ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

"Utilización de data almacenada con posicionamiento GPS en pen drives o en memorias MMC/SD para la elaboración de mapas que permitan interactuar con GOOGLE EARTH para la localización de objetivos y definición de trayectorias"

TESINA DE SEMINARIO

Previa la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentado por:

Michael David Peña Zhindon

Diego Bonilla Almendáriz

GUAYAQUIL – ECUADOR

AÑO 2010

AGRADECIMIENTO

A cada uno de los profesores que me han tocado a lo largo de mi vida estudiantil, tanto primaria, secundaria y universitaria. Pues toda la teoría y sus conocimientos me han ayudado para lograr aportar con esta tecnología innovadora.

DEDICATORIA

A mi madre, por su gran ayuda en los momentos difíciles.

TRIBUNAL DE SUSTENTACIÓN

Ing. Carlos Valdivieso

Profesor de Seminario de Graduación

Ing. Hugo Villavicencio V.

Delegado del Decano

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta tesina, nos corresponde exclusivamente; y el patrimonio intelectual del mismo a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL".

(Reglamento de exámenes y títulos profesionales de la ESPOL)

_____

Michael David Peña Zhindón

_____

Diego Bonilla Almendáriz

RESUMEN

El objetivo de este proyecto es la realización de un software que permita interactuar con GOOGLE EARTH, dado que para lograr su funcionamiento es necesario de las coordenadas enviadas en una trama GPLL, provenientes del módulo GPS.

Este software fue desarrollado con la ayuda de JAVA, implementado en NETBEANS. Este programa al recibir las tramas del GPS llamadas NMEA; las convierte en un formato compatible con GOOGLE EARTH, GOOGLE MAPS llamado KML.

A lo largo de la explicación en la elaboración de nuestro programa, se van a detallar las herramientas utilizadas en NETBEANS, que nos fue de gran ayuda para lograr el principal objetivo del proyecto, que es la gráfica de rutas. También se explicará todo concerniente a las tramas NMEA, como también la estructura KML.

Por último se adjuntará un pequeño tutorial acerca de la utilización de códigos utilizado en nuestro software.

CONTENIDO

Índice de figuras y tablas

CAPÍTULO 1

1.- DESCRIPCIÓN GENERAL DEL PROYECTO

1.1.- ANTECEDENTES

Como el propósito de este trabajo es el de realizar el trazado de trayectorias usando el programa de disponibilidad gratuita GOOGLE EARTH cabe mencionar que dicho programa fue desarrollado en un principio por Keyhole. Fue una empresa pionera en el desarrollo de software especializado de visualización de datos geoespaciales y adquirida por Google en el 2004. El 21 de mayo de 2005 Keyhole pasó a llamarse GOOGLE EARTH. En este programa se incorpora GOOGLE MAPS que también consigue información sobre rutas. Es por esto, que se llevó a cabo el desarrollo de GOOGLE EARTH. Esta

gran herramienta permite visualizar relieves, edificios en 3D, cualquier parte de la tierra.

Pero aquí es donde entra el funcionamiento del GPS, que es un sistema satelital de posicionamiento que cada segundo envía tramas con coordenadas de puntos para que por medio de nuestra aplicación pueda graficarse en GOOGLE EARTH.

Este proyecto, recoge una gran cantidad de puntos con sus respectivas tramas por puerto USB o MEMORIA SD. Es decir todos esos puntos serán convertidos en formato KML para poder interactuar con GOOGLE EARTH.

Actualmente, algunas empresas están aplicando esta tecnología de trazado de rutas, empleados por ejemplo en un automóvil. Se debe tener en cuenta que el GPS funciona en una red de 32 satélites de los cuales 28 están operativos y 4 son de respaldo. Todos estos satélites operan sobre el globo terrestre a una distancia de 20200km

y contienen trayectorias sincronizadas de toda la superficie de la tierra.

Cuando se desea determinar la posición, el receptor que se utiliza para ello localiza automáticamente como mínimo tres satélites de la red, de los que recibe señales indicando la identificación y la hora del reloj de cada uno de ellos. Con base a estas señales, el dispositivo sincroniza el reloj del GPS y calcula el tiempo que tardan en llegar las señales al equipo, y de este modo mide la distancia al satélite mediante triangulación, la cual se basa en determinar la distancia de cada satélite respecto al punto de medición. Conocidas las distancias, se determina fácilmente la propia posición relativa respecto a los tres satélites. Conociendo además las coordenadas o posición de cada uno de ellos por la señal que emiten, se obtiene la posición absoluta o coordenada reales del punto de medición.

El GPS está evolucionando hacia un sistema más sólido (GPS III), con una mayor disponibilidad y que reduzca la complejidad de las aumentaciones GPS. El GPS III da un nuevo sistema de navegación (NAVWAR), que tiene capacidades para que interrumpa el servicio GPS a una localización geográfica que se encuentra limitad, mientras incorpora funciones de GPS a EE.UU. También ofrecen mejoras en la navegación mediante la mejora de la interoperabilidad. Tiene hasta un incremento de diez veces en la potencia de la señal civil compatible con el sistema de la Unión Europea Galileo. Es por esto, que con la ayuda de software se incrementará las aplicaciones basadas en trazo de rutas.

1.2. Descripción del Proyecto

Para la realización de este proyecto se utilizarán los datos obtenidos desde un módulo SMARTGPS que nos va ayudar a recopilar información de las distintas rutas que se realicen para su futura presentación en GOOGLE EARTH. En la figura (1-1) se presenta el módulo utilizado para la obtención de datos.

FIGURA 1-1: Descripción del proyecto

Una vez adquirido las tramas, en este caso las NMEA (NATIONAL MARINE ELECTRONICS ASSOCIATION) que es un medio a través del cual los instrumentos marítimos y receptores GPS pueden comunicarse uno con los otros, que luego serán convertidas

mediante el software a un formato entendible para GOOGLE EARTH que es .KML que es un lenguaje basado en XML para poder representar datos geográficos en tres dimensiones. Cabe mencionar que existe otro tipo de formato llamado GPX o GPS Exchange format, que también se puede usar para describir puntos, recorridos y rutas.

Para la conversión deseada se utilizó Netbeans 6.9.1. Netbeans el cual permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados *módulos.*

1.3.- Aplicaciones

La aplicación que más otorga el GPS es la determinar una posición o localización. El módulo GPS es el primer sistema que permite determinar con un error mínimo nuestra posición en cualquier parte de la tierra y bajo cualquier circunstancia. Tiene algunas aplicaciones entre las cuales, se puede usar para la parte de topografía, ya que se cuenta con la precisión del sistema, los topógrafos cuentan con esta herramienta útil para la determinación de puntos de referencia, accidentes geográficos o infraestructuras, entre otros, lo que permite disponer de información topográfica precisa, sin errores y fácilmente actualizable.

Otra aplicación sería en la parte de navegación, dado que podemos calcular posiciones en cualquier momento y de manera repetida, conocidos dos puntos se puede determinar el recorrido o, a partir de dos o más puntos conocidos, determinar la mejor ruta entre ellos.

Cabe recalcar, que con un poco de programación se podría aumentar el número de aplicaciones, dando como ejemplo que nos

sirva como una alarma, es decir en servicios de cargas, si este se saliera del límite permitido entonces generaría una señal de alerta en la cual la podrían recibir en un teléfono celular. Y así se podría aumentar las aplicaciones de este proyecto, por su puesto con la ayuda de otro sistema, en este caso algún lenguaje de programación

1.4.- Proyectos similares

1.4.1.- HSGPS

HSGPS es una gran ayuda para pilotos de cara a la navegación con Pocket Pc Windows Mobile o smartphones con pantalla táctil. Windows Mobile es un conjunto de servicios personalizados de Windows Live específicamente creado para dispositivos móviles. Es ofrecido a través de tres canales, a través de cliente (para Windows Mobile y otros dispositivos móviles compatibles, como los teléfonos Nokia), basada en Web (para GPRS móviles habilitados con navegadores web).

Con el programa tenemos todos los datos relevantes de un solo vistazo. HSGPS trabaja sin mapas digitales por lo que no requiere actualización.

Entre sus funciones principales tenemos:

- Reconocimiento automático del dispositivo GPS.
- Visualización de los aeropuertos cercanos.
- Visualización gráfica y numérica.
- Altura y ruta en sistema métrico y náutico.
- Interfaz de diferentes idiomas.
- Editor de puntos de rutas.
- Aeropuertos de Europa (ficheros KML, GOOGLE EARTH.)

En la figura (1-2) se puede observar el sistema HSGPS en funcionamiento.

FIGURA 1-2: CONVERTIDOR HSGPS

1.4.2.- CONVERTIDOR DE KML TO GPX.-

Existe otro formato de trama que es reconocido por el módulo GPS, es el .GPX. Por supuesto tiene otras características, como su encabezado pero que al final al pasarlo a la estructura .KML, se podrá graficar la ruta igualmente como si fuera una trama NMEA. Utilizado para describir puntos, trayectorias y su aplicación es de

transferir datos GPS entre aplicaciones. En la figura (1-3) se puede apreciar el prototipo de este convertidor.

FIGURA 1-3: Convertidor KML a GPX

1.4.3.- GARMIN GPS RADIO RINO

La línea GPS RADIO RINO , permite total interacción con grupos de trabajo equipo de excursión, manteniendo a todos localizados entre sí, ya que cada pantalla muestra la posición dinámica de los demás, permitiendo hacer calculos de distancia y posición entre si. Calcula posiciones de precisión, recorridos, perímetros, areas, datos de sol y luna.

