello se buscó demostrar que no existe una aplicación como la propuesta descargando diferentes tipos de software, interactuando con ellos y sacando conclusiones de sus ventajas y desventajas con respecto a las características de la aplicación propuesta. Permitir que pueda ser utilizada por cualquier persona es gratificante, es por esto que la aplicación no utiliza ninguna licencia para activarla y, por ende, se distribuye de manera gratuita.

## Conclusiones

Adquirir o consolidar conocimientos es sumamente grato. Ya sean conocimientos de la carrera, como lo son el lenguaje para dispositivos móviles J2ME y el sistema de almacenamiento que utilizan los celulares (adquiridos) o las estructuras de datos e ingeniería de software (reforzados). O bien, conocimientos en otras ciencias, como lo es el estándar NMEA 0183 o el Sistema Geodésico WGS-84.

Las profesionistas que nos desarrollamos en el campo de trabajo de las Tecnologías de la Información, estamos en constante aprendizaje, no sólo porque desenvolvemos nuestros conocimientos con otros profesionistas, sino porque las tecnologías se desarrollan rápidamente y hay que estar a la vanguardia en todos los aspectos, lo que implica documentarse, investigar y estudiar toda la vida. Cuando un profesionista de este campo de trabajo pierde la inquietud de aprender o de estar a la vanguardia, pierde la esencia de la profesión. Esto es lo destacable de este trabajo, el deseo que deja por seguir adelante, por innovar, por trabajar para que las cosas se logren y sean útiles a la sociedad, utilizando siempre los recursos disponibles de la mejor manera y teniendo en mente que nunca se deja de aprender.

# Referencias

- [1] **Hangup**, GPS – Sistema de Posicionamiento Global – Concepto y General, [21/11/2007], <http://www.taringa.net/posts/info/963844/GPS---Sistema-de-Posicionamiento-Global---Concepto-y-General.html>, consultado el: 20/Mayo/2009.
- [2] **De la Cruz González, Felipe**, ¿Qué es el GPS?, [01/12/2004], <http://www.mailxmail.com/curso-introduccion-gps/que-es-gps>, consultado el: 19/Mayo/2009.
- [3] **De la Cruz González, Felipe**, Algo de Historia, [01/12/2004], <http://www.mailxmail.com/curso-introduccion-gps/algo-historia>, consultado el: 19/Mayo/2009.
- [4] **De la Cruz González, Felipe**, Cómo funciona el GPS (trilateración), [01/12/2004], <http://www.mailxmail.com/curso-introduccion-gps/como-funciona-gps-trilateracion>, consultado el: 19/Mayo/2009.
- [5] **De la Cruz González, Felipe**, Cómo funciona el GPS (cálculo de la distancia), [01/12/2004], <http://www.mailxmail.com/curso-introduccion-gps/como-funciona-gps-calculo-distancia>, consultado el: 19/Mayo/2009.
- [6] **De la Cruz González, Felipe**, Cómo funciona el GPS (sincronización), [01/12/2004], <http://www.mailxmail.com/curso-introduccion-gps/como-funciona-gps-sincronizacion>, consultado el: 20/Mayo/2009.
- [7] **De la Cruz González, Felipe**, Cómo funciona el GPS (posiciones de los satélites), [01/12/2004], <http://www.mailxmail.com/curso-introduccion-gps/como-funciona-gps-posiciones-satelites>, consultado el: 20/Mayo/2009.
- [8] **De la Cruz González, Felipe**, Cómo funciona el GPS (corrección de errores), [01/12/2004], <http://www.mailxmail.com/curso-introduccion-gps/como-funciona-gps-correccion-errores>, consultado el: 21/Mayo/2009.
- [9] **Topografía**, cartografía-DATUM, [13/12/2001], <http://www.elgeomensor.cl/datum.pdf>, consultdo el: 29/Abril/2010.
- [10] **©2010 Google**, Importación de datos en Google Earth – Guía del usuario de Google Earth, [25/03/2010], [http://earth.google.es/userguide/v4/ug\\_importdata.html](http://earth.google.es/userguide/v4/ug_importdata.html), consultado el: 30/Abril/2010.
- [11] **Salazar Dagoberto**, La forma general de la Tierra, [13/09/2006], <http://nacc.upc.es/tierra/node10.html>, consultado el: 30/Abril/2010.
- [12] **Calero, Enrique**, Sistema de Referencia WGS-84, [23/03/2003], <http://ecalero.tripod.com/sitebuildercontent/sitebuilderfiles/wgs-84.pdf>, consultado el: 30/Abril/2010.

## Referencias

- [13] **Klaus**, The NMEA 0183 Protocol, [04/08/2001], <http://www.tronico.fi/OH6NT/docs/NMEA0183.pdf>, consultado el: 21/Mayo/2009.
- [14] **Anonimo**, NMEA, [07/01/2004], [http://www.marimsys.com/paginas/nmea\\_codigo.htm](http://www.marimsys.com/paginas/nmea_codigo.htm), consultado el: 21/Mayo/2009.
- [15] **Mihai**, NMEA-0183 Protocol Description, [03/03/2004], <http://www.remember.ro/dl/nmea0183.pdf>, consultado el: 22/Mayo/2009.
- [16] **Christian**, ¿Qué es Bluetooth?, [18/10/2007], <http://tecnio.com/%C2%BFque-es-bluetooth/>, consultado el: 1/Junio/2009.
- [17] **blogElectronica**, ¿Qué es el bluetooth? | blogElectronica.com, [8/03/2008], <http://www.blogelectronica.com/conceptos-de-la-tecnologia-bluetooth/>, consultado el: 1/Junio/2009.
- [18] **Revista RED**, Bluetooth, afila sus dientes, [07/2002], <http://www.eveliux.com/mx/bluetooth-afila-sus-dientes.php>, consultado el: 1/Junio/2009.
- [19] **Pressman, Roger S.**, Software Engineering, A beginner's Guide.- Estados Unidos: McGraw-Hill, 1988, p.
- [20] **Ince, D. C.**, Ingeniería de Software.- 1ra ed.- Estados Unidos: Addison-Wesley Iberoamericana, 1989, p.
- [21] **Charly**, Ciclo de vida del Software, [24/08/2006], [http://www.cepeu.edu.py/LIBROS\\_ELECTRONICOS\\_3/lpcu097%20-%2001.pdf](http://www.cepeu.edu.py/LIBROS_ELECTRONICOS_3/lpcu097%20-%2001.pdf), consultado el: 26/Junio/2009.
- [22] **De Ezcurra, Héctor**, Cómo armar una conexión GPS-PC casera, [2002] <http://www.navigare.com.ar/conexcaseragps-pc.html>, consultado el: 14/Abril/2010.
- [23] **De Ezcurra, Héctor**, Cómo probar una conexión GPS-PC, [2002] <http://www.navigare.com.ar/comoprobarconexgps-pc.html>, consultado el: 14/Abril/2010.
- [24] **Anónimo**, Conexión del GPS al PC, [25/07/2001], <http://www.aeroveleta.com/taller/conexion.htm>, consultado el: 14/Abril/2010.
- [25] **yo**, OpenCPN+GPS, [20/01/2010] <http://opencpnayudaes.yolasite.com/resources/OpenCPN+GPS.pdf>, consultado el: 15/Abril/2010.
- [26] **Anonimo**, Cómo conectar un GPS al PDA, <http://www.laempresaweb.com/articulos/divulgacion/conectar-gps-pda/>, consultado el: 15/Abril/2010.



- [27] **Zanotti, Alejandro**, Montar cable serie para conectar GPS y PDA, [4/11/2005], [http://www.pdaexpertos.com/Tutoriales/Hardware/Montar\\_cable\\_serie\\_para\\_conectar\\_GPS\\_y\\_PDA.shtml](http://www.pdaexpertos.com/Tutoriales/Hardware/Montar_cable_serie_para_conectar_GPS_y_PDA.shtml), consultado el: 15/Abril/2010.
- [28] **Zavala, Jesus**, La Ingeniería de Software, [31/03/2008] <http://www.angelfire.com/scifi/jzavalar/apuntes/IngSoftware.html#DisConceptual>, consultado el: 12/Abril/2010.
- [29] **Vega, Miguel**, Casos de uso, [05/03/2004] <http://lsi.ugr.es/~mvega/docis/casos%20de%20uso.pdf>, consultado el: 12/Abril/2010.
- [30] **Lauro Soto**, Panorama General Diseño Físico y Lógico <http://www.mitecnologico.com/Main/PanoramaGeneralDise%F1oFisicoYLogico>, consultado el: 13/Abril/2010.
- [31] ©2010 **Google**, Uso de dispositivos GPS con Google Earth - Guía del usuario de Google Earth, [25/03/2010], [http://earth.google.es/userguide/v4/ug\\_gps.html](http://earth.google.es/userguide/v4/ug_gps.html), consultado el: 16/Abril/2010.
- [32] **Newman, Des & Lorraine**, Official OziExplorer Web Site - GPS Mapping Software for Tracking and Navigation. Supports Garmin, Magellan, Lowrance and GPS, [22/01/2010], <http://www.ozieplorer.com/>, consultado el: 16/Abril/2010.
- [33] **José**, Ayuda en español OpenCPN, [11/02/2010] <http://opencpnayudaes.yolasite.com/>, consultado el: 16/Abril/2010.
- [34] **Chourdakis, Michael**, Welcome to Turbo GPS, [20/11/2009], <http://www.turboirc.com/tgps/>, consultado el: 17/Abril/2010.
- [35] **Geo Terrestrial**, GeoTerrestrial GPSToday, [30/06/09], <http://geoterrestrial.com/gpstoday/>, consultado el: 18/Abril/2010.
- [36] **Peters, Dana**, GPS Track, [19/06/2009], <http://www.qcontinuum.org/gpstrack/index.htm>, consultado el: 18/Abril/2010.
- [37] **Glenn Baddeley**, GPS – NMEA sentence information, [18/02/2002], <http://aprs.gids.nl/nmea/#rmc>, consultado el: 26/Abril/2010.

