DOCUMENTACIÓN DEL PROTOCOLO NMEA0183 PARA APLICACIÓN EN UN ROBO

INGRID ALEXANDRA RAMÍREZ PEDRAZA

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE CIENCIAS FISICO-MECÁNICAS
ESPECIALIZACIÓN EN INGENIERÍA MECATRÓNICA
ESCUELA DE INGENIERIA MECÁNICA
BUCARAMANGA
2008

DOCUMENTACIÓN DEL PROTOCOLO NMEA0183 PARA APLICACIÓN EN UN ROBOT MÓVIL

INGRID ALEXANDRA RAMÍREZ PEDRAZA

Trabajo de monografía para optar al título de Especialista en Ingeniería Mecatrónica

Director
JORGE ENRIQUE MENESES FLORES
Ingeniero Mecánico

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE CIENCIAS FISICO-MECÁNICAS
ESPECIALIZACIÓN EN INGENIERÍA MECATRÓNICA
ESCUELA DE INGENIERIA MECÁNICA
BUCARAMANGA
2008

DEDICATORIA

A Dios por estar siempre conmigo,

a mi familia por su apoyo incondicional**.**

Ingrid.

AGRADECIMIENTOS

La autora expresa sus agradecimientos a:

JORGE ENRIQUE MENESES FLOREZ, Ingeniero Mecánico, profesor de la escuela de Ingeniería Mecánica de la UIS y director de este trabajo de monografía, por su colaboración durante el desarrollo del mismo.

ALFREDO DIAZ CLAROS, Administrador de Empresas, especialista en Telecomunicaciones de la Universidad Autónoma de Bucaramanga UNAB, por su asesoría y colaboración durante el desarrollo de este proyecto.

HARLINSON TRUJILLO, Ingeniero de Sistemas, estudiante de maestría en Ingeniería, por su acertada asesoría y colaboración durante el desarrollo de este proyecto.

CONTENIDO

LISTA DE CUADROS

LISTA DE FIGURAS

LISTA DE ANEXOS

RESUMEN

TÍTULO:
DOCUMENTACION DEL PROTOCOLO NMEA0183 PARA LA APLICACIÓN EN UN ROBOT MÓVIL

AUTORES:
RAMIREZ PEDRAZA, Ingrid Alexandra *

PALABRAS CLAVES:
GPS, NMEA0183, robot móvil.

DESCRIPCIÓN:

El Grupo de Investigación en Mecatrónica (GIMKT) han puesto su atención sobre el campo de los robots móviles. Estos dispositivos usan para realizar sus desplazamientos, sistemas de navegación satelital para conocer la posición en un determinado momento, consta de un grupo de satélites orbitando la tierra y receptores que reciben datos de posición, velocidad, etc.

El estudio del protocolo de transmisión de datos NMEA0183 de los sistemas de navegación global es el primer paso para establecer las bases necesarias para realizar futuras aplicaciones. Los datos que maneja la norma NMEA 0183 se encuentran organizados por medio de sentencias que tienen una función específica existiendo 3 clases básicas que son: del emisor, propietarias, de consulta. Las primeras indican la identidad de dispositivo emisor, las segundas fueron creadas por diferentes manufactureros de GPS, por último, las de consultas emitidas por un receptor para solicitar una información en particular.

Se buscó un software freeware "Open Source"; la aplicación NMEA PARSER DEMO de Visualgps (VGPS) fue la apropiada. Esta es una aplicación Windows desarollada en Microsoft Visual C/C++ versión 6.0, donde se muestra datos de sentencias básicas suministrando información sobre latitud, longitud, altitud, calidad de la señal y datos de los satélites en contacto con el GPS, número de sentencias recibidas, mediante una comunicación serial entre GPS y PC. Basado en el formato de las sentencias de la norma, la aplicación fue estructurada para analizar una sentencia por medio de una máquina de estado en donde se busca el comienzo y final de la sentencia, el campo de dirección, los datos, el cálculo del checksum, la información es almacenada, se llama la función apropiada para extractar los datos y mostrarlos en la ventana principal

---

* Trabajo de Grado

** Facultad de Ingenierías Físico-Mecánicas, Escuela de Ingeniería Mecánica. Ing. Jorge Enrique Meneses Flores

ABSTRACT

TITLE:
DOCUMENTACION OF THE NMEA0183 PROTOCOL FOR THE APPLICATION IN A MOBILE ROBOT

AUTHORS:
RAMIREZ PEDRAZA, Ingrid Alexandra *

KEYWORDS:
GPS, NMEA0183, mobile robot.

DESCRIPTION:

The Grupo de Investigación en Mecatrónica (GIMKT) have put their attention on the field of mobile robots. These devices use to conduct their movements, satellite navigation systems to determine the position at a given moment, consists of a group of satellites orbiting the earth and receivers that receive data of position, velocity, etc.. .