En la figura (1- 4) se muestra el GPS GARMIN RADIO RINO.

FIGURA 1-4: GPS GARMIN RADIO RINO

CAPÍTULO 2

2.- FUNDAMENTO TEÓRICO

2.1.- Requerimientos para aplicación del Proyecto

Este proyecto es de software combinado con el hardware implementado por otros dos grupos de compañeros encargados de proporcionar tramas en ciertos formatos. Se pudo observar ciertas ventajas empleando Java debido a que el es un que es muy completo, es decir tiene todas las herramientas necesarias para cumplir con el objetivo del proyecto. El software de este convertidor se lo programa en NETBEANS 6.9.1 que es orientado a objetos y sirve para darle una mayor interacción con el usuario.

Para la realización de todo el software, se formó una clase de java y un jframe donde estará ubicado nuestra plataforma, es decir los respectivos botones para las distintas opciones a elegir. En la figura (2-1) se tiene los requerimientos del proyecto.

FIGURA 2-1: Requerimientos del proyecto

NETBEANS es un proyecto de código abierto de gran éxito que contiene una gran cantidad de usuarios, como se menciona es un código que puede día a día ir mejorando respecto a sus funciones. Se caracteriza por su portabilidad, ya que es compatible con la mayoría de los sistemas operativos; tales como, Windows, Mac, Linux y Solaris.

SUN MICROSYSTEMS fundó este proyecto de código abierto de netbeans, en junio del año 2000 y sigue siendo hasta la fecha su patrocinador.

Un IDE de NETBEANS es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador, y un constructor de interfaz gráfica (GUI).

2.2.- Herramientas de software

2.2.1.- NETBEANS IDE.-

Existen algunas versiones de este sencillo lenguaje de programación orientado a objetos que nos permite crear aplicaciones muy importantes, algunas complejas o sencillas a sus usuarios al momento de programar en algunas sus versiones.

El IDE NETBEANS 6.9.1 introduce JavaFX Composer, una herramienta de diseño para la creación de aplicaciones gráficas JavaFX.

Las funciones más importantes NETABEANS son: Tener soporte para crear interfaces gráficas de forma visual, crear aplicaciones para móviles, aplicaciones web.

NETBEANS IDE es el entorno usado para programar en JAVA, en la figura (2-2) se puede observar el entorno gráfico de NETBEANS 6.9.1.

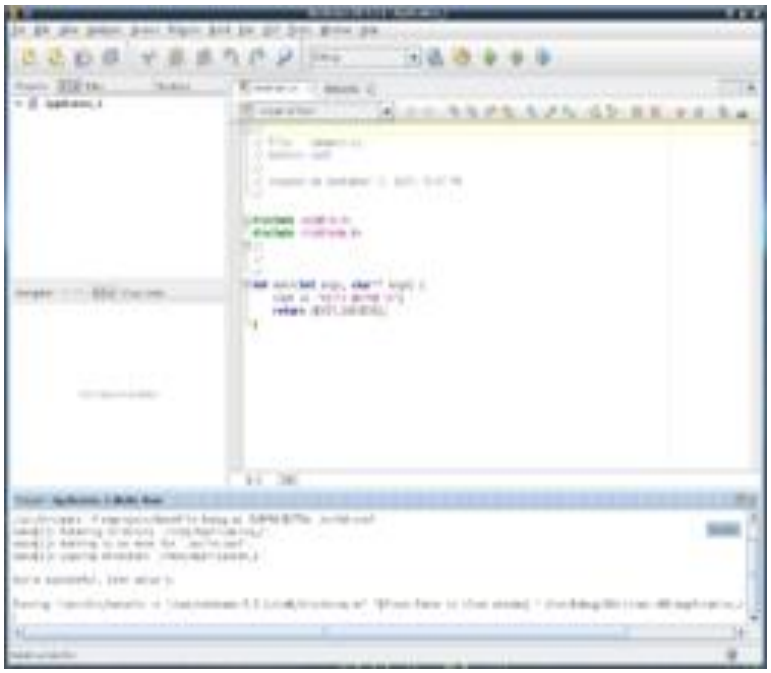


FIGURA 2-2: Entorno de NETBEANS 6.9.1

2.2.2.- TRAMAS NMEA.-

Como se puede observar en la figura (2-3), este tipo de trama que empieza con unas letras $GPGLL, son aquellas que nos envía el módulo GPS por medio del puerto USB que lo tiene integrado, hacia nuestra PC, en este caso.

```
20:23:12   $GPGLL,0210.32101,S,07953.09863,W,202312.00,A,D*6D
20:23:13   $GPGLL,0210.32015,S,07953.10065,W,202313.00,A,D*6E
```

FIGURA 2-3: TIPO DE TRAMA NMEA

El dato GPGLL nos dice la posición geográfica; estas son la latitud y longitud, el tiempo. La tabla (2-1) muestra la información que contiene cada trama de este tipo, tanto su longitud como su posición. Con esta información es con la que se ha basado para poder realizar los cálculos necesarios para su respectiva conversión hacia la trama KML.

Field	Description
$	Start of the data set
GP	Information originating from a GNSS appliance
GLL	Data set identifier
4717.115	Latitude: 47° 17.115 min
N	Northerly latitude (N=north, S= south)
00833.912	Longitude: 8° 33.912min
E	Easterly longitude (E=east, W=west)
130305.0	UTC positional time: 13h 03min 05.0sec
A	Data set quality: A means valid (V= invalid)
*	Separator for the checksum
32	Checksum for verifying the entire data set
<CR><LF>	End of the data set

TABLA 2-1: Especificación de trama NMEA

2.2.3.- TRAMAS KML.-

En la figura (2-4), se aprecia la forma de etiquetar la trama KML para su respectiva lectura en GOOGLE EARTH, también se ubican las coordenadas originadas de su conversión.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2"
 xmlns:gx="http://www.google.com/kml/ext/2.2">   <!-- required when using gx-prefixed elements -->

<Placemark>
  <name>gx:altitudeMode Example</name>
  <LookAt>
    <longitude>146.806</longitude>
    <latitude>12.219</latitude>
    <heading>-60</heading>
    <tilt>70</tilt>
    <range>6300</range>
    <gx:altitudeMode>relativeToSeaFloor</gx:altitudeMode>
  </LookAt>
  <LineString>
    <extrude>1</extrude>
    <gx:altitudeMode>relativeToSeaFloor</gx:altitudeMode>
    <coordinates>
      146.825,12.233,400
      146.820,12.222,400
      146.812,12.212,400
      146.796,12.209,400
      146.788,12.205,400
    </coordinates>
  </LineString>
</Placemark>