# Anexos

## Formato de Sentencias NMEA enviadas por el GPS

Las sentencias NMEA típicas que corresponden a los receptores GPS son las siguientes:

**GGA:** Datos arreglados del Sistema Global de Posicionamiento (Global Position System Fix Data)

```
$GPGGA,hhmmss.ss,lll.ll,a,yyyy.yy,b,x,yy,z.z,c.c,M,v.v,M,d.d,www*hh
```

hhmmss.ss: Hora UTC en la que fue tomado el dato (horas, minutos, segundos).

lll.ll: Latitud en grados y minutos.

a: Tipo de latitud: N o S.

yyyy.yy: Longitud en grados y minutos.

b: Tipo de longitud E u O.

x: Indicador de la Calidad de GPS (0 = no Válido; 1 = Ajuste de GPS; 2 = Ajuste de DGPS).

yy: Número de Satélites en uso.

z.z Dilución de Posición Horizontal.

c.c: Altitud de la Antena Sobre/Bajo Nivel del Mar Intermedio (geoide).

M: Medida de la Altitud de la antena en metros.

v.v: Separación Geoidal (Ajuste entre elipsoide terrestre WGS-84 y nivel del mar intermedio).

M: Unidad de medida de la separación Geoidal en metros.

d.d: Intervalo en Segundos desde la última actualización DGPS en segundos.

www: Identificador de la estación DGPS.

\*hh: Suma de Verificación.

**GLL:** Posición Geográfica, Latitud y Longitud (Geographic position, Latitude and Longitude)

```
$GPGLL,lll.ll,a,mmmm.mm,b,hhmmss.ss,A*ff
```

lll.ll: Latitud en grados y minutos

a: Tipo de latitud: N o S.

mmmm.mm: Longitud en grados y minutos.

b: Tipo de longitud E u O.

hhmmss.ss: Hora UTC en la que fue tomado el dato (horas, minutos, segundos).

A: Validez del dato (A = válido, V = no válido) del 4to satélite.

\*ff: Suma de Verificación.

**GSA:** Dilución de Precisión del Sistema Global de Navegación Satelital y satélites activos (GNSS DOP and Active Satellites)

```
$GPGSA,x,y,z1,z2,z3,z4,z5,z6,z7,z8,z9,z10,z11,z12,p,p,h,h,v.v*ff
```

## Anexos

x: Modo de operación:

M = Manual, Forzado a Operar en 2D o 3D

A = Automático 3D/2D

y: Modo de corrección de datos:

1 = Fijo no Disponible

2 = 2D

3 = 3D

z1 a z12: Identificador de satélites en vista (SV) usados para el ajuste de posición (máximo 12).

p.p: Dilución de Precisión Posicional (PDOP).

h.h: Dilución de Precisión Horizontal (HDOP).

v.v: Dilución de Precisión Vertical (VDOP).

\*ff: Suma de Verificación.

**GSV: Satélites en vista del Sistema Global de Navegación Satelital (GNSS Satellites in View)**

\$GPGSV,x,y,zz,aa,ee,rr,ii,s1,s2,s3,s4,t1,t2,t3,t4,c1,c2,c3,c4,\*ff

x: Número total de mensajes de este tipo en este ciclo.

y: Número de mensajes.

zz Número total de satélites en vista.

aa: Identificación PRN del 1er satélite.

ee: Elevación en grados, 90 como máximo, del 1er satélite.

rr: Azimut, grados del norte verdadero, de 000 a 359, del 1er satélite.

ii: Intensidad de la señal en decibeles (00-99) del 1er satélite.

s1: Identificación PRN del 2do satélite.

s2: Elevación en grados, 90 como máximo, del 2do satélite.

s3: Azimut, grados del norte verdadero, de 000 a 359, del 2do satélite.

s4: Intensidad de la señal en decibeles (00-99) del 2do satélite.

t1: Identificación PRN del 3er satélite.

t2: Elevación en grados, 90 como máximo, del 3er satélite.

t3: Azimut, grados del norte verdadero, de 000 a 359, del 3er satélite.

t4: Intensidad de la señal en decibeles (00-99) del 3er satélite.

c1: Identificación PRN del 4to satélite.

c2: Elevación en grados, 90 como máximo, del 4to satélite.

c3: Azimut, grados del norte verdadero, de 000 a 359, del 4to satélite.

c4: Intensidad de la señal en decibeles (00-99) del 4to satélite.

\*ff: Suma de Verificación.

**RMC:** Recomendación Mínima de datos específicos del Sistema Global de Navegación Satelital (Data Recommended Minimum Specific GNSS Data)

\$GPRMC, hhmmss.ss,A,IIII.II,a,yyyyy.yy,a,x.x,x.x,ddmmyy,x.x,a\*hh

Donde:

hhmmss.ss: Hora UTC en la que fue tomado el dato (horas, minutos, segundos).

A: Advertencia de navegación recibida (A = correcto, V = advertencia).

IIII.II: Latitud en grados y minutos.

a: Tipo de latitud N o S.

yyyyy.yy: Longitud en grados y minutos.

a: Tipo de longitud E u O.

x.x: velocidad sobre la tierra en knots.

x.x: Pista o curso cumplido en grados.

ddmmyy: Fecha UT en la que fue tomado el dato (día, mes, año).

x.x: Variación magnética oriental en grados

a: Orientación de la variación magnética (E = este, W = oeste).

\*hh: Suma de Verificación.

**VTG:** Ruta y velocidad sobre tierra (Course Over Ground and Ground Speed)

\$GPVTG,x.x,T,y.y,M,z.z,N,w.w,K\*hh

x.x: Pista terrestre, en grados.

T: Indica que la pista fue reparada.

y.y: Pista magnética, en grados

M: Indica que la pista fue reparada.

z.z: Velocidad en tierra.

N: Velocidad medida en Knots.

w.w: Velocidad en tierra.

K: Velocidad medida en kilómetros.

\*hh: Suma de Verificación.

## Código de la aplicación

-----  
 | XamanEk.java |  
 -----

```

package XamanEk;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;

/**
 * @author Jorge A. Solano
 */
public class XamanEk extends MIDlet {

    LayerManager LM;
    Display pantalla;
    Displayable pantAct;
    Image imgBienv, imgMapa;
    Alert bienvenida;
    Menu m;
    RedibujarImgs ri;
    ConexionBluetooth cb;
    Ayuda a;
    Version v;

    public XamanEk() {
        pantalla = Display.getDisplay(this); // Se obtienen una referencia al la pantalla
        initComponents(); // Este método inicializa todas nuestras interfaces
        setPanAct(bienvenida); // Se manda la alerta a pantalla
    }

    private void initComponents() {
        ri = new RedibujarImgs(); // Se crea una instancia tipo RedibujarImgs
        bienvenida = new Alert("", "", null, AlertType.INFO); // Se crea la alerta de bienvenida
        bienvenida.setTimeout(1000); // con un tiempo de 2 segundos
        bienvenida.setTitle("XAMAN EK"); // y como título "XAMAN EK"
        try{
            imgBienv = Image.createImage("/Imgs/a_xaman.png"); // Se carga la imagen
            // Redimensionar imagen al tamaño de la alerta
            imgBienv = ri.redImgAlerta(imgBienv, bienvenida.getWidth(), bienvenida.getHeight());
        } catch (Exception e) {
            imgBienv = null; // null si no está la imagen
        }
        bienvenida.setImage(imgBienv); // Se carga la imagen en la alerta
        LM = new LayerManager(); // Se crea un manejador de capas
    }

```

```

m = new Menu(this);           // Se crea una instancia tipo Menu
cb = new ConexionBluetooth(this); // Se crea una instancia tipo ConexionBluetooth
a = new Ayuda(this);         // Se crea una instancia tipo Ayuda
v = new Version(this);       // Se crea una instancia tipo Version
}

public void setPanAct(Displayable p) {
    if (p == bienvenida) { // Si la pantalla que se quiere mostrar es bienvenida
        pantalla.setCurrent(bienvenida, m); // Se muestra la alerta (bienvenida), luego el Menu (m)
        m.iniciar(); // Se invoca el método iniciar de la clase Menu
        new Thread(m).start(); // Se genera el Hilo
    } else { // De lo contrario
        if (p == cb.errorD) { // Cuando no encuentra dispositivos Bluetooth
            pantalla.setCurrent(cb.errorD, cb.listaDispositivos);
        } else {
            if (p == cb.errorC) { // Cuando no se puede establecer la conexión
                pantalla.setCurrent(cb.errorC, cb.listaDispositivos);
            } else {
                pantalla.setCurrent(p); // Se muestra en el Display la pantalla que se envía
            }
        }
    }
}

public void startApp() throws MIDletStateChangeException {
    if (pantAct == bienvenida) { // Si la pantalla actual es igual a bienvenida
        pantalla.setCurrent(bienvenida, m); // muestra la alerta y luego el menú
    } else { // De lo contrario
        if (pantAct != null) { // Si la pantalla Actual es diferente de null
            pantalla.setCurrent(pantAct); // se muestra la pantalla que se haya guardado (esto por
            // si entra una llamada, se muestra la última pantalla guardada)
        }
    }
}

public void pauseApp() {
}

public void mostrarUbicacion(){
    setPanAct(cb.listaDispositivos);
    cb.buscarDispositivos();
    //Temporal
    //cb.conecTemp();
}