The study of protocol data transmission NMEA0183 of global navigation systems is the first step in establishing the basis for future applications. The data that handles standard NMEA 0183 are organized through sentences that have a specific function that there are 3 basic classes: the emisor, owners of consultation. The first point to the identity of the sending device, the latter were created by different manufacturers of GPS, and finally the query issued by a receiver to request information in particular.

We search a freeware open source software, the application Visualgps NMEA PARSER DEMO (VGPS) was appropriate. This is a Windows application developed in Microsoft Visual C / C + + version 6.0, which shows data sentences providing basic information about latitude, longitude, altitude, signal quality and data from satellites in contact with the GPS, number of sentences received through a serial communication between GPS and PC. Based on the format of the sentences of the standard, the application was structured to analize a sentence by a state machine which seeks the beginning and end of the sentence, the address, data, calculating the checksum. Then the information is stored in the members created for each sentences, then called the appropriate role to extract data and conveniently be displayed in the main window of application.

---

*Degree Work.

** Facultad de Engineering Physical-Mechanical, School of Mechanical Engineering Ing. Jorge Enrique Meneses Flores

INTRODUCCIÓN

Las técnicas de posicionamiento y navegación por satélite y por GPS han evolucionado de manera vertiginosa desde su aparición, traspasando los fines para los que en un principio fueron diseñadas y llegando a utilizarse en aplicaciones que ni se imaginaron en el momento de su desarrollo. Los avances en el mundo de las comunicaciones y de los sistemas electrónicos, los que han permitido que estas técnicas lleguen a todo tipo de usuarios, traspasando la pequeña comunidad militar o científica para las que fueron diseñadas. Aunque el posicionamiento global nació con fines militares, continuamente se le están encontrando nuevos usos civiles

La tecnología GPS puede ser utilizada en cualquier lugar, menos en aquellos en los cuales es imposible recibir señal, como por ejemplo dentro de edificios, subterráneos o bajo el agua. En el aire, los GPS son utilizados par la navegación aérea tanto militar, comercial y general, en el mar son utilizados por aficionados a la náutica, pescadores y marinos profesionales. Las aplicaciones terrestres en cambio están más diversificadas. La comunidad científica, por ejemplo, utiliza la tecnología GPS para obtener datos de posición y tiempo muy precisos.

Cualquier necesidad que precise mantener un registro o control de ubicación o posición geográfica, encontrar un camino quizás en medio de condiciones hostiles o saber la dirección y velocidad con la cual se desplaza puede sacar provecho de los beneficios del Sistema de Posicionamiento Global.

Este informe abordará la información y fundamentos necesarios para diseños y desarrollos posteriores de aplicaciones que utilicen Sistemas de Posicionamiento Global. Pensando en las diferentes clases de lectores que

puede tener acceso a este informe, se ha decidido presentarlo estructurado en 4 secciones como se muestra a continuación:

Sección I: VISIÓN PRELIMINAR DEL PROYECTO

Esta sección es una inducción al trabajo realizado y pretende que el lector se entere de los aspectos más relevantes y los objetivos de este proyecto.

En el **capítulo 1** se dan a conocer los objetivos generales y específicos a desarrollar en este proyecto. La importancia y los alcances del proyecto se presentan en el **capítulo 2**, aclarando la importancia y necesidad de desarrollar este trabajo de monografía.

Sección II: DESARROLLO DEL PROYECTO

Esta sección le presenta al lector de una forma cronológica y paso a paso, el desarrollo del proyecto.

En el **capítulo 3**, se hace referencia a la búsqueda de la información sobre el protocolo de comunicación de datos o Norma NMEA 0183 utilizada en los Sistemas de Posicionamiento Global, además de como se realiza la comunicación entre un receptor GPS y un PC. Por último se refiere al modo como se diseño y organizó toda la información recolectada sobre esta norma.

En el **capítulo 4** se trata sobre el receptor GPS Garmin eTrex Summit que fue utlizado en este trabajo de monografía.

En el **capítulo 5** esta dedicado a la búsqueda del software freeware que cumpliera con los objetivos de este trabajo de monografía.

En el **capítulo 6** se hace una descripción de la documentación del código fuente del software que fue seleccionado para este trabajo.

El **capítulo 7** da a conocer las conclusiones generales y específicas del proyecto realizado.

En el **capítulo 8** se describen las recomendaciones para los futuros trabajos y proyectos que se desarrollen en la línea de robots móviles en el campo mecatrónico.

Sección III: FUNDAMENTACIÓN TEÓRICA

Esta sección muestra el resultado de un estudio bibliográfico de los diferentes temas relacionados con los sistemas de posicionamiento global y se aconseja, sea el principio de la lectura para aquellos lectores que no estén familiarizados con el tema tratado en este proyecto. A continuación se describe en forma breve el contenido de cada uno de los capítulos que comprende esta sección:

En el **capítulo 9**, se hace una breve introducción a los Sistemas de Posicionamiento Global (GPS), su funcionamiento, aplicaciones y usos.