</kml>
```

FIGURA 2-4: Cabecera de la trama KML

En el gráfico anterior, se pudo observar como GOOGLE EARTH entiende la trama que se le envía para su posterior trazo de las rutas. De acuerdo a las coordenadas, se asigna un signo positivo o negativo. La figura (2-5) muestra la asignación del signo según su latitud y longitud.

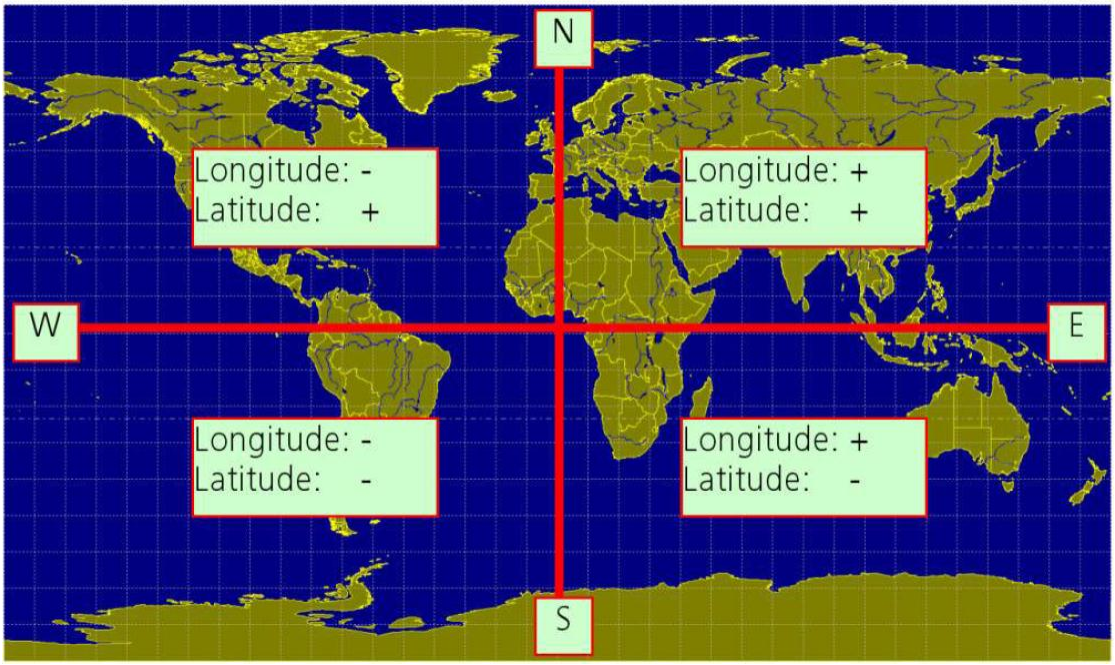


FIGURA 2-5: LONGITUD Y LATITUD

2.2.4.- Conversión de NMEA A KML.-

La figura (2-5) indica que la latitud que se encuentre dentro del rango de 0º y 180º E, y latitudes entre 0º y 90º N, van a recibir valores positivos.

En cambio las longitudes entre el rango de 0º y 180º O, y latitudes entre 0º y 90º S, van a recibir valores negativos.

Se tiene un ejemplo:

0210.32101, S, 07953.09863,O

El valor de latitud 0210.32101 S está expresada en el formato GPGLL, debería estar en el formato decimal y se describe así:

21º 0.32101’ = (21 + 0.32101/60)º negativo

El valor de longitud 79º 53’09863 O, también tiene que ser convertido a un valor decimal

79º 53’09863 = (79 + 53.09863/60) º negativo

En la figura (2-6), se puede observar la conversión de NMEA a KML. Es decir con su encabezado y las coordenadas con su respectivo signo.

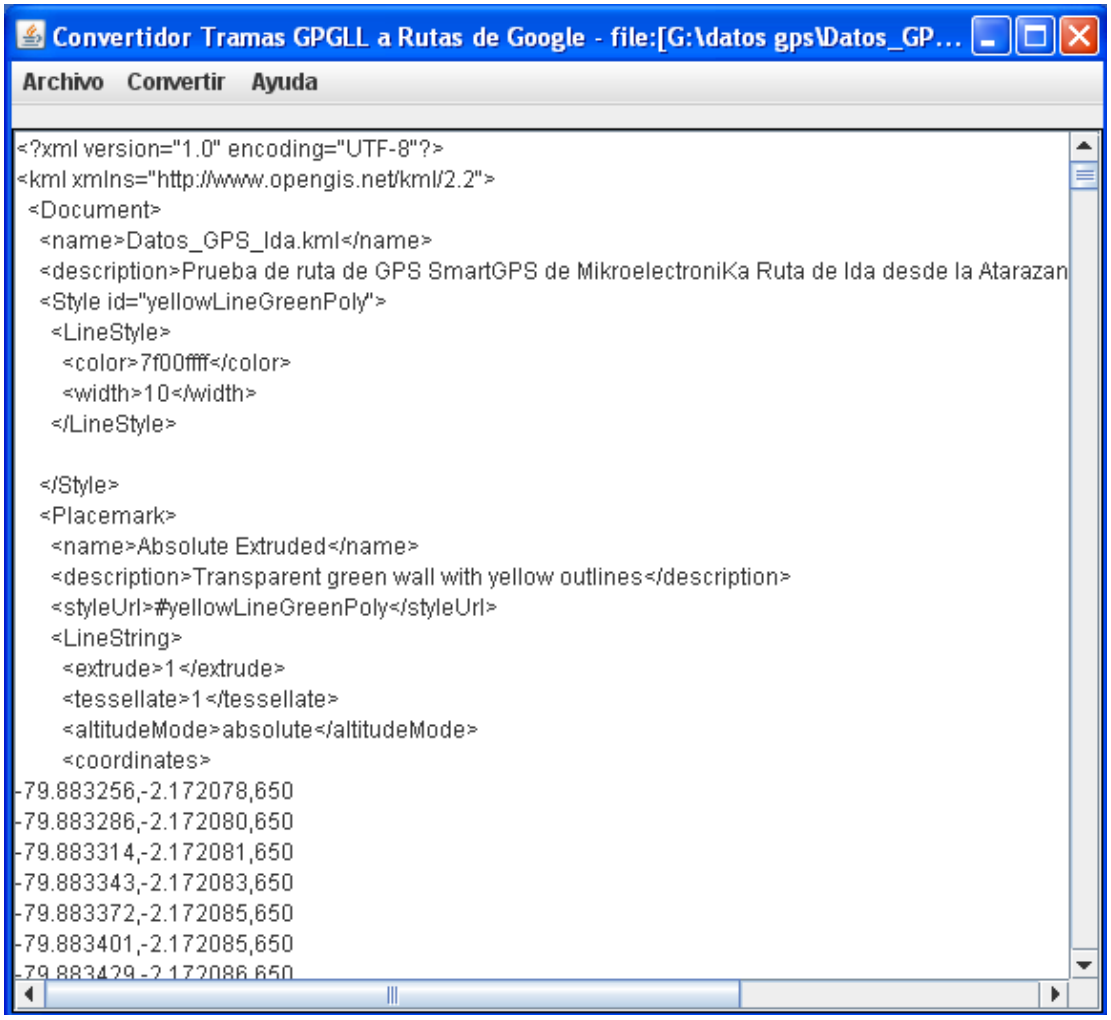


FIGURA 2-6 CONVERSION NMEA A KML

Se puede observar la figura (2-7), una ruta graficada. Para lograr esto, luego de la conversión, se arrastra el archivo con formato .KML hacia GOOGLE EARTH, y este lo grafica automáticamente.

FIGURA 2-7 RUTA GRAFICADA EN GOOGLE EARTH

CAPÍTULO 3

3.- DISEÑO E IMPLEMENTACIÓN DEL PROYECTO

En este capítulo se explica el proceso de diseño e implementación del proyecto. La primera prueba fue desarrollar una calculadora en JAVA, usando la plataforma de NETBEANS 6.9.1. Con este ejemplo, se entendió el uso de algunas herramientas fundamentales en el desarrollo del software.

Una vez entendido algunas características del software y su entorno, se empezó a desarrollar el procedimiento para lograr la conversión que se necesita para graficar las rutas obtenidas de otros compañeros. Se desarrolló un diagrama de flujo antes de desarrollar

las funciones del programa principal para ser eficaz en el funcionamiento del proyecto.

3.1.- Prueba inicial

El primer paso fue tomar datos del módulo GPS para ver como envía la trama y poder hacer el estudio de su conversión. Como se mencionó al inico del documento, se usó el módulo SMART GPS. Con estos datos a la mano, se pudo observar los campos que incluia la trama, en este caso es la NMEA. Su entendimiento era primordial para el desarrollo del proyecto.

El programa lleva un paquete por nombre TEXTO, y esta desarrollado en la plataforma de NETBEANS contiene un textoclass.java que contiene el programa principal, el enlace hacia el frame y la conversion; el textoform.java que es donde se encuentra el formulario con sus botones y opciones.

3.2.- Descripción del proyecto final

3.2.1.- Diagrama de bloques

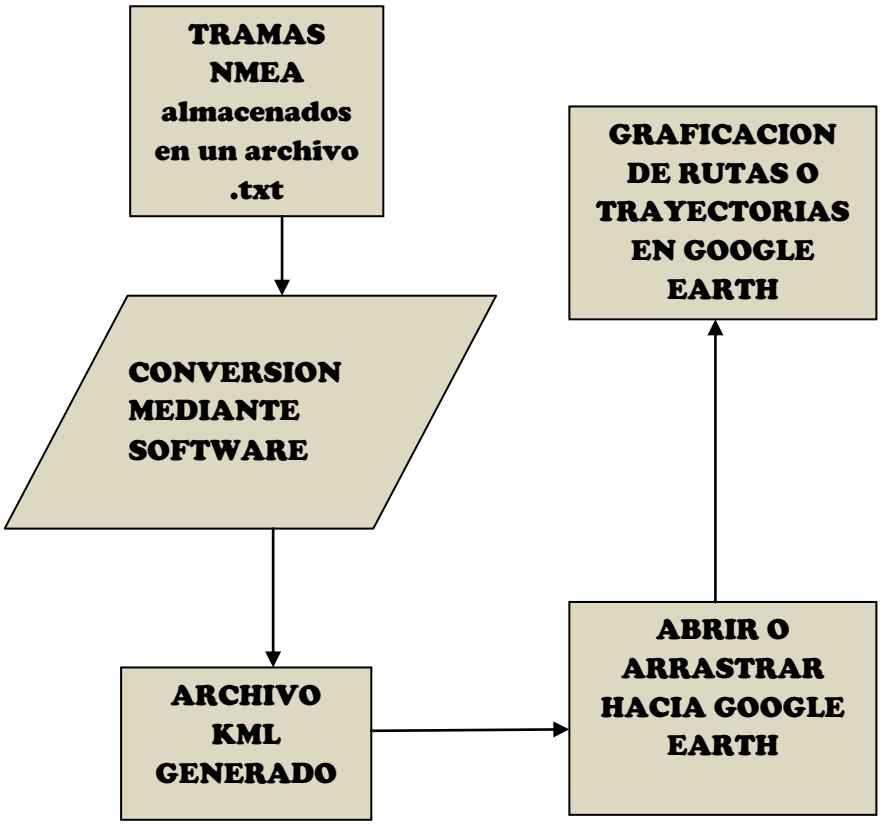


FIGURA 3-1: Diagrama de bloques del proyecto

En la figura (3-1), podemos observar que se debe obtener los datos, en este caso van hacer desde el un pendrive dado por otros compañeros. Una vez con los datos obtenidos, se sigue a la conversion que el memoria principal del proyecto, para que finalmente se obtenga la estructura KML y poder arrastrar hacia GOOGLE EARTH para su respectiva grafica de ruta o camino.

3.2.2.- Algoritmo del PROGRAMA PRINCIPAL

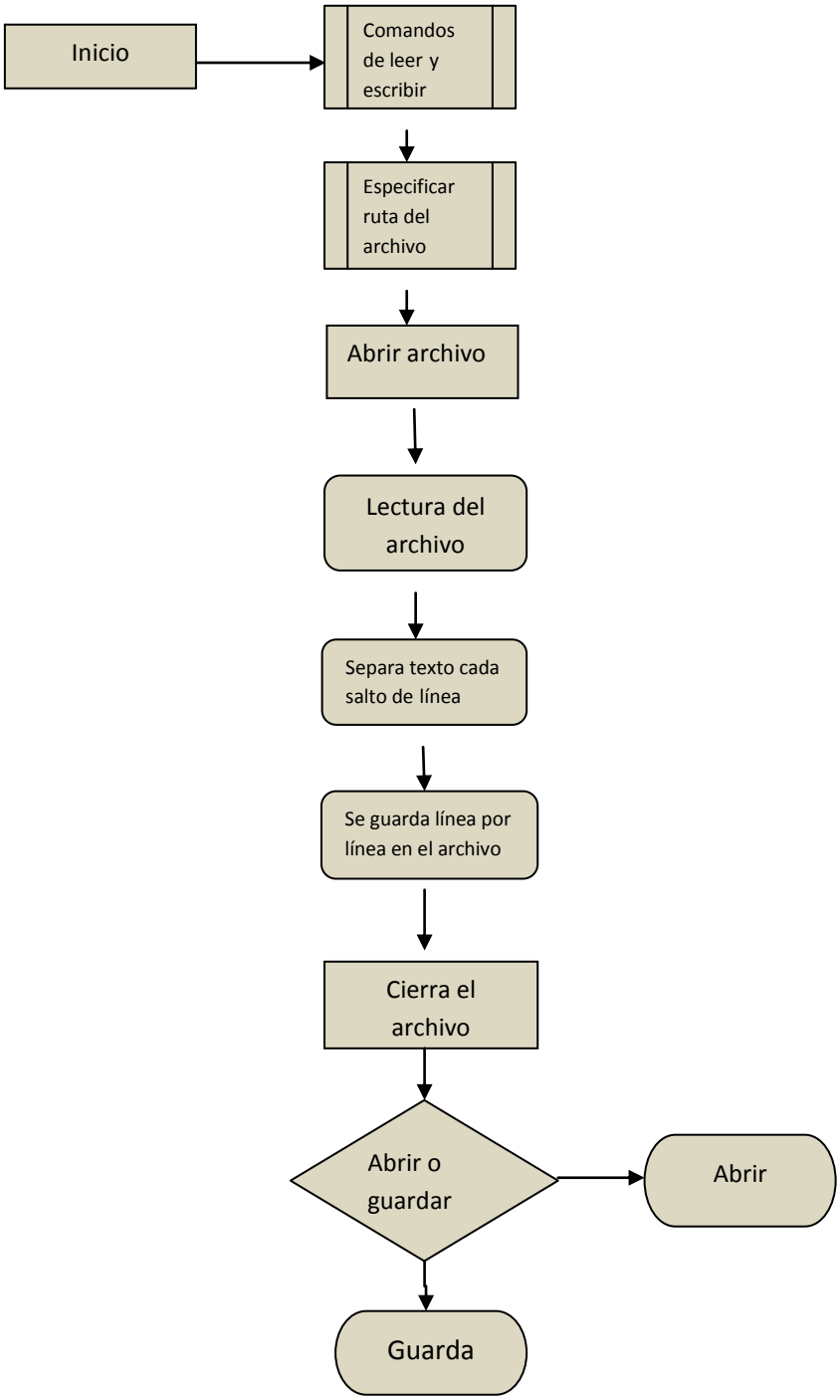


FIGURA 3-2: Algoritmo del Programa Principal

En la figura (3-2), se ve el algoritmo del programa principal se especifican los comandos para leer y escribir un archivo. Luego se da la ruta completa del archivo a manipular. Entonces se abre el archivo, se comienza a leer linea por linea.

Luego de esto, se separa el texto cada salto de linea y se guarda linea por linea en otro achivo. Despues se realiza la conversion de los datos. Se cierra el fichero y se elige la opcion que se desea, tanto abrir archivo o guardar archivo, entonces de acuerdo a lo que se elige, se guarda o se abre el fichero.

3.3.- USO DE LIBRERIAS EN JAVA

Para comenzar el programa principal es necesario mencionar a las librerias qu hemos de usar. A continuacion en la figura (3 – 3) se detallan las librerias usadas para el desarrollo de la conversion.