```

## Anexos

```
public void regresar() {
    m.posicion = 0;
    setPanAct(m);
    m.iniciar();
    new Thread(m).start();
}

public void ayuda(){
    setPanAct(a.datos);    // Se invoca el método
}

public void version(){
    setPanAct(v.datos);    // Se invoca el método
}

public void salir(){
    destroyApp(true);    // Se invoca el método
}

public void destroyApp(boolean unconditional) {
    System.gc();    // Llamada al Garbage Collector
    notifyDestroyed();    // Se notifica la destrucción del MIDlet
}

public LayerManager getLM() {
    return LM;
}
}
```

-----  
| Menu.java |  
-----

```
package XamanEk;

import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;

/**
 * @author Jorge A. Solano
 */
public class Menu extends GameCanvas implements Runnable {

    LayerManager LM;
    Thread t;
    Image fondoImg, mostrarUbicImg, ayudaImg, versionImg, salidaImg;
    Image fondoImgFija, mostrarUbicImgFija, ayudaImgFija, versionImgFija, salidaImgFija;
    Sprite fondoSpr, mostrarUbicSpr, ayudaSpr, versionSpr, salidaSpr;
```

```

Sprite fondoSprFijo, mostrarUbicSprFijo, ayudaSprFijo, versionSprFijo, salidaSprFijo;
XamanEk xaman;
Graphics g;
boolean menu = true, stop = false, primVez = true;
int posicion = 0, vel = 100;
int alto, ancho;

public Menu(XamanEk xaman) {
    super(true);
    try {
        // Se cargan las imágenes
        mostrarUbicImg = Image.createImage("/Imgs/m_ubicacion.png");
        ayudalng = Image.createImage("/Imgs/m_ayuda.png");
        versionlmg = Image.createImage("/Imgs/m_version.png");
        salidalng = Image.createImage("/Imgs/m_salida.png");
        fondolmg = Image.createImage("/Imgs/m_xaman.png");
        // Se redimensionan los íconos
        mostrarUbiclmg = xaman.ri.redlcono(mostrarUbiclmg,0);
        ayudalng = xaman.ri.redlcono(ayudalng,0);
        versionlmg = xaman.ri.redlcono(versionlmg,0);
        salidalng = xaman.ri.redlcono(salidalng,0);

        // Se redimensionan los íconos fijos
        mostrarUbiclmgFija = xaman.ri.redlcono(mostrarUbiclmg,1);
        ayudalngFija = xaman.ri.redlcono(ayudalng,1);
        versionlmgFija = xaman.ri.redlcono(versionlmg,1);
        salidalngFija = xaman.ri.redlcono(salidalng,1);

        // Se redimesiona el fondo
        fondolmg = xaman.ri.redlmg(fondolmg);
    } catch (Exception e) {
        throw new RuntimeException ("No se puede cargar la imagenes de \"Menu\" : " + e);
    }
    // Referencias a las variables de XamanEk
    this.xaman = xaman;
    this.LM = xaman.getLM();
    g = getGraphics();
    t = new Thread(this);

    // Obtiene el ancho y alto de la pantalla
    alto = getHeight();
    ancho = getWidth();

    // Se crean los Sprites
    // Iconos fijos
    mostrarUbicSprFijo = new Sprite(mostrarUbiclmgFija, mostrarUbiclmgFija.getWidth(),
mostrarUbiclmgFija.getHeight());
    ayudaSprFijo = new Sprite(ayudalngFija, ayudalngFija.getWidth(), ayudalngFija.getHeight());

```



## Anexos

```
        versionSprFijo = new Sprite(versionImgFija, versionImgFija.getWidth(),
versionImgFija.getHeight());
        salidaSprFijo = new Sprite(salidaImgFija, salidaImgFija.getWidth(), salidaImgFija.getHeight());
        // Iconos móviles
        mostrarUbicSpr = new Sprite(mostrarUbicImg, mostrarUbicImg.getWidth(),
mostrarUbicImg.getHeight());
        ayudaSpr = new Sprite(ayudaImg, ayudaImg.getWidth(), ayudaImg.getHeight());
        versionSpr = new Sprite(versionImg, versionImg.getWidth(), versionImg.getHeight());
        salidaSpr = new Sprite(salidaImg, salidaImg.getWidth(), salidaImg.getHeight());
        fondoSpr = new Sprite(fondoImg, fondoImg.getWidth(), fondoImg.getHeight());
        // Se agregan los sprites al Manejador de capas (LayerManager)
        LM.append(mostrarUbicSpr);
        LM.append(ayudaSpr);
        LM.append(versionSpr);
        LM.append(salidaSpr);
        LM.append(mostrarUbicSprFijo);
        LM.append(ayudaSprFijo);
        LM.append(versionSprFijo);
        LM.append(salidaSprFijo);
        LM.append(fondoSpr);

        // Se colocan los sprites en la posición adecuada para ser mostrados
        fondoSpr.setRefPixelPosition(0,0);
        // Sprites fijos
        mostrarUbicSprFijo.setRefPixelPosition(ancho-mostrarUbicSprFijo.getWidth()-10,(alto/10)*2);
        ayudaSprFijo.setRefPixelPosition(ancho-ayudaSprFijo.getWidth()-10,(alto/10)*4);
        versionSprFijo.setRefPixelPosition(ancho-versionSprFijo.getWidth()-10,(alto/10)*6);
        salidaSprFijo.setRefPixelPosition(ancho-salidaSprFijo.getWidth()-10,(alto/10)*8);
        // Sprites móviles
        mostrarUbicSpr.setRefPixelPosition(ancho-mostrarUbicSpr.getWidth()-5,((alto/10)*2)-5);
        ayudaSpr.setRefPixelPosition(ancho-ayudaSpr.getWidth()-5,((alto/10)*4)-5);
        versionSpr.setRefPixelPosition(ancho-versionSpr.getWidth()-5,((alto/10)*6)-5);
        salidaSpr.setRefPixelPosition(ancho-salidaSpr.getWidth()-5,((alto/10)*8)-5);
    }

    /*
    * Se invoca en el menú setPantAct()
    * Hace visibles las imágenes de inicio e inicia el manejador de la aplicación
    */
    public void iniciar() {
        menu = true;
        fondoSpr.setVisible(true);
        // Sprites Fijos
        mostrarUbicSprFijo.setVisible(true);
        ayudaSprFijo.setVisible(true);
        versionSprFijo.setVisible(true);
        salidaSprFijo.setVisible(true);
    }
}
```

```

// Sprites Móviles
mostrarUbicSpr.setVisible(true);
ayudaSpr.setVisible(false);
versionSpr.setVisible(false);
salidaSpr.setVisible(false);
}

public void stop() {}

/*
 * Repinta la pantalla, muestra el menú y lee los controles
 */
public void run() {
    while (menu) { // Mientras menú sea igual a "true"
        drawScreen(g); // Se pinta la pantalla y
        teclasPresionadas(); // se "escuchan" las teclas presionadas
        try {
            Thread.sleep(vel); // Velocidad con la que se escucha el teclado
        } catch (InterruptedException ie) {
        }
    }
}

/*
 * Escuchador del teclado (posición):
 * 0: Mostrar Ubicación
 * 1: Ayuda
 * 2: Versión
 * 3: Salir
 */
private void teclasPresionadas() {
    int estadoTecla = getKeyStates();
    if (estadoTecla != 0) { // Si se presiona una tecla...
        if ((estadoTecla & FIRE_PRESSED) != 0) { // Si se presiona boton Enter...
            switch(posicion) { // Se verifica la posición en la que ser presionó Enter
                case 0: // Si presiona en la primera posición (botón Mostrar)
                    botonMostrar(); // Se invoca al método botonMostrar()
                    break;
                case 1: // Si presiona en la segunda posición (botón Ayuda)
                    botonAyuda(); // Se invoca al método botonAyuda()
                    break;
                case 2: // Si presiona en la tercera posición (botón Versión)
                    botonVersion(); // Se invoca al método botonVersion()
                    break;
                case 3: // Si presiona en la cuarta posición (botón Salir)
                    botonSalir(); // Se invoca al método botonSalir()
                    break;
            }
        }
    }
}