Sección IV: ANEXOS DEL PROYECTO

En esta sección se anexan el documento con la recopilación a cerca de la Norma NMEA 0183, el cual es de gran utilidad para iniciar el proceso de aprendizaje sobre el protocolo de comunicación de datos satelital, además de la documentación del código fuente del software freeware seleccionado para interfase de comunicación entre un GPS y un PC.

1. OBJETIVOS

1.1 OBJETIVO GENERAL

Dar inicio a las líneas de investigación y desarrollo en el campo mecatrónico relacionadas con el área de robots móviles, mediante el desarrollo de proyectos que integren áreas de conocimiento de las Ingenierías Mecatrónica, Mecánica y Sistemas.

1.2 OBJETIVOS ESPECIFICOS

- Recopilar la información existente sobre la norma NMEA 0183 y organizar un documento que sea de información y fundamento sobre el formato de este protocolo de transmisión de datos.

- Obtener y probar un software freeware que sea una aplicación de interfase de comunicación entre un GPS y un computador personal (PC) en forma postprocesada o en tiempo real.

- Documentar un código fuente (lenguaje C) generado por el software mencionado en el numeral 1.2.2.

2. IMPORTANCIA Y ALCANCES DE PROYECTO

Para estar a la vanguardia con la tecnología, la Universidad Industrial de Santander y más concretamente el Grupo de Investigación en Mecatrónica (GIMKT) han puesto su atención sobre el campo de los vehículos autónomos o robots móviles. Estos dispositivos usan sistemas de navegación para realizar sus desplazamientos. Tienen dos clases de sistemas de navegación, el sistema local y el sistema global, el primero consta de sensores de aproximación, térmicos, fotoceldas, etc.… para hacer control de movimiento ampliamente desarrollado a nivel universitario con diversas aplicaciones como móviles siguiendo una determinada trayectoria, etc.

El segundo sistema ha tenido un avance tecnológico, iniciado con el uso de marcas, señales, luego teniendo a las estrellas como guía, también se elaboraron instrumentos como compases, cronómetros, giroscopios, pero en la década de los 60s se implementó un sistema de navegación satelital para conocer la posición en un determinado momento; este es el sistema de navegación más usado en la actualidad por su confiabilidad y exactitud, consta de un grupo de satélites orbitando la tierra y receptores que reciben los datos de posición, velocidad entre otros (GPS). Los datos recibidos pueden ser descargados posteriormente o en tiempo real. Para esto se utilizan protocolos de comunicación, el de más auge en este momento es el NMEA0183. Desgraciadamente este sistema de navegación ha sido poco explorado en nuestro ámbito universitario.

Los robots móviles que utilizan sistemas de navegación global tienen mejor autonomía en su desplazamiento, haciéndolos ser capaces de evitar obstáculos, seleccionando una mejor proyectaría y una notable inteligencia que hacen que se tengan en cuenta para muchas aplicaciones en múltiples

sectores como en la agricultura, la construcción, la minería, aeroespacial, militar, industrial, etc.

En el sector de la minería se utilizan para la localización exacta de yacimientos menores mediante el desplazamiento por lugares poco asequibles para los humanos; también son usados para la inspección y localización de fugas o problemas en plantas nucleares, generadores de vapor, o tanques de almacenamiento con contenido peligroso.

En el campo militar son utilizados para el reconocimiento de vehículos, ambulancias automatizadas, suministros de tropa y localización de minas o artefactos explosivos. En aplicación marítimas, cuando se requieres hacer inspecciones de plataformas de perforación, para la instalación y mantenimiento de cableado trasatlántico.

Como se puede apreciar el campo de aplicación de los robots móviles es muy diverso, dando por sentada la necesidad apremiante de obtener los conocimientos necesarios para poder manejar esta tecnología; razón de ser de este trabajo de monografía, ya que con ello se dará inicio a la línea de investigación y desarrollo en sistemas de navegación global para vehículos autónomos.

El conocimiento y estudio del protocolo de transmisión de datos NMEA0183 de los sistemas de navegación global es el primer paso para establecer las bases o fundamentos necesarios para realizar futuras aplicaciones específicas.

A continuación se describe paso a paso la forma como se desarrolló este trabajo de monografía.

Fundamentación teórica. En esta etapa se recopiló una gran cantidad de información de diferentes medios como son: Libros, revistas, papers e Internet referente al origen, conceptos y aplicaciones sobre sistemas de posicionamiento global (GPS) y de la norma NMEA0183. Esto con el fin de ubicar su vital importancia dentro del campo de la automatización.

Esta información ha sido clasificada de tal forma que el lector adquiera conocimientos básicos sobre la necesidad, origen, funcionalidad y aplicabilidad de los Sistemas de Posicionamiento Global y su protocolo de transmisión de datos.

Después de esta recopilación y clasificación se procedió a hacer un estudio detenido de manuales de usuario de dispositivo GPS, para tener un manejo acertado de estos dispositivos.