```
import java.io.*;                    Renombrar un fichero
import java.util.StringTokenizer;    Divide un string en substrings
import javax.swing.JFileChooser;     Abarca componentes del frame
import javax.swing.JTextArea;        Para usar un texto en el frame
```

FIGURA 3-3 LIBRERIAS USADAS EN EL PROGRAMA PRINCIPAL

Enseguida se tiene el diseño del formulario del proyecto. En esta parte se tiene la los botones que usamos, los campos tanto para abri

y guardar las tramas. En la figura (3-4) se puede apreciar el ambiente gráfico.

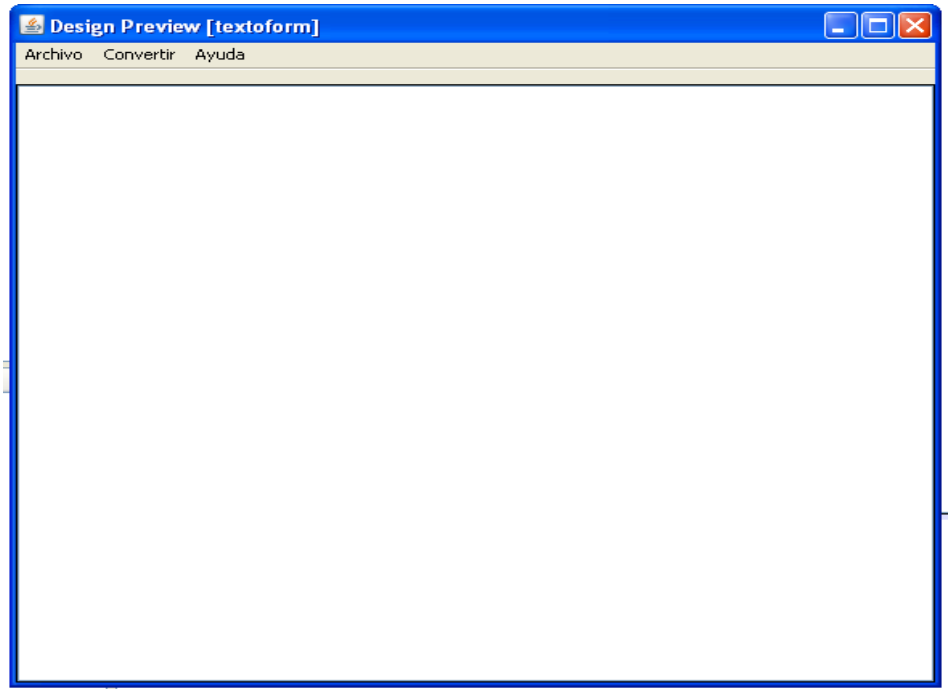


Figura 3-4: Diseño del Formulario del proyecto.-

Detallando las instrucciones, primero se parte de que se tiene un paquete llamado texto. Entonces al crear un nuevo textform, que es el área de trabajo, se escriben las instrucciones tales como public class, inicializando la variable textoform. Sirve para iniciar el formulario.

Luego se desea abrir el archivo, para esto se especifica un boton con una variable respectiva y luego se busca la dirección del fichero.

Otra opción es la de guardar el archivo, como se observa en la f igura 3-3, están las sentencias de código para implementar esta función.

Si se tiene un archivo abierto y se desea guardar también se tiene un ActionEvent, que es un evento que es como lo interpreta NETBEANS.

```
package texto;
/**
 * @web http://jc-mouse.blogspot.com/
 * @author  Mouse
 */
public class textoform extends javax.swing.JFrame {

    /** Creates new form textoform */
    public textoform() {
        initComponents();
        this.setTitle(title + "new document]");
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    Generated Code

    private void smOpenActionPerformed(java.awt.event.ActionEvent evt) {
        // Abre archivo de texto
       isnew=tc.Dialog("Open",TEXT0);
       this.setTitle(title + tc.ruta+"]");
    }

    private void smSave2ActionPerformed(java.awt.event.ActionEvent evt) {
        // Guarda en un nuevo archivo
       isnew=tc.Dialog("Save As",  TEXT0);
        this.setTitle(title + tc.ruta+"]");
}

    private void smSave1ActionPerformed(java.awt.event.ActionEvent evt) {
        // Guarda modificaciones de un archivo abierto
        if (isnew) {
           isnew = tc.Dialog("Save As", TEXT0);
           this.setTitle(title + tc.ruta + "]");
        }
        else
        {
            isnew=tc.Dialog("Save", TEXT0);
            this.setTitle(title + tc.ruta+"]");
        }
```

En la siguiente parte del código, se detalla los diferentes botones usados para las distintas opciones, tanto para convertir o la opción *acerca de* y su contenido de texto.