```

```

} else { // Si no se presiona enter, se verifica las teclas de movimiento
    // Para mostrar los íconos según la posición actual
    if ((estadoTecla & UP_PRESSED) != 0) { // Si presiona tecla hacia arriba
        if (posicion > 0) { // Si posición es mayor a cero
            posicion--; // La posición desciende
        } else { // Si posición es igual a cero
            posicion = 3; // Posición toma el valor de tres
        }
    } else {
        if ((estadoTecla & DOWN_PRESSED) != 0) { // Si presiona tecla hacia abajo
            if (posicion < 3) { // Si posición es menor a tres
                posicion++; // La posición avanza
            } else { // Si la posición es igual a tres
                posicion = 0; // Posición toma el valor de cero
            }
        }
    }
}
/* Según el valor que tenga la posición se muestra el ícono correspondiente*/
switch (posicion) {
    case 0: // Si la posición es cero, se muestra ícono mostrarUbicación
        iconoMostrar();
        break;
    case 1: // Si la posición es cero, se muestra ícono ayuda
        iconoAyuda();
        break;
    case 2: // Si la posición es cero, se muestra ícono versión
        iconoVersion();
        break;
    case 3: // Si la posición es cero, se muestra ícono salir
        iconoSalir();
        break;
}
}
}
}

/*
 * Oculta todos los sprites (íconos) excepto el de mostrarUbicación
 */
private void iconoMostrar() {
    mostrarUbicSpr.setVisible(true);
    ayudaSpr.setVisible(false);
    versionSpr.setVisible(false);
    salidaSpr.setVisible(false);
}

```

```

/*
 * Oculta todos los sprites (íconos) excepto el de ayuda
 */
private void iconoAyuda() {
    mostrarUbicSpr.setVisible(false);
    ayudaSpr.setVisible(true);
    versionSpr.setVisible(false);
    salidaSpr.setVisible(false);
}

/*
 * Oculta todos los sprites (íconos) excepto el de versión
 */
private void iconoVersion() {
    mostrarUbicSpr.setVisible(false);
    ayudaSpr.setVisible(false);
    versionSpr.setVisible(true);
    salidaSpr.setVisible(false);
}

/*
 * Oculta todos los sprites (íconos) excepto el de salir
 */
private void iconoSalir() {
    mostrarUbicSpr.setVisible(false);
    ayudaSpr.setVisible(false);
    versionSpr.setVisible(false);
    salidaSpr.setVisible(true);
}

/*
 * Oculta el menú, todos los sprites (íconos) e invoca al método
 * mostrarUbicación de la clase principal
 */
private void botonMostrar() {
    menu = false;
    fondoSpr.setVisible(false);
    mostrarUbicSpr.setVisible(false);
    ayudaSpr.setVisible(false);
    versionSpr.setVisible(false);
    salidaSpr.setVisible(false);
    mostrarUbicSprFijo.setVisible(false);
    ayudaSprFijo.setVisible(false);
    versionSprFijo.setVisible(false);
    salidaSprFijo.setVisible(false);
    drawScreen(g);
    xaman.mostrarUbicacion();
}

```

## Anexos

```
/*
 * Oculta el menú, todos los Sprites (íconos) e invoca al método
 * ayuda de la clase principal
 */
private void botonAyuda() {
    menu = false;
    fondoSpr.setVisible(false);
    mostrarUbicSpr.setVisible(false);
    ayudaSpr.setVisible(false);
    versionSpr.setVisible(false);
    salidaSpr.setVisible(false);
    mostrarUbicSprFijo.setVisible(false);
    ayudaSprFijo.setVisible(false);
    versionSprFijo.setVisible(false);
    salidaSprFijo.setVisible(false);
    drawScreen(g);
    xaman.ayuda();
}
```

```
/*
 * Oculta el menú, todos los Sprites (íconos) e invoca al método
 * versión de la clase principal
 */
private void botonVersion() {
    menu = false;
    fondoSpr.setVisible(false);
    mostrarUbicSpr.setVisible(false);
    ayudaSpr.setVisible(false);
    versionSpr.setVisible(false);
    salidaSpr.setVisible(false);
    mostrarUbicSprFijo.setVisible(false);
    ayudaSprFijo.setVisible(false);
    versionSprFijo.setVisible(false);
    salidaSprFijo.setVisible(false);
    drawScreen(g);
    xaman.version();
}
```

```
/*
 * Oculta el menú, todos los Sprites (íconos) e invoca al método
 * salir de la clase principal
 */
private void botonSalir() {
    menu = false;
    xaman.salir();
}
```

```

private void drawScreen(Graphics g) {
    LM.setViewWindow(0,0,ancho,alto);
    LM.paint(g,0,0);
    flushGraphics();
}
}

```

-----  
| ConexionBluetooth.java |  
-----

```

package XamanEk;

import java.io.*;
import javax.microedition.io.*;
import javax.bluetooth.*;
import java.util.Hashtable;
import javax.microedition.lcdui.*;
import javax.microedition.io.StreamConnection;

/**
 * @author Jorge A. Solano
 */
public class ConexionBluetooth implements CommandListener, DiscoveryListener {

    DiscoveryAgent agente = null; // Busca servicios
    Command buscarDisp, recibirDatos, salir, regresar, detener;
    List listaDispositivos = null;
    Form formTrabajo = null;
    Gauge objCargando = null;
    Alert errorD = null, errorC = null;
    Hashtable tablaDispositivos;
    XamanEk xaman;
    // Conexión btsp
    Form muestraDatosGPS;
    TextField datosGPS;
    StreamConnection conexion = null;
    MostrarMapa mm;

    /**
     * Constructor de la clase ConexionBluetooth
     */
    public ConexionBluetooth(XamanEk xaman) {
        mm = new MostrarMapa(this); // Se crea una instancia tipo MostrarMapa
        // Se inician los botones
        salir = new Command("Salir", Command.EXIT, 1); // Botón Salir
        buscarDisp = new Command("Buscar Dispositivos", Command.SCREEN, 1); // Botón buscar
        Disp
    }
}

```

```

recibirDatos = new Command("Recibir Datos", Command.SCREEN, 1); // Botón recibir datos
regresar = new Command("Regresar", Command.BACK, 1); // Botón regresar
detener = new Command("Detener", Command.STOP, 1); // Botón detener
// Alerta de error cuando no se encuentran dispositivos
errorD = new Alert("Error:", "No se encontró ningún dispositivo!", null, AlertType.ERROR);
// Alerta de error cuando no se pueden recibir los datos del GPS
errorC = new Alert("Error:", "No se pudo establecer la conexión con el dispositivo", null,
AlertType.ERROR);
// Inicia el indicador de "Cargando"
objCargando = new Gauge("", false, Gauge.INDEFINITE, Gauge.CONTINUOUS_RUNNING);
// Tabla de dispositivos encontrados (los que se ocupan para recibir datos)
tablaDispositivos = new Hashtable();
// Lista de dispositivos encontrados (los que se muestran)
listaDispositivos = new List("Dispositivos Encontrados", Choice.IMPLICIT);
// Se tiene acceso a los métodos de la clase principal
this.xaman = xaman;
// Comandos que utiliza las lista de dispositivos
listaDispositivos.addCommand(regresar);
listaDispositivos.addCommand(salir);
listaDispositivos.addCommand(buscarDisp);
listaDispositivos.addCommand(recibirDatos);
listaDispositivos.setCommandListener(this);
// Comandos que utiliza el formulario que se muestra la figura "cargando"
formTrabajo = new Form("Espere por favor");
formTrabajo.append(objCargando);
formTrabajo.addCommand(regresar);
formTrabajo.addCommand(recibirDatos);
formTrabajo.addCommand(detener);
formTrabajo.addCommand(salir);
formTrabajo.setCommandListener(this);
// Campo de texto para mostrar los datos NMEA recibidos
datosGPS = new TextField("", "", 500, TextField.ANY);
// Formulario donde se inserta la página
muestraDatosGPS = new Form("Datos del GPS");
muestraDatosGPS.append(datosGPS);
muestraDatosGPS.addCommand(buscarDisp);
muestraDatosGPS.addCommand(recibirDatos);
muestraDatosGPS.addCommand(regresar);
muestraDatosGPS.addCommand(salir);
muestraDatosGPS.setCommandListener(this);
}

```

```

/**
 * Método para buscar dispositivos Bluetooth disponibles
 */
public void buscarDispositivos() {
    listaDispositivos.deleteAll(); // Se eliminan dispositivos descubiertos anteriormente
    tablaDispositivos.clear(); // Se limpia la tabla de dispositivos descubiertos
    try {
        // Obtiene agente de búsqueda
        agente = LocalDevice.getLocalDevice().getDiscoveryAgent();
        // Empieza la pesquisa de dispositivos
        agente.startInquiry(DiscoveryAgent.GIAC, this);
        // Muestra la figura del objCargando al usuario
        objCargando.setLabel("Buscando Dispositivos");
        xaman.setPanAct(formTrabajo);
    } catch (Exception e) {
        msjErrorDispos(e);
    }
}

/**
 * Establece la conexión con el dispositivo GPS Y captura los datos
 */
public void conectarComm() throws IOException {
    // Selecciona el dispositivo de la tabla de dispositivos
    RemoteDevice device = (RemoteDevice)tablaDispositivos.get((new
Long(listaDispositivos.getSelectedIndex() + 1)));
    // La conexión se realiza utilizando la dirección bluetooth del dispositivo seleccionado
    String url = "btspp://" + device.getBluetoothAddress().toString() + ":1";
    // Muestra la figura del objCargando al usuario
    objCargando.setLabel("Intentando conectar con " + device.getFriendlyName(true).toString());
    xaman.setPanAct(formTrabajo);
    // Intenta abrir la conexión
    conexion = (StreamConnection)Connector.open(url, Connector.READ);
    // Flujo de datos recibidos
    InputStreamReader reader = new InputStreamReader(conexion.openInputStream());
    String datosRecibidos = ""; // Se utilizará para almacenar los datos capturados
    int recibeCaracter;
    int a = 0;
    boolean recibido = true;
    /* Se captura la trama NMEA $GPRMC*/
    while (recibido){
        recibeCaracter = reader.read();
        if ((char)recibeCaracter == 'R'){
            recibeCaracter = reader.read();
            if ((char)recibeCaracter == 'M'){
                recibeCaracter = reader.read();
                if ((char)recibeCaracter == 'C'){
                    while ((char)recibeCaracter != '*'){