- Organización del documento que contenga la información y fundamentación del protocolo de transmisión de datos.

- Se realiza con el fin de dar a conocer el origen, su interfase, los formatos, significados y usos propios del protocolo NMEA 0183.

- Búsqueda de un software freeware cuya aplicación sea la comunicación entre un GPS y un computador personal que permita conocer el proceso de transmisión de datos del protocolo NMEA 0183.

- Seguimiento, estudio y documentación del código fuente del software elegido en la etapa anterior que sea fundamento para aplicaciones y/o proyectos posteriores.

3. ORGANIZACIÓN DEL DOCUMENTO SOBRE LA NORMA NMEA0183

En el presente capitulo se hace una explicación breve y concisa del proceso de de búsqueda y recopilación de la información correspondiente a la norma NMEA 0183; cabe observar, que esta recopilación es fundamental para poder entender como son transmitidos los datos mediante este protocolo.

3.1 BÚSQUEDA DE INFORMACIÓN SOBRE NMEA 0183

En la actualidad el porcentaje de la utilización de los Sistemas de Posicionamiento Global es alto, ya que existen diversos campos de uso como en la navegación tanto aérea como marítima, localización y rastreo, incluso en el área de mejoramiento corporal y entrenamiento deportivo. La gran variedad de dispositivos que se encuentran en el mercado dan gran acceso a esta tecnología.

Pero cuando se quiere saber a fondo como funciona la transmisión de datos en estos sistemas, se encuentran vacíos y/o malas interpretaciones en la información.

La búsqueda de información en el medio educativo e industrial sobre el protocolo de transmisión de datos no arrojó resultados favorables para el desarrollo de esta monografía, razón en la cual la búsqueda se centró en la Web.

Fue necesaria una búsqueda de las versiones existentes de la norma NMEA 0183 para seleccionar la información referente a la más actual.

Se realizaron comparaciones en los documentos encontrados, pues en la Web se publicaron varias interpretaciones para algunas sentencias. Además, existen algunas sentencias que son obsoletas y han dejado de ser usadas, como otras que su formato es desconocido.

3.2 FORMATO DE LA NORMA NMEA 0183

Los datos que maneja la norma o protocolo de comunicación NMEA 0183 se encuentran organizados por medio de sentencias que tienen una función específica.

Cada sentencia se presenta en formato ASCII y esta compuesta por los siguientes campos. Ver cuadro 1:

Cuadro 1. Campos que componen una sentencia NMEA0183

Símbolo	***Nombre***	***Función***
$	Código ASCII 36	Carácter de comienzo de sentencia
RMB	Campo de dirección	Indica la función o que clase de información contiene esta sentencia.
13.5,N,23	Campo de datos	Corresponde a la información que contiene la sentencia.
*3D	Checksum (opcional)	Confirma que los datos enviados son los mismos recibidos.
<CR><LF>	Retorno de carro, final de carrera	Caracteres de final de sentencia.

3.3 CLASES DE SENTENCIAS

En la norma existen 3 clases básicas de sentencias que son.

- Sentencias del emisor
- Sentencias propietarias
- Sentencias de consulta.

Las primeras indican la identidad de dispositivo emisor (quien envía la sentencia), las segundas fueron creadas por los diferentes manufactureros de GPS con funciones específicas que no están estipuladas en la norma, y por último, las sentencias de consultas emitidas por un receptor para solicitar una sentencia o información en particular.

3.4 COMUNICACIÓN ENTRE UN GPS Y UN PC

Usualmente se utiliza el enlace “serial” a través del puerto COM, por medio del protocolo RS232C. Donde un uno (1) es representado por un valor menor a 0.5 V y un cero (0) corresponde a un valor de 4 V. De esta forma una cadenas de 1 y 0 son envidados de ida y venida.

Cada dispositivo GPS trae consigo software para poder realizar transferencias de datos desde el GPS o desde el PC, como mapas, rutas, perfiles de elevación, etc...

Se necesitan condiciones especiales para realizar la comunicación y transmisión de datos, información que se encuentra consignada en el documento de recopilación de la norma NMEA 0183.

3.5 DISEÑO Y ORGANIZACIÓN

Observando la forma como esta compuesta la norma NMEA 0183, se pensó en organizar el documento dando inicialmente la información de los orígenes de la norma, su razón de ser y propósitos, para luego dar a conocer los parámetros técnicos y dispositivos que la componen.

Parte fundamental de este trabajo de monografía es la forma de comunicación física entre un dispositivo (GPS) que maneje la norma NMEA 0183 y un computador personal (PC), la cual es explicada claramente en el documento.

Por último y no menos importante está consignado el formato que utiliza la norma, los campos componentes de una sentencia en general, las clases de sentencias, y la explicación detallada de 97 sentencias de mayor utilización. Ver figura 1.