```
    }

    private void TEXTOKeyReleased(java.awt.event.KeyEvent evt) {
        // TODO add your handling code here:

    }

    private void smNewActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        TEXTO.setText("");
        isnew=true;
        this.setTitle(title + "new document]");
    }

    private void smAcercaActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        TEXTO.setText("CONVERTIDO TRAMAS NMEA A KML");
    }

    private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
        // Convierte archivo de texto con tramas GPGLL a .KML
       isnew=tc.Dialog("Convertir",TEXTO);
       this.setTitle(title + tc.ruta+"]");
    }

    /**
     * @param args the command line arguments
     */
    textoclass tc = new textoclass();
    Boolean isnew = true;
    String title = "Convertidor Tramas GPGLL a Rutas de Google - file:[";

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new textoform().setVisible(true);
            }
        });
    }
```

Al final de todo el procedimiento, están las declaraciones de variables que se usaron para todo tipo de objetos utilizados en el formulario, que no se tienen que modificar, puesto que generaría error.

```
    // Variables declaration - do not modify
    private javax.swing.JTextArea TEXT0;
    private javax.swing.JMenu jMenu1;
    private javax.swing.JMenu jMenu2;
    private javax.swing.JMenu jMenu3;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JMenuItem jMenuItem1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JMenuItem smAcerca;
    private javax.swing.JMenuItem smNew;
    private javax.swing.JMenuItem smOpen;
    private javax.swing.JMenuItem smSave1;
    private javax.swing.JMenuItem smSave2;
    // End of variables declaration

}
```

3.4.- Funciones implementadas en el Programa Principal

3.4.1.- Inicialización

Primero se obtiene la ruta del archivo que se va a convertir. Para esto, con la ayuda de una variable buffer, le asignamos el archivo a ese buffer. Para después empezar a leer el archivo línea por línea.

```
public textoclass(){
}

private String OpenFile(String ruta){
    String t="";
    try {
        archivo = new File (ruta);
        fr = new FileReader (archivo);
        br = new BufferedReader(fr);
        // Lectura del fichero linea por linea
        String linea;
        while((linea=br.readLine())!=null)
            t = t + linea + "\n";
    }
```

3.4.2.- Comprobación el archivo.-

Esta función si el archivo que se tiene se ha cerrado correctamente.

Esta instrucción finally nos sirve para verificar esa tarea.

```
   catch(Exception e){
      e.printStackTrace();
   }finally{
      try{
         if( null != fr ){
            fr.close();
         }
      }catch (Exception e2){
         e2.printStackTrace();
      }
   }
     return t;
}
```

3.4.3.- Uso de bandera de retorno.-

En las siguientes instrucciones se tiene una bandera de retorno que sirve para controlar las opciones que se elijan. También se tiene la

ventana para abrir el archivo, el cual la bandera nos va a regresar un FALSE si se abre.

```
public boolean  Dialog(String Opcion, JTextArea tArea){
    //esta variable se utiliza como una bandera de retorno
    //no es necesaria, solo sirve para tener un control
    boolean b=true;
    //ventana para "abrir un archivo", retorna FALSE si se abre
    if (Opcion.equals("Open")){
            JFileChooser fileChooser = new JFileChooser();
            int result = fileChooser.showOpenDialog(null);
            if ( result == JFileChooser.APPROVE_OPTION ){
                ruta = fileChooser.getSelectedFile().getAbsolutePath();
                tArea.setText(OpenFile(ruta));
                b=false;
            }
    }
```

3.4.4.- Función para abrir archivo.-

En las siguientes instrucciones se tiene una bandera de retorno que sirve para controlar las opciones que se elijan. También se tiene la ventana para abrir el archivo, el cual la bandera nos va a regresar un FALSE si se abre.

```
public boolean  Dialog(String Opcion, JTextArea tArea){
    //esta variable se utiliza como una bandera de retorno
    //no es necesaria, solo sirve para tener un control
    boolean b=true;
    //ventana para "abrir un archivo", retorna FALSE si se abre
    if (Opcion.equals("Open")){
            JFileChooser fileChooser = new JFileChooser();
            int result = fileChooser.showOpenDialog(null);
            if ( result == JFileChooser.APPROVE_OPTION ){
                ruta = fileChooser.getSelectedFile().getAbsolutePath();
                tArea.setText(OpenFile(ruta));
                b=false;
            }
    }
```

3.4.5.- Función para guardar archivo.-

Las siguientes instrucciones son necesarias para guardar el archivo que se desea. Se selecciona la ruta hacia donde vamos a grabar nuestro programa. Con estas instrucciones se crea un bot dentro del textform que es donde se tiene el formulario.

```
else if (Opcion.equals("Save As")){
        JFileChooser fileChooser = new JFileChooser();
        int result = fileChooser.showSaveDialog(null);
        if ( result == JFileChooser.APPROVE_OPTION ){
            ruta = fileChooser.getSelectedFile().getAbsolutePath();
            ruta = ruta + ".txt";
            SaveFile(tArea.getText() ,ruta);
            b=false;
        }
}
```

3.*4.6.*- Guarda automáticamente un archivo.-

En este codigo se enfatiza que se puede guardar automaticamente un archivo que esta previamente abierto, es decir no se necesita crear otra ventana dentro del formulario para esta opcion.

```
        else if(Opcion.equals("Save")){
            SaveFile(tArea.getText() ,ruta);
            b=false;
          }
         else if(Opcion.equals("Convertir")){
                  JFileChooser fileChooser = new JFileChooser();
                  int result = fileChooser.showOpenDialog(null);
                  if ( result == JFileChooser.APPROVE_OPTION ){
                      ruta = fileChooser.getSelectedFile().getAbsolutePath();
                      Convertir(ruta,tArea);
                      b=false;

          }}
          return b;
    }
}
```

3.4.7.- Función de convertir.-

Se convierte las tramas NMEA a KML por medio de las siguientes instrucciones que se muestran acontinuacion. Se utiliza las variables de longitud y latitud como tipo double. Se lee el archivo y luego se guarda en un buffer como respaldo. Luego se ve la ruta, y se escribe

en un fichero nuevo, adjuntando el formato de conversion KML, es decir su encabezado.

```
private void Convertir(String ruta, JTextArea tArea){
      String renglon,s;
      double latitud,longitud,temp;
    try
    {
        archivo = new File (ruta);
        fr = new FileReader (archivo);
        br = new BufferedReader(fr);

                        JFileChooser fileChooser = new JFileChooser();
            int result = fileChooser.showSaveDialog(null);
            if ( result == JFileChooser.APPROVE_OPTION ){
                rutag = fileChooser.getSelectedFile().getAbsolutePath();
                rutag = rutag + ".kml";}
                //SaveFile(tArea.getText() ,rutag);}

        fichero = new FileWriter(rutag);
        pw = new PrintWriter(fichero);
```

3.5.- Programa principal del proyecto

package texto;

import java.io.*;

import java.util.StringTokenizer;

import javax.swing.JFileChooser;

import javax.swing.JTextArea;

public class textoclass {

//para leer

```
File archivo = null;
FileReader fr = null;
BufferedReader br = null;
//para escribir
FileWriter fichero = null;
PrintWriter pw = null;
//ruta absoluta del archivo a manipular
String ruta = "",rutag="";

   public textoclass(){
   }
      private String OpenFile(String ruta){
      String t="";
      try {
         archivo = new File (ruta);
         fr = new FileReader (archivo);
         br = new BufferedReader(fr);
         // Lectura del fichero línea por línea
         String linea;
         while((linea=br.readLine())!=null)
          t = t + linea + "\n";
```

```
    }
  catch(Exception e){
     e.printStackTrace();
  }finally{
     try{
        if( null != fr ){
           fr.close();
        }
     }catch (Exception e2){
        e2.printStackTrace();
     }
  }
    return t;
}

private void SaveFile(String t, String ruta){
//se separa el texto cada salto de linea
StringTokenizer st = new StringTokenizer(t,"\n");
try
    {
        fichero = new FileWriter(ruta);
```

```
        pw = new PrintWriter(fichero);
        //se guarda linea por linea en el archivo
        while(st.hasMoreTokens()){
           String line = st.nextToken();
           pw.println(line);
        }
     } catch (Exception e) {
        e.printStackTrace();
     } finally {
       try {
       if (null != fichero)
         fichero.close();
       } catch (Exception e2) {
         e2.printStackTrace();
       }
     }
}


private void Convertir(String ruta, JTextArea tArea){
     String renglon,s;
```

```
  double latitud,longitud,temp;
 try
 {
    archivo = new File (ruta);
    fr = new FileReader (archivo);
    br = new BufferedReader(fr);

                JFileChooser fileChooser = new JFileChooser();
       int result = fileChooser.showSaveDialog(null);
       if ( result == JFileChooser.APPROVE_OPTION ){
          rutag = fileChooser.getSelectedFile().getAbsolutePath();
          rutag = rutag + ".kml";}

    fichero = new FileWriter(rutag);
    pw = new PrintWriter(fichero);

    pw.println("<?xml version=\"1.0\" encoding=\"UTF-
8\"?>\r\n<kml xmlns=\"http://www.opengis.net/kml/2.2\">\r\n
<Document>\r\n    <name>Datos_GPS_Ida.kml</name>\r\n
<description>Prueba de ruta de GPS SmartGPS de MikroelectroniKa Ruta
de Ida desde la Atarazana a Samanes hecho en KML.</description>\r\n
<Style id=\"yellowLineGreenPoly\">\r\n      <LineStyle>\r\n
<color>7f00ffff</color>\r\n       <width>10</width>\r\n
```