```



```

        recibeCaracter = reader.read();
        datosRecibidos += (char)recibeCaracter;
        recibido = false;
    }
}
}
}
}
conexion.close(); // Se cierra la conexión serial
/* Obtiene las coordenadas (de ser posible) */
int advNaveg = 0, pos = 0;
String gradLat = "", minLat = "", segLat = "", gradLong = "", minLong = "";
String segLong = "", tipoLat = "", tipoLong = "";
char tramaRMC [] = datosRecibidos.toCharArray(); // Transforma string de datos a arreglo
for (int b = 0 ; b < tramaRMC.length ; b++){
    if (tramaRMC[b] == 'V'){ // Si se recibe V los datos no son correctos
        advNaveg = 1;
        b = tramaRMC.length;
    } else {
        if (tramaRMC[b] == 'A'){ // Si se recibe A los datos son correctos
            advNaveg = 2;
            pos = b+2; // guarda la posición donde empiezan las coordenadas
            b = tramaRMC.length;
        }
    }
}
if (advNaveg == 2){ // Si los datos son correctos
    /* Se obtiene Latitud: grados*/
    for (a = pos ; a < pos+2 ; a++){
        gradLat += tramaRMC[a];
    }
    /* Latitud: minutos*/
    for ( ; a < tramaRMC.length ; a++){
        if (tramaRMC[a] == '.'){
            pos = a+1;
            a = tramaRMC.length;
        } else {
            minLat += tramaRMC[a];
        }
    }
}
}

```

```

/* Latitud: segundos (en decimal)*/
for (a = pos ; a < tramaRMC.length ; a++){
    if (tramaRMC[a] == ','){
        pos = a+1;
        a = tramaRMC.length;
    } else {
        segLat += tramaRMC[a];
    }
}
/* Tipo de latitud: Norte (N) o Sur (S)*/
tipoLat += tramaRMC[pos];
/* Longitud: grados*/
pos = pos+2;
for (a = pos ; a < pos+3 ; a++){
    gradLong += tramaRMC[a];
}
/* Longitud minutos*/
for ( ; a < tramaRMC.length ; a++){
    if (tramaRMC[a] == '.'){
        pos = a+1;
        a = tramaRMC.length;
    } else {
        minLong += tramaRMC[a];
    }
}
/* Longitud segundos (en decimal)*/
for ( a = pos; a < tramaRMC.length ; a++){
    if (tramaRMC[a] == ','){
        pos = a+1;
        a = tramaRMC.length;
    } else {
        segLong += tramaRMC[a];
    }
}
/* Latitud: Este (E) u Oeste (W) */
tipoLong += tramaRMC[pos];
/* Se convierten los minutos decimales a segundos*/
// Transforma el string a número
char segLatArr[] = segLat.toCharArray();
char segLongArr[] = segLong.toCharArray();
segLat = "";
segLong = "";
segLat += '!';
segLong += '!';
for ( a = 0 ; a < 2 ; a++){
    segLat += segLatArr[a];
}

```

```

for ( a = 0 ; a < 2 ; a++){
    segLong += segLongArr[a];
}
// Convierte el string a double
double segLatNum = Double.valueOf(segLat).doubleValue();
double segLongNum = Double.valueOf(segLong).doubleValue();
// transforma los decimales a segundos
segLatNum = segLatNum*60;
segLongNum = segLongNum*60;
// transforma el número a String
segLat = String.valueOf(segLatNum);
segLong = String.valueOf(segLongNum);
// Quita los decimales de los segundos obtenidos
segLatArr = segLat.toCharArray();
segLongArr = segLong.toCharArray();
segLat = "";
segLong = "";
for ( a = 0; a < segLatArr.length ; a++){
    if (segLatArr[a] == '.'){
        a = segLatArr.length;
    } else {
        segLat += segLatArr[a];
    }
}
for ( a = 0; a < segLongArr.length ; a++){
    if (segLongArr[a] == '.'){
        a = segLongArr.length;
    } else {
        segLong += segLongArr[a];
    }
}
/* Busca si las coordenadas obtenidas están dentro de los mapas cargados en la BD*/
int hayMapa = buscarCoords(gradLat, minLat, segLat, tipoLat, gradLong, minLong, segLong,
tipoLong);
if (hayMapa == 1){ // Si existe el mapa se muestra
    xaman.setPanAct(mm);
    mm.iniciar();
    new Thread(mm).start();
} else { // Si no existe el mapa simplemente muestra las coordenadas
    datosGPS.setString("Latitud: " + gradLat + "° " + minLat + "' " + segLat + "" " + tipoLat + " "
+
        "\n" + "Longitud: " + gradLong + "° " + minLong + "' " + segLong + "" " + tipoLong + ""
+
        "\n\n No hay mapas disponibles para esta posición");
    xaman.setPanAct(muestraDatosGPS);
}
} else {

```

```

    if (advNaveg == 1){ // Si los datos no son correctos
        datosGPS.setString("Advertencia de navegación recibida por el GPS.\n\n" +
            "El GPS no contiene coordenadas válidas. Cambie de " +
            "posición el dispositivo e inténtelo nuevamente.");
        xaman.setPanAct(muestraDatosGPS);
    }
}
}

/* Busca las coordenadas en la Base de Datos, si existen regresa 1, de lo contrario
 * regresa 0
 */
public int buscarCoords (String gradLat, String minLat, String segLat, String tipoLat,
    String gradLong, String minLong, String segLong, String tipoLong){
    int latitud, longitud, hayMapa = 0;
    // Convierte los datos de la latitud a enteros
    int gradLatNum = Integer.parseInt(gradLat);
    int minLatNum = Integer.parseInt(minLat);
    int segLatNum = Integer.parseInt(segLat);
    // Agrupa los datos de la latitud en un solo número: GGMMSS
    latitud = gradLatNum * 10000 + minLatNum * 100 + segLatNum;
    // Convierte los datos de la longitud a enteros
    int gradLongNum = Integer.parseInt(gradLong);
    int minLongNum = Integer.parseInt(minLong);
    int segLongNum = Integer.parseInt(segLong);
    // Agrupa los datos de la longitud en un solo número: GGGMMSS
    longitud = gradLongNum * 10000 + minLongNum * 100 + segLongNum;
    // Carga las coordenadas de los mapas que se poseen en la BD
    BDCargaCoords cargaCoords = new BDCargaCoords();
    // Verifica que las coordenadas recibidas estén entre algún valor existente en la BD
    BDNodoC1 coords = cargaCoords.buscarCoords(tipoLat, tipoLong, latitud, longitud);
    if (coords == null){ // No existen coordenadas ni mapa
        hayMapa = 0;
    } else { // Existen coordenadas y mapa
        int posX, posY, anchoX, anchoY, latSI[], latID[], longSI[], longID[];
        int latX [] = {gradLatNum, minLatNum, segLatNum};
        int longX [] = {gradLongNum, minLongNum, segLongNum};
        // Se separan los valores de latitud y longitud (SI e ID)
        latSI = obtenerGradsMinSeg(coords.latSI+ "");
        latID = obtenerGradsMinSeg(coords.latID+ "");
        longSI = obtenerGradsMinSeg(coords.longSI+ "");
        longID = obtenerGradsMinSeg(coords.longID+ "");
        // Obtiene la distancia entre longitud SI y la longitud del punto recibido
        posX = obtenerDist(longSI, longX);
        // Obtiene la distancia entre longitud SI y la longitud ID del mapa
        anchoX = obtenerDist(longSI, longID);
        // Obtiene la distancia entre latitud SI y la latitud del punto recibido
        posY = obtenerDist(latSI, latX);
    }
}

```

```

// Obtiene la distancia entre latitud SI y la latitud ID del mapa
anchoY = obtenerDist(latSI,latID);
// Obtiene el nombre del mapa (concatenación entre latSI longSI - latID longID
String coordStr = coords.coordenadas();
// Carga el mapa a mostrar (clase MostrarMapa)
mm.cargarMapa(coordStr, posX, posY, anchoX, anchoY);
// Variable que indica que el mapa se encontro y puede ser mostrado
hayMapa = 1;
}
return hayMapa;
}

```

```

private int[] obtenerGradsMinSeg(String pal){
char posicion [] = pal.toCharArray();
int nums[] = new int[3];
int i;
String a = "";
// Obtiene los segundos
for (i = posicion.length-2 ; i <= posicion.length-1 ; i++ ){
a += posicion[i];
}
nums[2] = Integer.parseInt(a);
a = "";
// Obtiene los minutos
for (i = posicion.length-4 ; i <= posicion.length-3 ; i++ ){
a += posicion[i];
}
nums[1] = Integer.parseInt(a);
a = "";
// Obtiene los grados
for (i = 0 ; i <= posicion.length-5 ; i++ ){
a += posicion[i];
}
nums[0] = Integer.parseInt(a);
return nums;
}

```

```

private int obtenerDist(int [] mayor, int [] menor){
int res [] = {0,0,0}, a = 0, dist = 0, b = 0;
// Resta de grados
res [a] = mayor[a]-menor[a];
a++;