Figura 1. Explicación de los campos de una sentencia NMEA

$PGRME Estimated Position Error (*Error de posición estimada*)

$PGRME,x.x,M,x.x,M,x.x,M*hh<CR><LF>

1	Error horizontal estimado (HPE) (x.x)
2	Unidades, metros (M)
3	Error vertical estimado (VPE) (x.x)
4	Unidades, metros (M)
5	Error de posición equivalente esférica global (x.x)
6	Unidades, metros (M)
7	Checksum (*hh)

Después de la lectura del documento, se estará en capacidad de entender la esencia de la norma e identificar las partes más significativas para realizar futuras aplicaciones mecatrónicas.

3.5.1 GPS Garmin eTrex Summit. En el presente capitulo se hace una explicación breve de las características del dispositivo GPS eTrex Summit, el cuál fue utilizado para el afianzamiento en el uso de los Sistemas de Posicionamiento Global, además para la selección y prueba del software freeware que nos permitiera comunicación entre un GPS y un computador personal.

Este dispositivo GPS es un producto Garmin, marca líder en comunicación y navegación. Ver figura 2

Figura 2. Dispositivo receptor GPS Garmin eTrex Summit

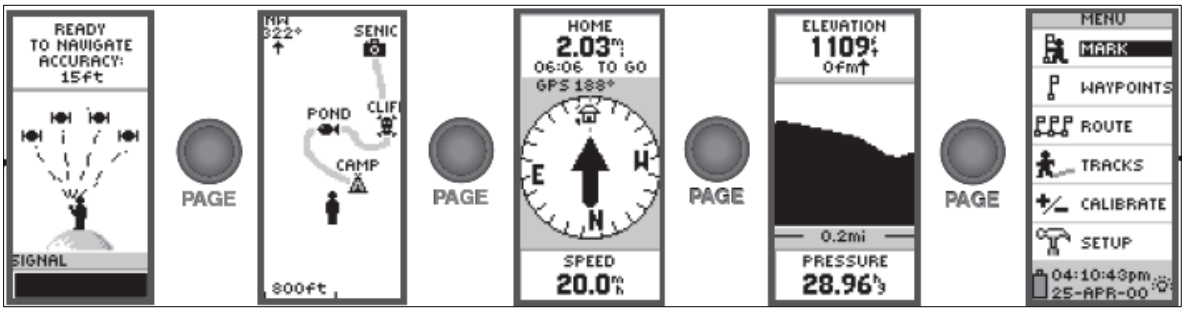
Fue diseñado tanto para uso recreativo como para marítimo. Contiene todas las funciones GPS y adicionalmente altímetro barométrico y brújula. Es de fácil manejo por su tamaño.

Posee un computador de viaje que proporciona datos sobre velocidad (actual, promedio, máxima), dirección, elevación, posición, odómetro entre

otras características. Los datos de elevación como ascenso y descenso total, promedio, máximo o mínimo, velocidad vertical y presión son generados por medio del computador de elevación.

Este GPS opera utilizando información recogida de satélites, para tener datos confiables debe haber recibido señal fuerte por lo menos de tres satélites para poder determinar su posición.

Figura 3. Páginas principales del GPS eTrex Summit.



Toda la información que puede suministrar el GPS es encontrada en cinco páginas principales que son: vista del cielo, mapa, aguja, elevación y menú. Ver figura 3.

En la página de vista del cielo, se puede observar de cuantos satélites estamos recibiendo señal, la calidad de señal (fuerte o débil) y la exactitud de la posición.

En la pantalla correspondiente al mapa se puede observar el trazado de un camino según el usuario realice un recorrido. Hay la opción de colocar marcaciones de camino (waypoint) para detallar el recorrido y facilitar la orientación.

La página de aguja entrega la información guía hacia un lugar destino cuando se esté dirigiendo hacia él.

La página de elevación proporciona la información sobre la variación de los datos de elevación que se presenten durante la trayectoria o recorrido, incluyendo el modo gráfico de esta variación.

La última página es la de menú, la cuál contiene opciones avanzadas de manejo del GPS. En este menú se encuentra la opción de la interfase para poder transmitir los datos obtenidos por el GPS. Ver figura 4.

Figura 4. Página de interfase de transmisión de datos

Para este caso se selecciona la opción “NMEA OUT” (salida en NMEA) que soporta la salida de datos en la norma NMEA 0183. El GPS tiene un cable de comunicación con el cuál se hace la descarga de datos a un PC, conectándolo al puerto serial del mismo.

4. SOFTWARE FREEWARE APLICACIÓN DE INTERFASE DE COMUNICACIÓN ENTRE UN GPS Y UN PC

En el presente capitulo se hace una explicación de las características que debe poseer el software elegido para una aplicación de interfase de comunicación entre un GPS y un PC, donde al realizarse una transmisión de datos, se observe el método de este proceso.

Según los objetivos propuestos en este trabajo de monografía, es necesario buscar un software que además de ser freeware sea "Open Source", es decir, que su código fuente este a disposición del público, para realizar un seguimiento de su programación y funcionamiento; esto es importante ya que se tienen bases confiables para realizar aplicaciones futuras.