```
</LineStyle>\r\n\r\n    </Style>\r\n    <Placemark>\r\n
<name>Absolute Extruded</name>\r\n      <description>Transparent
green wall with yellow outlines</description>\r\n
<styleUrl>#yellowLineGreenPoly</styleUrl>\r\n      <LineString>\r\n
<extrude>1</extrude>\r\n        <tessellate>1</tessellate>\r\n
<altitudeMode>absolute</altitudeMode>\r\n        <coordinates>\r\n");

while ((renglon = br.readLine()) != null) {

   latitud=longitud=temp=0;

   s="";

   for (int i = 0; i < renglon.length(); i++){


      if (renglon.charAt(i) == '$'){

         do i++;

            while(renglon.charAt(i) != ',');


         }else if (renglon.charAt(i) == 'S'|| renglon.charAt(i) == 'W' ||
renglon.charAt(i) == 'N'|| renglon.charAt(i) == 'O'){


                  switch(renglon.charAt(i)){

                    case 'S': latitud=temp; latitud = 0 - latitud; break;

                    case 'W': longitud=temp;longitud= 0 - longitud; break;

                    case 'N': latitud=temp; break;

                    case 'O': longitud=temp; break;
```

```
                }
                s=",";
                //System.out.println(latitud);
                //System.out.println(longitud);
              }else if(renglon.charAt(i) != ',' &&(latitud == 0 || longitud ==
0)){
                    s +=String.valueOf(renglon.charAt(i));
                    }else if(s == ","){
                        s="";
                      }else if(latitud == 0 || longitud == 0){
                    temp = Double.parseDouble(s);
                    s="";
                   }else{}
      }
      latitud=(int)(latitud/100)+((latitud%100)/60);
      longitud=(int)(longitud/100)+((longitud%100)/60);
      pw.println(longitud+","+latitud+",650\r\n");
   }
      pw.println("\r\n      </coordinates>\r\n     </LineString>\r\n
</Placemark>\r\n  </Document>\r\n</kml>\r\n");
      }
```

```
            catch (Exception e) {
        e.printStackTrace();
        } finally {
          try {
          // Nuevamente aprovechamos el finally para
          // Asegura que se cierra el fichero.
          if (null != fichero)
            fichero.close();
          } catch (Exception e2) {
            e2.printStackTrace();
          }
        }

    }
//según la opción muestra en pantalla una ventana de dialogo
 //para "abrir" o "guardar" un archivo
 public boolean  Dialog(String Opcion, JTextArea tArea){
    //esta variable se utiliza como una bandera de retorno
    //no es necesaria, solo sirve para tener un control
    boolean b=true;
    //ventana para "abrir un archivo", retorna FALSE si se abre
```

```
if (Opcion.equals("Open")){
      JFileChooser fileChooser = new JFileChooser();
      int result = fileChooser.showOpenDialog(null);
      if ( result == JFileChooser.APPROVE_OPTION ){
         ruta = fileChooser.getSelectedFile().getAbsolutePath();
         tArea.setText(OpenFile(ruta));
         b=false;
      }
}
```

//ventana para guardar un archivo

```
else if (Opcion.equals("Save As")){
      JFileChooser fileChooser = new JFileChooser();
    int result = fileChooser.showSaveDialog(null);
      if ( result == JFileChooser.APPROVE_OPTION ){
         ruta = fileChooser.getSelectedFile().getAbsolutePath();
         ruta = ruta + ".txt";
         SaveFile(tArea.getText() ,ruta);
         b=false;
      }
}
```