```

```

// Resta de minutos
if (res[a-1] > 0) {
    res[a] = (60 - menor[a]) + mayor[a];
    if (res[a-1] > 1) {
        b = res[a]+((res[a-1]-1)*60);
    } else {
        b = res[a];
    }
} else {
    res[a] = mayor[a] - menor[a];
}
a++;
// Resta de segundos
if (res[a-1] > 0) {
    res[a] = (60 - menor[a]) + mayor[a];
    if (res[a-1] > 1) {
        b = b + res[a]+((res[a-1]-1)*60);
    } else {
        b = b + res[a];
    }
} else {
    res[a] = mayor[a] - menor[a];
    b = b + res[a];
}
dist = b;
return dist;
}

// Como se van descubriendo dispositivos se agregan a la lista
public void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod) {
    try {
        // Se agregan los nombres de los dispositivos
        listaDispositivos.append(btDevice.getFriendlyName(false), null);
        // Se agrega el dispositivo a la tabla hash
        tablaDispositivos.put(new Long(listaDispositivos.size()), btDevice);
    } catch(Exception e) {
    }
}

/*
 * Este método se tiene que declarar por utilizar DiscoveryListener
 */
public void servicesDiscovered(int transID, ServiceRecord[] servRecord) {
}

```

## Anexos

```
/*
 * Este método se tiene que declarar por utilizar DiscoveryListener
 */
public void serviceSearchCompleted(int transID, int respCode) {
}

public void inquiryCompleted(int discType) {
    // Cuando se completa la búsqueda de dispositivos
    if(listaDispositivos.size() == 0) { // Si no se encontraron dispositivos
        xaman.setPanAct(errorD); // se muestra una alerta
    } else { // De lo contrario
        xaman.setPanAct(listaDispositivos); // muestra la lista de dispos encontrados
    }
}

// ***** Manejo de errores *****
/*
 * Error cuando no se encontraron dispositivos al alcance
 */
private void msjErrorDispos(Exception e) {
    xaman.setPanAct(new Alert("No se encontraron dispositivo cercanos. Inténtelo de nuevo." +
        "\n\n", e.getMessage(), null, AlertType.ERROR));
    //e.printStackTrace();
}

// ***** Manejo de comandos seleccionados *****
public void commandAction(Command cmd, Displayable disp) {
    if(cmd == salir) { // comando salir
        xaman.salir(); // Destruye la aplicación
    } else {
        if(cmd == recibirDatos) { // comando recibirDatos
            new Thread(new Runnable() { // Ejecuta un hilo
                public void run() {
                    try {
                        conectarComm(); // Realiza la conexión con el GPS
                    } catch (Exception e) {
                        errorC.setString("Error: " + e.getMessage());
                        xaman.setPanAct(errorC);
                    }
                }
            }).start();
        } else {

```





```

public MostrarMapa (ConexionBluetooth cb){
    super(true);
    salir = new Command("Salir", Command.EXIT, 0);
    recibirDatos = new Command("Recibir Datos", Command.SCREEN, 1);
    regresar = new Command("Regresar", Command.BACK, 0);
    this.cb = cb;

    this.addCommand(recibirDatos);
    this.addCommand(regresar);
    this.addCommand(salir);
    this.setCommandListener(this);
}

public void cargarMapa (String mapa, int posX, int posY, int anchoX, int anchoY){
    try {
        // Se carga el mapa a mostrar
        mapalng = Image.createImage("/Mapas/" + mapa + ".png");
        // Se carga pin que muestra ubicación
        pinlmg = Image.createImage("/Imgs/pin.png");
        // Redimensiona el pin
        pinlmg = cb.xaman.ri.redPin(pinlmg);
        // Medidas del mapa
        anchoMapa = mapalng.getWidth();
        altoMapa = mapalng.getHeight();
        // Ubicación del pin
        xPin = (posX*mapalng.getWidth())/anchoX;
        yPin = (posY*mapalng.getHeight())/anchoY;
        // Ubicación del mapa
        ubicarMapa();
    } catch (Exception e){
        mapalng = pinlmg = null;
        anchoMapa = altoMapa = xMapa = yMapa = xPin = yPin = 0;
        throw new RuntimeException ("No se puede cargar la imagen : " + e);
    }
    this.LM = cb.xaman.getLM();
    g = getGraphics();
    tmm = new Thread(this);
    // Crea los sprites
    mapaSpr = new Sprite(mapalng, mapalng.getWidth(), mapalng.getHeight());
    pinSpr = new Sprite(pinlmg, pinlmg.getWidth(), pinlmg.getHeight());
    // Agrega los sprites al manejo de capas
    LM.append(pinSpr);
    LM.append(mapaSpr);
    // Posiciona el mapa y el pin
    mapaSpr.setRefPixelPosition(xMapa,yMapa);
    pinSpr.setRefPixelPosition(xPin, yPin-pinlmg.getHeight());
}

```

```

public void iniciar() {
    mostrar = true;
    mapaSpr.setVisible(true);
    pinSpr.setVisible(true);
}

private void ubicarMapa(){
    int mov = 0;
    xMapa = yMapa = 0;
    if (xPin > mapalmg.getWidth()/2){
        if (yPin > mapalmg.getHeight()/2){ // Ubicación en el tercer cuadrante
            // Si la mitad del mapa (horizontal) es menor que el tamaño de la pantalla
            if (mapalmg.getWidth()/2 < getWidth()){
                mov = getWidth() - (mapalmg.getWidth()/2);
                xMapa = xMapa - (mapalmg.getWidth()/2) + mov;
                xPin = xPin - (mapalmg.getWidth()/2) + mov;
            } else {
                xMapa = xMapa - (mapalmg.getWidth()/2);
                xPin = xPin - (mapalmg.getWidth()/2);
            }
        }
        // Si la mitad del mapa (vertical) es menor que el tamaño de la pantalla
        if (mapalmg.getHeight()/2 < getHeight()){
            mov = getHeight() - (mapalmg.getHeight()/2);
            yMapa = yMapa - (mapalmg.getHeight()/2) + mov;
            yPin = yPin - (mapalmg.getHeight()/2) + mov;
        } else {
            yMapa = yMapa - (mapalmg.getHeight()/2);
            yPin = yPin - (mapalmg.getHeight()/2);
        }
    } else { // Ubicación en el segundo cuadrante
        // Si la mitad del mapa (horizontal) es menor que el tamaño de la pantalla
        if (mapalmg.getWidth()/2 < getWidth()){
            mov = getWidth() - (mapalmg.getWidth()/2);
            xMapa = xMapa - (mapalmg.getWidth()/2) + mov;
            xPin = xPin - (mapalmg.getWidth()/2) + mov;
        } else {
            xMapa = xMapa - (mapalmg.getWidth()/2);
            xPin = xPin - (mapalmg.getWidth()/2);
        }
    }
} else {
    if (yPin > mapalmg.getHeight()/2){ // Ubicación en el cuarto cuadrante
        // Si la mitad del mapa (vertical) es menor que el tamaño de la pantalla
        if (mapalmg.getHeight()/2 < getHeight()){
            mov = getHeight() - (mapalmg.getHeight()/2);
            yMapa = yMapa - (mapalmg.getHeight()/2) + mov;
            yPin = yPin - (mapalmg.getHeight()/2) + mov;
        } else {
    }
}
}

```

## Anexos

```
        yMapa = yMapa - (mapalng.getHeight()/2);
        yPin = yPin - (mapalng.getHeight()/2);
    }
} else { // El punto está en el primer cuadrante
    xMapa = 0;
    yMapa = 0;
}
}
}

public void stop() {}

public void run() {
    while (mostrar) {
        drawScreen(g);
        teclasPresionadas(); // se "escuchan" las teclas presionadas
        try {
            Thread.sleep(vel); // Velocidad con la que se escucha el teclado
        } catch (InterruptedException ie) {
        }
    }
}

private void teclasPresionadas(){
    int estadoTecla = getKeyStates();
    if (estadoTecla != 0 ) {
        switch (estadoTecla) {
            case RIGHT_PRESSED:
                if(xMapa < 0) {
                    xMapa += 10;
                    xPin += 10;
                }
                break;
            case LEFT_PRESSED: // bien
                if(xMapa > getWidth()-anchoMapa) {
                    xMapa -= 10;
                    xPin -= 10;
                }
                break;
            case UP_PRESSED: // bien
                if(yMapa > getHeight()-altoMapa) {
                    yMapa -= 10;
                    yPin -= 10;
                }
                break;
        }
    }
}
```

```

    case DOWN_PRESSED:
        if(yMapa < 0) {
            yMapa += 10;
            yPin += 10;
        }
        break;
    default:
        return;
    }
}
mapaSpr.setRefPixelPosition(xMapa, yMapa);
pinSpr.setRefPixelPosition(xPin, yPin);
}

private void limpiarPantalla() {
    mostrar = false;
    mapaSpr.setVisible(false);
    pinSpr.setVisible(false);
    drawScreen(g);
}

private void botonSalir() {
    mostrar = false;
    cb.xaman.salir();
}

private void drawScreen(Graphics g) {
    LM.setViewWindow(0,0,getWidth(),getHeight());
    LM.paint(g,0,0);
    flushGraphics();
}

public void commandAction(Command c, Displayable p) {
    if(c == salir){
        botonSalir();
    } else {
        if (c == regresar){
            limpiarPantalla();
            cb.xaman.setPanAct(cb.listaDispositivos);
        } else {