La mayoría de los programas están digitados en lenguaje C, debido a la gran potencialidad que proporciona.

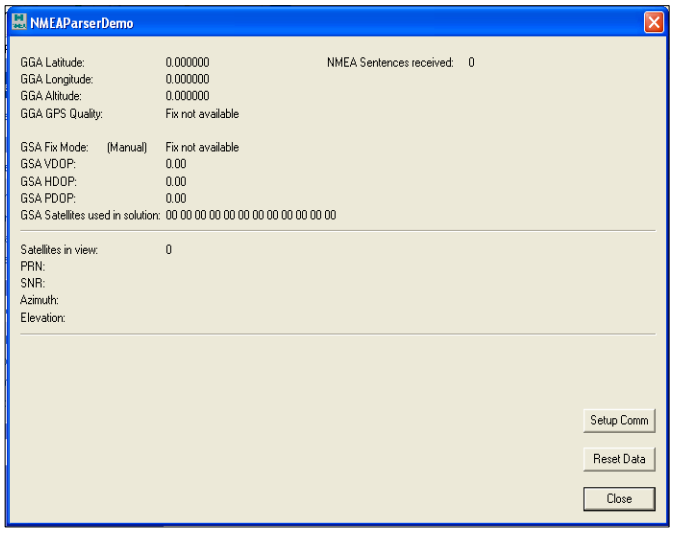
Durante la búsqueda de la aplicación, se encontraron varias que estaban programadas en C, pero que tenían restricciones en su código fuente.

La aplicación NMEA PARSER DEMO, realizada por programadores de la firma Visualgps (VGPS) fue la más apropiada para los fines de esta monografía. Quienes licencian el código fuente para que sea usado libremente en aplicaciones sobre este campo.

El NMEA PARSER DEMO es una aplicación Windows cuyo medio de desarrollo es Microsoft Visual C/C++ versión 6.0, donde en una pantalla principal se muestra datos de algunas sentencias básicas suministrando información sobre latitud, longitud, altitud, calidad de la señal recibida y datos

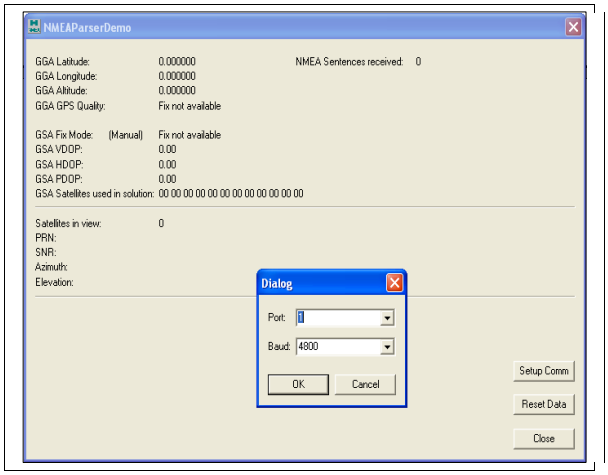
de los satélites en contacto con el GPS, número de sentencias recibidas del GPS, entre otros mediante una comunicación serial entre el GPS y el PC. Ver figura 5.

Figura 5. Ventana principal del NMEA PARSER DEMO



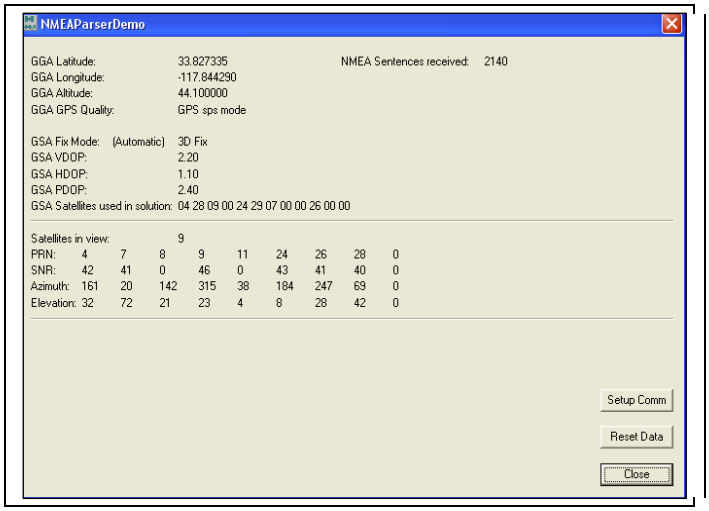
En esta misma pantalla existen 3 botones adicionales donde se puede configurar el puerto COM, reiniciar la toma de los datos y cerrar la aplicación. Ver figura 6.