```
      //No  muestra ninguna ventana, guarda automáticamente el
archivo
      //respecto a un archivo previamente abierto
       else if(Opcion.equals("Save")){
          SaveFile(tArea.getText() ,ruta);
          b=false;
       }
      else if(Opcion.equals("Convertir")){
            JFileChooser fileChooser = new JFileChooser();
            int result = fileChooser.showOpenDialog(null);
            if ( result == JFileChooser.APPROVE_OPTION ){
               ruta = fileChooser.getSelectedFile().getAbsolutePath();
               Convertir(ruta,tArea);
               b=false;
       }}
       return b;
   }
}
```

CAPÍTULO 4

4.- SIMULACIÓN Y PRUEBAS

En este capítulo se describe las simulaciones y pruebas ejecutadas en el proyecto, como se explicó anteriormente, el código de todo el programa está hecho en NETBEANS 6.9.1, por lo que su simulación está hecha en esa plataforma. Se muestran las ventanas usadas en el desarrollo de la aplicación.

Se muestra en la primera parte las pruebas que realizó para crear el formulario con sus respectivas opciones, como esta compuesto el formulario para su ejecución. Se desea realizar una aplicación altamente fácil para el manejo de sus funciones.

Finalmente se muestra, el diseño compilado sin errores para empezar el uso del mismo, para obtener tramas KML, que permita su fácil interacción con GOOGLE EARTH.

4.1.- Simulación en NETBEANS 6.9.1.-

En la figura (4-1),a través de la pestana DESIGN de NETBEANS se puede observar cómo se obtendrá el diseño cuando se ejecute el programa con la opción RUN FILE. Es como un borrador en donde se puede cambiar su diseño a convenir, por supuesto dándo una mejor estética y presentación, para que sea de fácil uso.

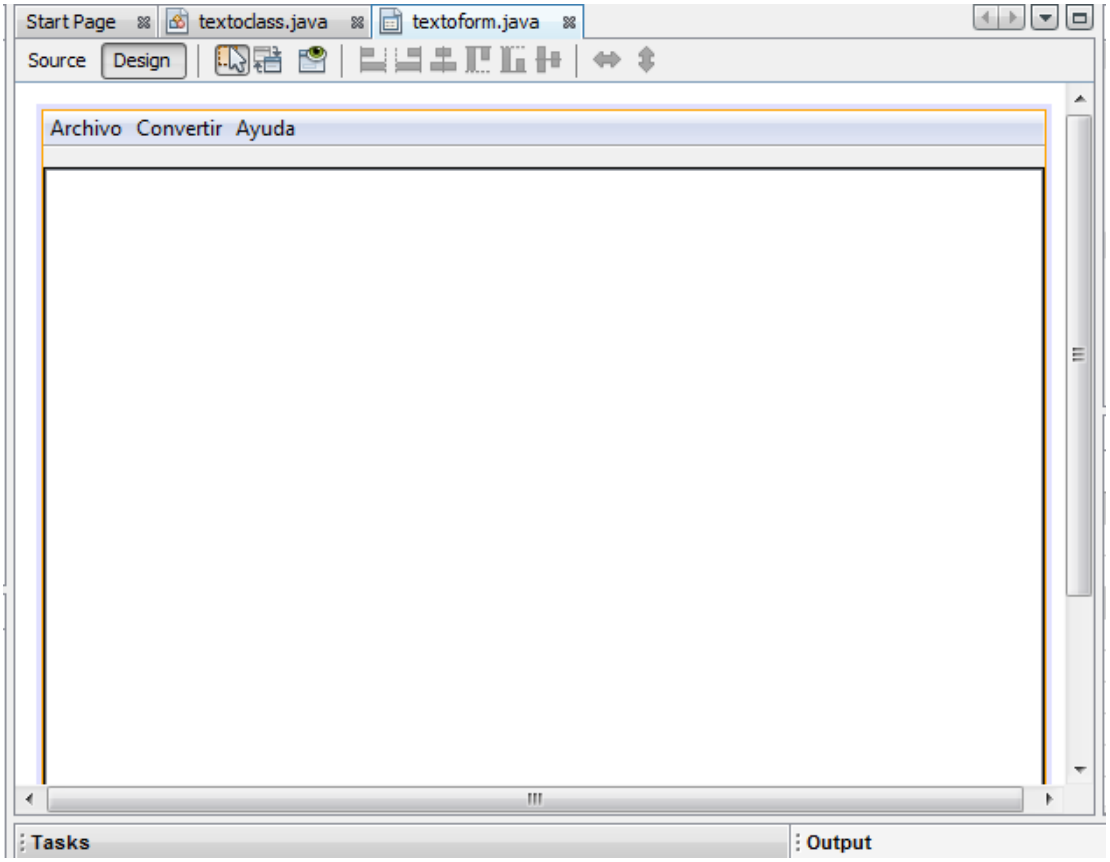


FIGURA 4-1: Simulación en NETBEANS

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES:

1. Se logró desarrollar un software que permite la fácil y dinámica interacción con el programa de disponibilidad gratuita GOOGLE EARTH, obteniendo de este modo gráficas de rutas y trayectorias realizadas.
2. Se aprendió el funcionamiento de la plataforma de NETBEANS 6.9.1, que es un lenguaje de programación orientado a objetos, la cual consta de un constructor de interfaz gráfica con lo cual se puede diseñar gráfico complejos.
3. Se pudo obtener satisfactoriamente los datos otorgados por dos compañeros, que fueron de un pen drive y una tarjeta de memoria SD. Estos datos contienen tramas con posición geográfica enviadas por el módulo GPS.

4. Para obtener los datos, fue necesario conocer el módulo GPS, ahí se observó su funcionamiento y sus principales aplicaciones; la cual nos dio el posicionamiento de los puntos necesarios para la representación en GOOGLE EARTH.
5. Para la conversión, los datos obtenidos inician con la el campo $, que es el identificador de la trama NMEA, los diferentes tipos de tramas la utilizan, el programa descarta cualquier otra letra o símbolo como inicio de la trama.

RECOMENDACIONES:

1. Para comenzar se debería obtener el instalador de NETBEANS 6.9.1, que es un código abierto y se puede descargar de su página principal. Cabe mencionar que para instalarlo es necesario instalar primero los drivers.
2. Para iniciarse en la programación orientada a objetos, es necesario entender el funcionamiento de la plataforma que se usa, para esto se debe realizar varios ejercicios antes y leer el tutorial.

3. Luego de haber realizado los ejemplos y haber investigado las principales funciones que tiene NETBEANS, se debe realizar un diagrama de bloques del programa principal, para estimar las variables y funciones a utilizar.
4. Es necesario investigar sobre el módulo GPS, para saber la forma en que los datos son enviados y que es lo que se va a receptar. Esto incluye todas sus aplicaciones y sus características
5. Se debe tener en cuenta que para la conversión de la trama NMEA a la estructura .KML, que los datos guardados en un archivo .txt, tengan el inicio de la trama su identificador $.
6. Se recomienda el uso de un tutorial de java, el cual está en la parte de anexos, con esto se podrá entender su plataforma.

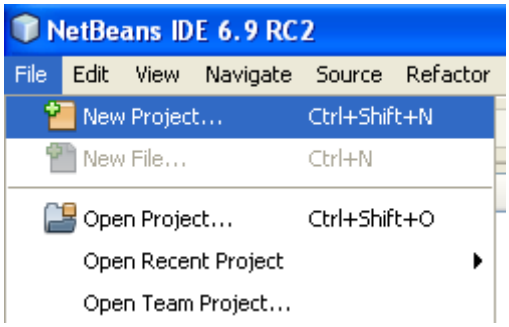
ANEXOS

ANEXO A.- Configuración del proyecto

Para crear un proyecto de IDE:

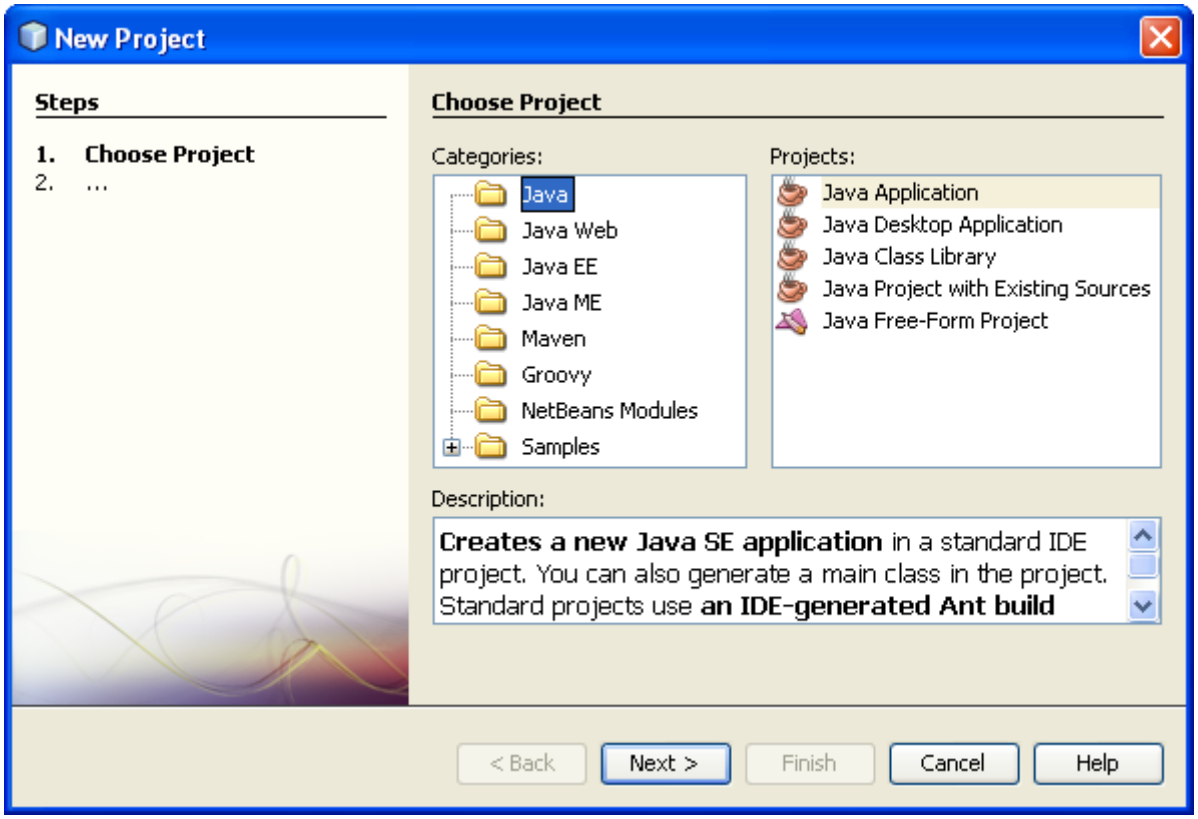
NetBeans IDE inicio.

En el IDE, haga clic en Archivo> Nuevo proyecto (Ctrl + Mayúsculas + N), como se muestra en la figura siguiente.



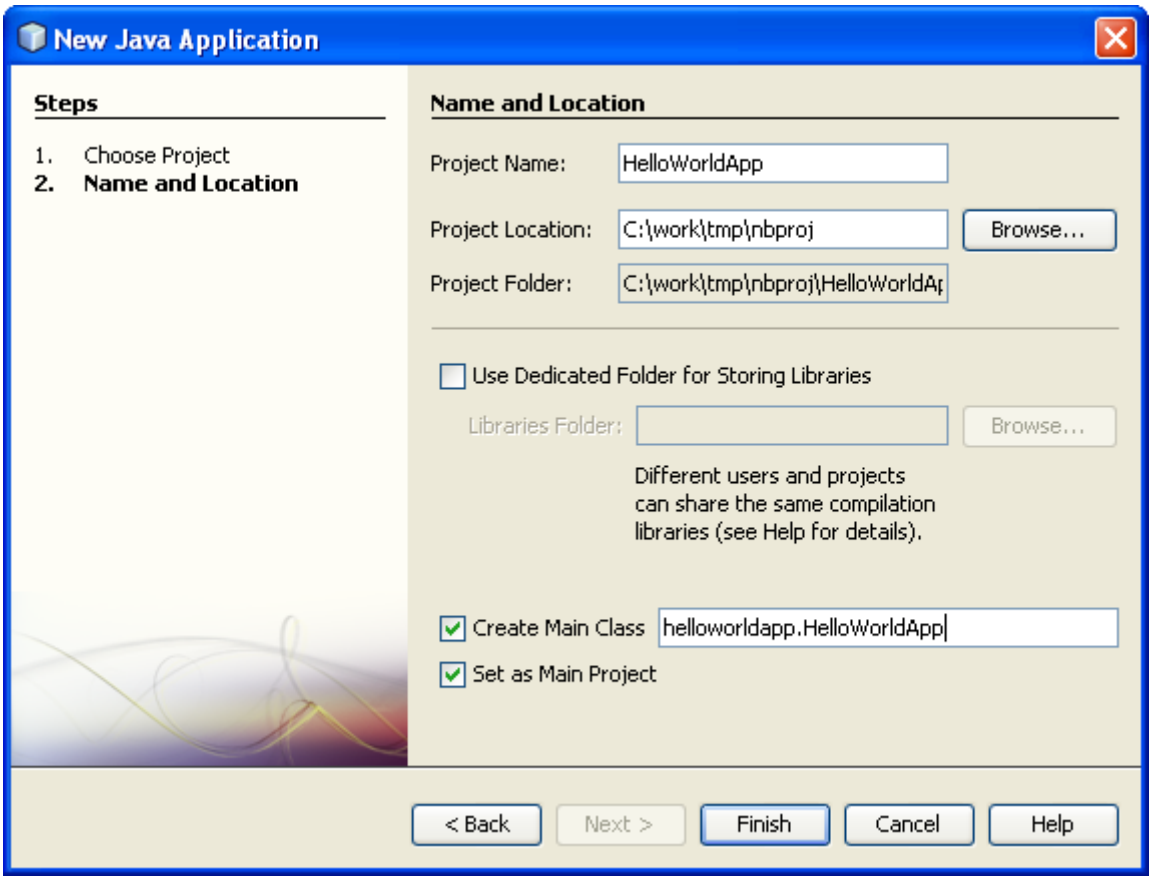
En el asistente Nuevo proyecto, expanda la categoría de Java y seleccione Java Application como se muestra en la figura siguiente. A continuación,

haga clic en Siguiente.



1. En la página Nombre y Ubicación del asistente, haga lo siguiente (como se muestra en la figura siguiente):

- En el campo Nombre del proyecto, el tipo HelloWorldApp .
- Deja el uso dedicado de carpetas para almacenar bibliotecas casilla de verificación sin seleccionar.
- En la clase principal campo Crear, escriba helloworldapp.HelloWorldApp .
- Deje la casilla de verificación Establecer como principal del proyecto seleccionado.

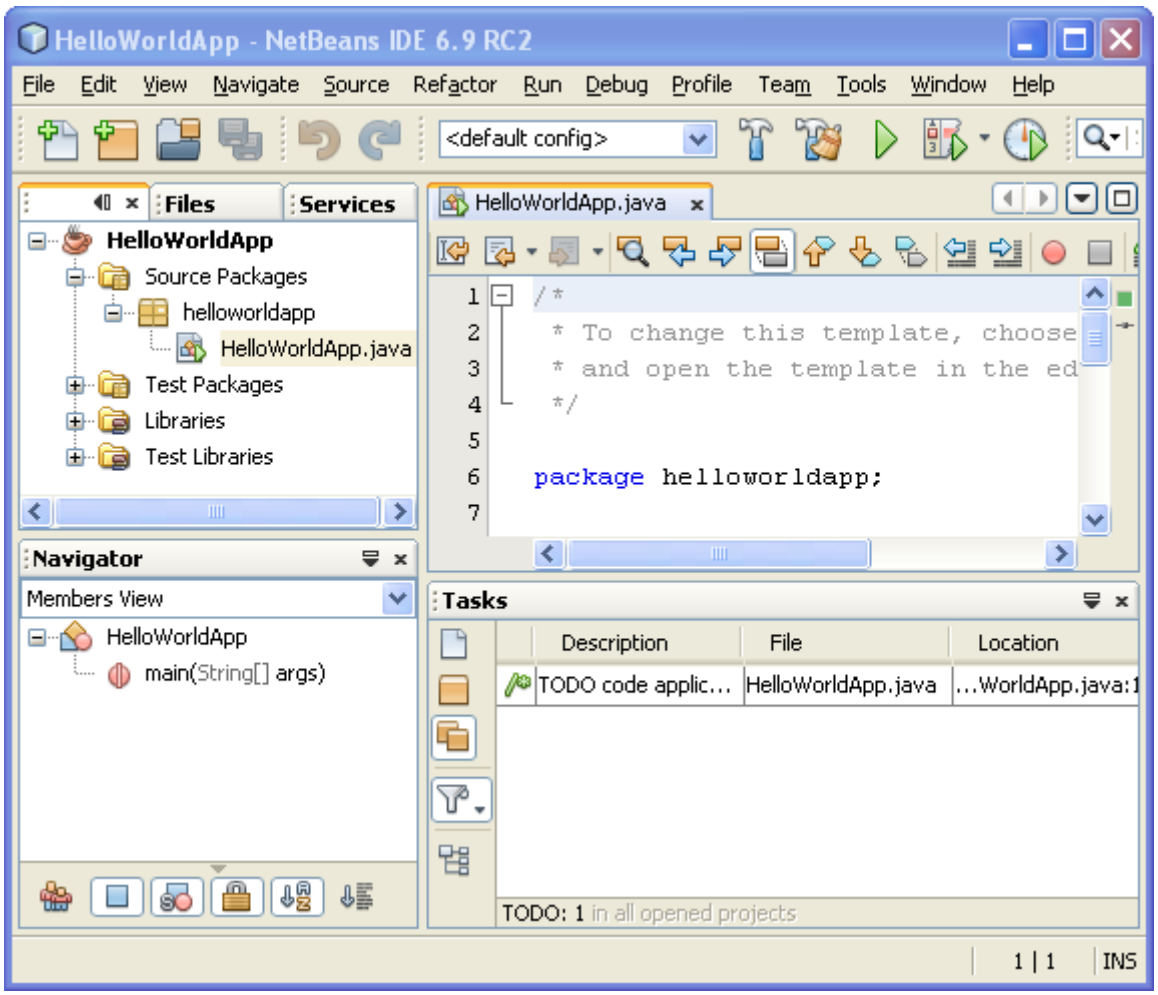


2. Haga clic en Finalizar.

El proyecto se crea y se abre en el IDE. Usted debe ver los siguientes componentes:

- La ventana de proyectos, que contiene una vista de árbol de los componentes del proyecto, incluyendo los archivos fuente, las librerías que el código depende, y así sucesivamente.
- La ventana del editor de origen con un archivo llamado HelloWorldApp abierto.
- La ventana del navegador, que puede utilizar para navegar rápidamente entre los elementos dentro de la clase seleccionada.

- La ventana de tareas, que enumera los errores de compilación, así como otras tareas que están marcadas con palabras clave como XXX y TODO.



Agregar código para el archivo de origen de creación

Debido a que han salido de la casilla Create Main Class seleccionado en el Asistente para nuevo proyecto, el IDE ha creado una clase principal esqueleto para usted. Usted puede agregar el "Hola Mundo!" mensaje con el código esqueleto mediante la sustitución de la línea:

```
/ / TODO del código de aplicación lógica aquí
```

con la línea:

```
        System.out.println ("Hola Mundo!");
```

Guardar los cambios, seleccione Archivo> Guardar.

El archivo debe ser similar al siguiente ejemplo de código.

```
/ *
 * Para cambiar esta plantilla, seleccione Herramientas | Plantillas
 * Y abra la plantilla en el editor.
 * /

paquete HelloWorldApp;

/ **
 *
 * @ <Nombre de autor <su
 * /
public class HelloWorldApp {

   / **
    * @ Param args argumentos de la línea de comandos
    * /
   public static void main (String [] args) {
        System.out.println ("Hola Mundo!");
   }
```

```
}
```

Compilar y ejecutar el Programa de

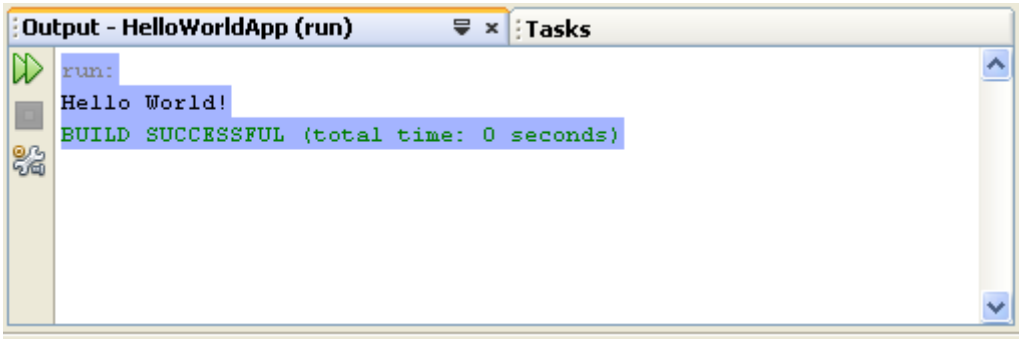
Debido a compilar el IDE en función de Ahorro, no tiene que compilar manualmente el proyecto con el fin de que se ejecute en el IDE. Cuando se guarda un archivo de código fuente de Java, el IDE automáticamente compila.

La característica de compilación en Guardar se puede desactivar en la ventana Propiedades del proyecto. Haga clic en el proyecto, seleccione Propiedades. En la ventana Propiedades, seleccione la ficha Compilar. La compilación de casilla de verificación Guardar se encuentra justo en la parte superior. Tenga en cuenta que en la ventana Propiedades del proyecto puede configurar las opciones numerosas para su proyecto: Proyecto de Bibliotecas, envases, construcción, funcionamiento, etc

Para ejecutar el programa:

- Elija ejecutar el proyecto> Run Main (F6).

La siguiente figura muestra lo que ahora debería ver.



¡Enhorabuena! Su programa funciona!

Si hay errores de compilación, que están marcados con glifos rojo en el margen izquierdo y derecho del Editor de código fuente. Los glifos en el margen izquierdo indican los errores de las líneas correspondientes. Los glifos en el margen derecho mostrar todas las áreas del archivo que tienen errores, como errores en las líneas que no son visibles. Puede ratón sobre una marca de error para obtener una descripción del error. Puede hacer clic en un glifo en la margen derecha para saltar a la línea con el error.

Generar e implementar la aplicación

Que ha escrito y la prueba de ejecutar la aplicación, puede utilizar la limpieza y construcción de mandar construir su aplicación para su implementación. Una vez, cuando se utiliza la limpieza y orden de construcción, el IDE ejecuta un script que realiza las tareas siguientes:

- Elimina los archivos compilados previamente y otras acumulaciones de productos.
- Vuelve a compilar la aplicación y crea un archivo JAR que contiene los archivos compilados.

Para generar la aplicación:

- Elija Ejecutar> Limpieza y Construcción del Proyecto Principal (Mayúsculas + F11)

Usted puede ver la construcción de salidas al abrir la ventana de archivos y expandiendo el nodo HelloWorldApp. El código de bytes del archivo compilado HelloWorldApp.class está dentro de la build/classes/helloworldapp subnodo. Un archivo JAR que contiene el despliegue HelloWorldApp.class está dentro de la dist nodo

BIBLIOGRAFÍA

1. A Simon & Schuster Company, Deitel y Deitel, Cómo programar en JAVA, Primera Edición, 1998

2. GPS , GPS Data logger with SD card storage and GOOGLE EARTH map interface, http://www.mikroe.com/eng/products/view/372/data-logger-with-sd-card-storage-article/ , 23/08/2010

3. GPS ,Essentials of Satellite Navigation Compendium, www.u-blox.com , 23/08/2010

4. GPS System, http://www.mikroe.com/eng/downloads/get/640/en_article_c_avr_04_09.pdf , 23/08/2010

5. NETBEANS, PLATAFORMA NETEBEANS 6.9.1, http://netbeans.org/ , 5/09/2010

6. Tutorial de ejemplo en Netbeans, http://www.youtube.com/watch?v=3E_Ua69vHCg , 5/09/2010

7. Protocolo NMEA, http://es.wikipedia.org/wiki/NMEA ,5/09/2010