```

## Anexos

```
        if (c == recibirDatos){
            limpiarPantalla();
            new Thread(new Runnable() { // Ejecuta un hilo
                public void run() {
                    try {
                        cb.conectarComm(); // Realiza la conexión con el GPS
                    } catch (Exception e) {
                        cb.errorC.setString("Error: " + e.getMessage());
                        cb.xaman.setPanAct(cb.errorC);
                    }
                }
            }).start();
        }
    }
}
}
```

-----  
| BDNodo.java |  
-----

```
package XamanEk;
```

```
// Cuadrante 1
```

```
public class BDNodoC1 {
    int latSI, longSI, latID, longID;
    public BDNodoC1 leftChild; // Nodo hijo izquierdo
    public BDNodoC1 rightChild; // Nodo hijo derecho

    public BDNodoC1 (int latSI, int longSI, int latID, int longID) {
        this.latSI = latSI;
        this.longSI = longSI;
        this.latID = latID;
        this.longID = longID;
        leftChild = null;
        rightChild = null;
    }

    public BDNodoC1() {
    }
}
```

```
// Esta cadena de caracteres forma el nombre del mapa
public String coordenadas(){
    String coords = latSI + "" + longSI + "-" + latID + "" + longID + "";
    coords.trim();
    return coords;
}
} // Termina clase BDNodo
```

```
-----
| BDCargaCoords.java |
-----
```

```
package XamanEk;

public class BDCargaCoords {
    BDArbolC1 coords = new BDArbolC1();

    public BDCargaCoords (){
        cargarCoords();
    }

    public void cargarCoords() {
        /* Aquí se cargan las coordenadas de los mapas. Los mapas (imágenes) se insertan en la
        carpeta Maps. Se ingresan las coordenadas de la esquina superior izquierda y a la
        esquina inferior derecha del mapa en el siguiente orden:

        Latitud Superior Izquierda GGMMSS
        Longitud Superior Izquierda GGGMMSS
        Latitud Inferior Derecha GGMMSS
        Longitud Inferior Derecha GGGMMSS

        Donde GG y GGG son grados, MM son minutos y SS segundos
        */
        coords.insertar(192008, 991148, 191923, 991050); // Coords CU
        coords.insertar(191823, 990930, 191748, 990849); // Coords Edio. Azteca
    }

    public BDNodoC1 buscarCoords (String tipoLat, String tipoLong, int latitud, int longitud){
        BDNodoC1 buscar = null;
        /* Si la latitud es norte y la longitud es oeste*/
        if (tipoLat.equals("N") && tipoLong.equals("W")){ // Está en el cuadrante 1
            buscar = coords.buscar(latitud,longitud); // Busca las coords en al BD
            return buscar; // Regresaa el nodo encontrado
        } else {
```

## Anexos

```
    if (tipoLat.equals("N") && tipoLong.equals("E")){ // Cuadrante 2
        return buscar;
    } else {
        if (tipoLat.equals("S") && tipoLong.equals("W")){ // Cuadrante 3
            return buscar;
        } else {
            if (tipoLat.equals("S") && tipoLong.equals("E")){ // Cuadrante 4
                return buscar;
            }
        }
    }
}
return buscar;
}
```

-----  
| BDArbolC1.java |  
-----

```
package XamanEk;
```

```
public class BDArbolC1 {
    private BDNodoC1 raiz; // Primer nodo del árbol (nodo raíz)

    public BDArbolC1 () { // Constructor
        raiz = null; // No hay nodos en el árbol
    }

    public BDNodoC1 getRaiz() {
        return raiz;
    }

    public void insertar(int latSI, int longSI, int latID, int longID) {
        // Inserta los parámetros en el nodo
        BDNodoC1 nuevoNodo = new BDNodoC1(latSI, longSI, latID, longID);
        if (raiz == null) { // Si no hay nodo raíz
            raiz = nuevoNodo; // el nuevo nodo es el nodo raíz
        } else { // Si hay nodo raíz
            BDNodoC1 current = raiz; // Empieza en nodo raíz
            BDNodoC1 parent;
```

```

while(true) {          // Existe internamente
    parent = current;
    // Si el mapa que se inserta es menor a uno existente, se pone a la izquierda
    if (latSI <= current.latSI && latID >= current.latID && longSI <= current.longSI && longID >=
current.longID) {     // va a la izquierda
        current = current.leftChild;
        if (current == null) {    // Si llega al final de la línea
            parent.leftChild = nuevoNodo; // inserta a la izquierda
            return;
        }
    } else {           // si el mapa es mayor va a la derecha
        current = current.rightChild;
        if (current == null) {    // si llega al final de la línea
            parent.rightChild = nuevoNodo; // inserta a la derecha
            return;
        }
    }
}
}
} // Fin método insertar

public BDNodeC1 buscar(int latitud, int longitud){
    if (raiz == null) { // Si el nodo raiz es nulo, no hay coordenadas en la base de datos
        System.out.println("No hay coordenadas en la base de datos");
        return null;
    } else {
        BDNodeC1 current=raiz;
        while (current != null) { // Mientras se llega al final del árbol
            // Busca si las coordenadas del punto están entre las coordenadas del mapa
            if (latitud >= current.latSI || latitud <= current.latID || longitud >= current.longSI ||
longitud <= current.longID) {
                current=current.rightChild;
            } else {
                // Si las coordenadas del punto están dentro del mapa regresa la referencia
                if (latitud <= current.latSI && latitud >= current.latID && longitud <= current.longSI &&
longitud >= current.longID) {
                    return current;
                } else {
                    current=current.leftChild;
                }
            }
        }
        return null;
    }
}
} // end class BDArbol

```



-----  
 | Ayuda.java |  
 -----

```

package XamanEk;

import javax.microedition.lcdui.*;

/**
 * @author Jorge A. Solano
 */
public class Ayuda implements CommandListener {

    Command regresar, salir;
    Form datos;
    StringItem etiqueta;
    XamanEk xaman;
    Image ayudalmg;
    String mensaje = "es una aplicación que intenta conectar este dispositivo " +
        "con un GPS que posea Bluetooth y que esté dentro del alcance. " +
        "Por lo que es necesario tener un GPS cerca y encendido." +
        "\n\nPara obtener la posición actual hay que presionar el botón \"Ubicación\". " +
        "Al presionar este botón, la aplicación buscará dispositivos Bluetooth " +
        "cercanos. Si no se tiene activado el Bluetooth, la aplicación pedirá " +
        "permiso para activarlo. En este caso puede que la búsqueda de dispositivos " +
        "falle, hay que volver a realizarla. En caso de no encontrar dispositivos se " +
        "emite una alerta de error. Hay que verificar que el bluetooth del dispositivo " +
        "GPS se encuentre activado." +
        "\n\nSi la búsqueda encontró el dispositivo GPS hay que posicionarse sobre el " +
        "nombre y seleccionar el botón \"Recibir Datos\". La aplicación intenta conectarse " +
        "con el GPS para obtener las coordenadas. Si la conexión no se puede establecer se " +
        "emite una alerta de error, hay que verificar que seleccionamos el dispositivo " +
        "adecuado y que éste tiene encendido su Bluetooth. " +
        "\n\nSi la conexión se lleva a cabo correctamente, se lee la información recibida. " +
        "Si el GPS no ha podido establecer conexión con mínimo tres satélites la información " +
        "no será correcta y la aplicación enviará una alerta de error debido a que los datos " +
        "leídos no contienen coordenadas, en este caso hay que posicionar el GPS en " +
        "otro sitio y volver a presionar el botón \"Recibir Datos\"." +
        "\n\nSi los datos recibidos son correctos se procede a buscar el mapa. Si existe un " +
        "mapa correspondiente a las coordenadas obtenidas por el GPS se muestra en pantalla " +
        "y se pone un pin amarillo en la posición recibida. Si las coordenadas recibidas no " +
        "coinciden con los mapas que posee la aplicación, se muestran las coordentas recibidas " +
        "(Latitud y Longitud)." +
        "\n\n Es posible desplazar el mapa con las teclas de desplazamiento o con las teclas " +
        "numéricas.";

```

```

public Ayuda(XamanEk xaman){
    regresar = new Command("Regresar", Command.BACK, 0);
    salir = new Command("Salir", Command.EXIT, 0);
    etiqueta = new StringItem("XamanEk", mensaje, StringItem.PLAIN);
    try {
        ayudaImg = Image.createImage("/Imgs/m_ayuda.png");
    } catch (Exception e){
        throw new RuntimeException ("No se puede cargar la imagen de Ayuda: " + e);
    }

    this.xaman = xaman;

    datos = new Form("Ayuda");
    datos.append(ayudaImg);
    datos.append(etiqueta);
    datos.addCommand(regresar);
    datos.addCommand(salir);
    datos.setCommandListener(this);
}

public void commandAction(Command c, Displayable d) {
    if(c == salir){
        xaman.salir();
    } else {
        if (c == regresar){
            xaman.regresar();
        }
    }
}
}
}

```

-----  
| Versión.java |  
-----

```

package XamanEk;

import javax.microedition.lcdui.*;

/**
 * @author Jorge A. Solano
 */