Figura 6. Ventana de control del puerto COM



Una vez conectado al puerto COM por medio del cable de comunicación propio de cada dispositivo GPS, se inicia la aplicación haciendo doble clic en el archivo NMEA PARSER DEMO.EXE, se revisa la configuración del puerto que cumpla con los parámetros técnicos expresados por la norma NMEA 0183 y se verifica que el formato de salida de datos procedentes del GPS esté en el adecuado, así, se observará en la ventana principal de la aplicación los datos y la información transferida desde el GPS y solicitada por la aplicación, como se evidencia en la figura 7.

Figura 7. Funcionamiento del NMEA PARSER DEMO



Basado en el formato de las sentencias de la norma NMEA 0183, la aplicación fue estructurada para analizar una sentencia por medio de una máquina de estado en donde se busca el comienzo de la sentencia, el campo de dirección, los datos de la sentencia, el cálculo del checksum y el final de la sentencia. Luego la información del tipo de sentencia y sus correspondientes datos son almacenados en los miembros creados para cada sentencia para posteriormente se llame la función apropiada para extractar los datos convenientemente y ser mostrados en la ventana principal de la aplicación.

5. DOCUMENTACIÓN DEL CÓDIGO FUENTE DE LA APLICACIÓN NMEA PARSER DEMO

En el presente capitulo se hace una explicación del proceso de documentación del código fuente del NMEA PARSER DEMO, la cuál aclarará dudas y generará ideas para aplicaciones futuras teniendo en cuenta el uso de la tecnología de posicionamiento global en las áreas de mecatrónica, sistemas y mecánica.

Esta aplicación fue desarrollada en Microsoft Visual C/C++ versión 6.0, por el señor Monte Variakojis en el año 2002.

Microsoft Visual C++ tiene una aplicación llamada Visual Source Safe, la cuál impide que las aplicaciones sean compiladas paso a paso y que en este caso el programador utilizó. Pero haciendo un estudio y seguimiento de todos los archivos del código fuente se pudo conocer la estructura y la forma como funciona la aplicación.

La aplicación consta de 17 archivos más algunos que son generados por Microsoft Visual C++. De los cuales fueron documentados 10 siendo los más significativos para el objetivo de la aplicación.

La aplicación consta de un archivo principal, el cuál maneja las funciones correspondientes a la inicialización del puerto, la configuración del temporizador y la actualización de la información en la ventana principal de la aplicación, además de las acciones tomadas por el usuario por medio de los 3 botones disponibles.

El funcionamiento del NMEA PARSER DEMO es la siguiente, una vez iniciada la aplicación, se inicializa el puerto COM y se configura un temporizador para la lectura de datos desde el puerto cada 100 milisegundos, esto activa una serie de funciones como el traslado de los datos y almacenamiento de los mismos en el buffer del analizador, y la función de proceso de análisis de las sentencias recibidas según el formato que establece la norma NMEA 0183.

Adicional a estos eventos, se inicializa un segundo temporizador el cuál está encargado de actualizar el texto de la información presentada en la ventana principal de la aplicación; el tiempo de actualización de los datos es de 1000 milisegundos.

En esta aplicación se analizan 6 tipos de sentencias que son:

- PGGA, que suministra información sobre tiempo, latitud, longitud, altitud, número de satélites en uso y datos para determinar la posición del dispositivo GPS receptor.

- PGSA, que suministra información sobre los satélites activos y la dilución de posición (factor de corrección) debido a la geometría del satélite.

- PGSV, suministra información de los satélites vistos, azimut, elevación.

- PRMB y GPRMC, suministran información de navegación mínima recomendada.

- PZDA, suministra información sobre la hora local y la fecha.

La forma como fue documentado el código fuente de la aplicación fue de la siguiente manera, se tomaron los archivos significativos y se realizó

comentarios línea a línea, además de explicar la forma como la aplicación funciona.

Es necesario que el lector tenga conocimientos básicos de programación en lenguaje C, para que comprenda en su totalidad de los comentarios realizados.

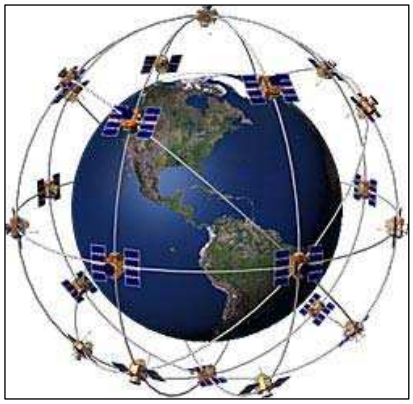
Adicionalmente, se hizo la traducción del paper publicado en noviembre de 2001 por el autor de la aplicación, el cuál fue una gran guía durante el proceso de documentación de la misma.

6. SISTEMAS DE POSICIONAMIENTO GLOBAL

En estos días de tantos cambios tecnológicos y del auge de la Tecnología de la Información nos encontramos con los Sistemas de Posicionamiento Global (GPS por su siglas en inglés), que a grandes rasgos es una red de 24 satélites, para de esta manera dar una cobertura total desde el espacio, hacia toda la superficie terrestre.