```

## Anexos

```
public class Version implements CommandListener {

    Command regresar, salir;
    Form datos;
    XamanEk xaman;
    Image versionImg;

    public Version(XamanEk xaman){
        regresar = new Command("Regresar", Command.BACK, 0);
        salir = new Command("Salir", Command.EXIT, 0);
        datos = new Form("Versión");
        try {
            versionImg = Image.createImage("/Imgs/v_xaman.png");
            versionImg = xaman.ri.redImgForm(versionImg, datos.getWidth(), datos.getHeight());
        } catch (Exception e){
            throw new RuntimeException ("No se puede cargar la imagen de Ayuda: " + e);
        }

        this.xaman = xaman;

        datos.append(versionImg);
        datos.addCommand(regresar);
        datos.addCommand(salir);
        datos.setCommandListener(this);
    }

    public void commandAction(Command c, Displayable d) {
        if(c == salir){
            xaman.salir();
        } else {
            if (c == regresar){
                xaman.regresar();
            }
        }
    }
}
```

-----  
| RedibujarImgs.java |  
-----

```
package XamanEk;

import javax.microedition.lcdui.*;

/**
 * @author Jorge A. Solano
 */
```

```

public class RedibujarImgs extends Canvas {

    protected void paint(Graphics g) {
    }

    /**
     * Este método redimensiona una imagen al tamaño de la pantalla del dispositivo
     * @param img es la imagen a ser redimensionada
     * @return La imagen redimensionada
     */
    public Image redImg(Image img) {
        int anchoImg = img.getWidth();        // Se obtiene ancho imagen
        int altoImg = img.getHeight();        // Se obtiene alto imagen
        // Se crea imagen con el ancho y alto de la pantalla
        Image tmp = Image.createImage(getWidth(), altoImg);
        Graphics g = tmp.getGraphics();        // Regresa el objeto Graphics
        int razon = (anchoImg << 16) / getWidth(); // Desplaza los bits de anchoImg 16 veces
        int pos = razon/2;
        // Redimension Horizontal
        for (int i = 0 ; i < getWidth() ; i++) {
            g.setClip(i, 0, 1, altoImg);
            g.drawImage(img, i - (pos >> 16), 0, Graphics.LEFT | Graphics.TOP);
            pos += razon;
        }
        Image resizedImage = Image.createImage(getWidth(), getHeight());
        g = resizedImage.getGraphics();
        razon = (altoImg << 16) / getHeight();
        pos = razon/2;
        // Redimension Vertical
        for (int j = 0 ; j < getHeight() ; j++) {
            g.setClip(0, j, getWidth(), 1);
            g.drawImage(tmp, 0, j - (pos >> 16), Graphics.LEFT | Graphics.TOP);
            pos += razon;
        }
        return resizedImage;
    }

    /**
     * Este método redimensiona una imagen al tamaño de la alerta del dispositivo
     * @param img es la imagen a ser redimensionada
     * @param anchoA es el ancho de la alerta según el dispositivo
     * @param altoA es el alto de la alerta según el dispositivo
     * @return La imagen redimensionada
     */
}

```

```

public Image redImgAlerta(Image img, int anchoA, int altoA) {
    int anchoImg = img.getWidth();        // Se obtiene ancho imagen
    int altoImg = img.getHeight();        // Se obtiene alto imagen
    int anchoAlerta = anchoA-10;          // Ancho de la alerta
    int altoAlerta = altoA-10;            // Alto de la alerta
    // Se crea imagen con el ancho y alto de la pantalla
    Image tmp = Image.createImage(anchoAlerta, altoImg);
    Graphics g = tmp.getGraphics();        // Regresa el objeto Graphics
    int razon = (anchoImg << 16) / anchoAlerta; // Desplaza los bits de anchoImg 16 veces
    int pos = razon/2;
    // Redimension Horizontal
    for (int i = 0 ; i < anchoAlerta ; i++) {
        g.setClip(i, 0, 1, altoImg);
        g.drawImage(img, i - (pos >> 16), 0, Graphics.LEFT | Graphics.TOP);
        pos += razon;
    }
    Image resizedImage = Image.createImage(anchoAlerta, altoAlerta);
    g = resizedImage.getGraphics();
    razon = (altoImg << 16) / altoAlerta;
    pos = razon/2;
    // Redimension Vertical
    for (int j = 0 ; j < altoAlerta ; j++) {
        g.setClip(0, j, anchoAlerta, 1);
        g.drawImage(tmp, 0, j - (pos >> 16), Graphics.LEFT | Graphics.TOP);
        pos += razon;
    }
    return resizedImage;
}

/**
 * Este método reduce una imagen dada para formar los íconos del menú
 * según el tipo: 0 íconos grandes y móviles, 1 íconos pequeños y fijos
 * @param img es la imagen a ser redimensionada
 * @return La imagen redimensionada
 */
public Image redIcono(Image img, int tipo) {
    int anchoImg = img.getWidth();        // Se obtiene ancho imagen
    int altoImg = img.getHeight();        // Se obtiene alto imagen
    int ancho, alto;
    if (tipo == 0){
        ancho = getWidth()/2;            // Medidas del Ícono: ancho
        alto = getHeight()/8;            // alto
    } else {
        ancho = getWidth()/3;            // Medidas del Ícono: ancho
        alto = getHeight()/10;           // alto
    }
}

```

```

// Se crea imagen con el ancho requerido para el Ã-cono
Image tmp = Image.createImage(ancho, altoImg);
Graphics g = tmp.getGraphics(); // Regresa el objeto Graphics
int razon = (anchoImg << 16) / ancho; // Desplaza los bits de anchoImg 16 veces
int pos = razon/2;
// Redimension Horizontal
for (int i = 0 ; i < ancho ; i++) {
    g.setClip(i, 0, 1, altoImg);
    g.drawImage(img, i - (pos >> 16), 0, Graphics.LEFT | Graphics.TOP);
    pos += razon;
}
// Se crea imagen con el ancho y alto requerido para el Ã-cono
Image resizedImage = Image.createImage(ancho, alto);
g = resizedImage.getGraphics();
razon = (altoImg << 16) / alto;
pos = razon/2;
// Redimension Vertical
for (int j = 0 ; j < alto ; j++) {
    g.setClip(0, j, ancho, 1);
    g.drawImage(tmp, 0, j - (pos >> 16), Graphics.LEFT | Graphics.TOP);
    pos += razon;
}
return resizedImage;
}
}

/**
 * Este mÃ©todo redimensiona una imagen al tamaÃ±o de un formulario
 * @param img es la imagen a ser redimensionada
 * @param anchoF es el ancho de la alerta segÃ¼n el dispositivo
 * @param altoF es el alto de la alerta segÃ¼n el dispositivo
 * @return La imagen redimensionada
 */
public Image redImgForm(Image img, int anchoF, int altoF) {
    int anchoImg = img.getWidth(); // Se obtiene ancho imagen
    int altoImg = img.getHeight(); // Se obtiene alto imagen
    int anchoForm = anchoF; // Ancho del formulario
    int altoForm = altoF-5; // Alto del formulario
    // Se crea imagen con el ancho y alto de la pantalla
    Image tmp = Image.createImage(anchoForm, altoImg);
    Graphics g = tmp.getGraphics(); // Regresa el objeto Graphics
    int razon = (anchoImg << 16) / anchoForm; // Desplaza los bits de anchoImg 16 veces
    int pos = razon/2;
    // Redimension Horizontal
    for (int i = 0 ; i < anchoForm ; i++) {
        g.setClip(i, 0, 1, altoImg);
        g.drawImage(img, i - (pos >> 16), 0, Graphics.LEFT | Graphics.TOP);
        pos += razon;
    }
}

```

```

Image resizedImage = Image.createImage(anchoForm, altoForm);
g = resizedImage.getGraphics();
razon = (altoImg << 16) / altoForm;
pos = razon/2;
// Redimension Vertical
for (int j = 0 ; j < altoForm ; j++) {
    g.setClip(0, j, anchoForm, 1);
    g.drawImage(tmp, 0, j - (pos >> 16), Graphics.LEFT | Graphics.TOP);
    pos += razon;
}
return resizedImage;
}

public Image redPin(Image img) {
    int anchoImg = img.getWidth();        // Se obtiene ancho imagen
    int altoImg = img.getHeight();       // Se obtiene alto imagen
    int anchoPin = (5*getWidth())/100;   // Ancho del pin
    int altoPin = (5*getHeight())/100;   // Alto del pin
    // Se crea imagen con el ancho y alto de la pantalla
    Image tmp = Image.createImage(anchoPin, altoPin);
    Graphics g = tmp.getGraphics();      // Regresa el objeto Graphics
    int razon = (anchoImg << 16) / anchoPin; // Desplaza los bits de anchoImg 16 veces
    int pos = razon/2;
    // Redimension Horizontal
    for (int i = 0 ; i < anchoPin ; i++) {
        g.setClip(i, 0, 1, altoPin);
        g.drawImage(img, i - (pos >> 16), 0, Graphics.LEFT | Graphics.TOP);
        pos += razon;
    }
    Image resizedImage = Image.createImage(anchoPin, altoPin);
    g = resizedImage.getGraphics();
    razon = (altoImg << 16) / altoPin;
    pos = razon/2;
    // Redimension Vertical
    for (int j = 0 ; j < altoPin ; j++) {
        g.setClip(0, j, anchoPin, 1);
        g.drawImage(tmp, 0, j - (pos >> 16), Graphics.LEFT | Graphics.TOP);
        pos += razon;
    }
    return resizedImage;
}
}
}

```