Esta constelación GPS consta de 6 órbitas, prácticamente circulares, con inclinación de 55 grados y uniformemente distribuidas en el plano del ecuador. Hay 4 satélites por órbita, uniformemente distribuidos y con altitud de 20180 Km., además un satélite logra 2 vueltas alrededor de la tierra, por cada 24 horas. Ver figura 8.

Figura 8. Constelación GPS.



Esta tecnología existe desde 1967 y fue desarrollada con fines militares por los Estados Unidos, pero la información tenía retraso de tiempo y fue hasta el año de 1978 que implantaron el sistema NAVSTAR (NAVigation Satellite Timing And Ranning.)

6.1 CONFIGURACIÓN DE UN SISTEMA GPS

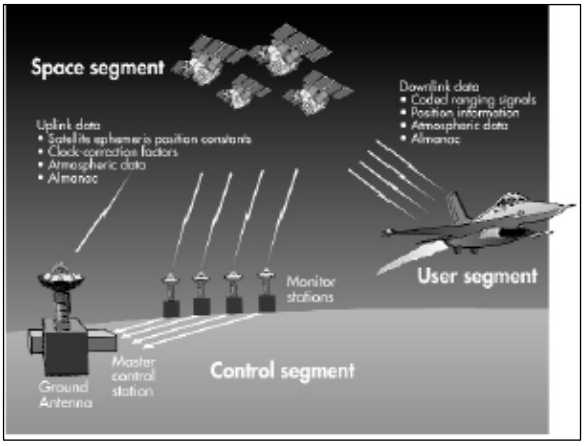
La Configuración del sistema GPS actual consta de 3 sectores:

Espacial, sobre el cual están todos los satélites ocupados para el seguimiento

Control, consta de 5 estaciones desde donde se controlan los satélites, se procesa la información y se sincronizan los relojes de cada satélite.

Usuario, comprende a los equipos utilizados por los usuarios finales, para conocer y medir alguna ubicación sobre la tierra.

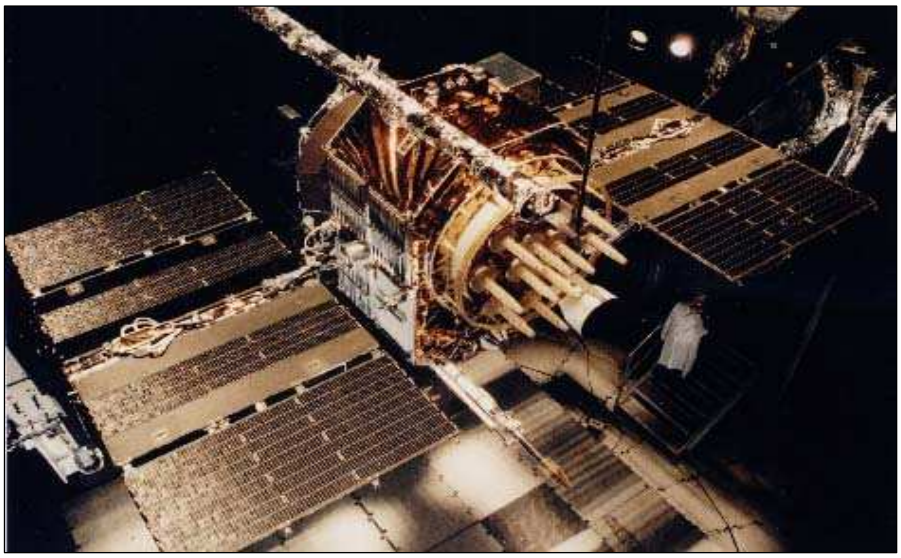
Figura 9. Configuración de un sistema GPS.



6.1.1 Sector Espacial. El sistema espacial completo de GPS incluye 24 satélites formado por 21 unidades operativas y 3 de repuesto localizadas aproximadamente 10200 Km. sobre la tierra. Ellas están posicionadas en tal forma que se puede recibir señal procedente de seis satélites cerca de un 100% del tiempo en cualquier punto de la tierra. Se necesitan muchas señales para conseguir la mejor información de posición

Los satélites están equipados con relojes de alta precisión que mantienen una precisión del tiempo de 3 nanosegundos. Esta precisión es importante debido a que el receptor debe conocer exactamente cuanto tiempo demora su señal llegar a cada satélite y retornar. Sabiendo la cantidad exacta de tiempo que la señal demora en ir y devolverse a cada satélite, se puede calcular la posición del receptor. Ver figura 10.

Figura 10. Satélite GPS



6.1.2 Sector Control. Este sector consiste de estaciones de monitoreo no asistido por humanos localizadas alrededor del mundo, Hawaii y Kwajalein en el Océano Pacífico, Diego García en el Océano Índico, la isla Ascención en el Océano Atlántico y una estación maestra en la Base Aérea de Falcon en Colorado como se observa en la figura 11 y cuatro estaciones con grandes antenas que transmiten las señales de los satélites. Las estaciones también rastrean y monitorean los satélites

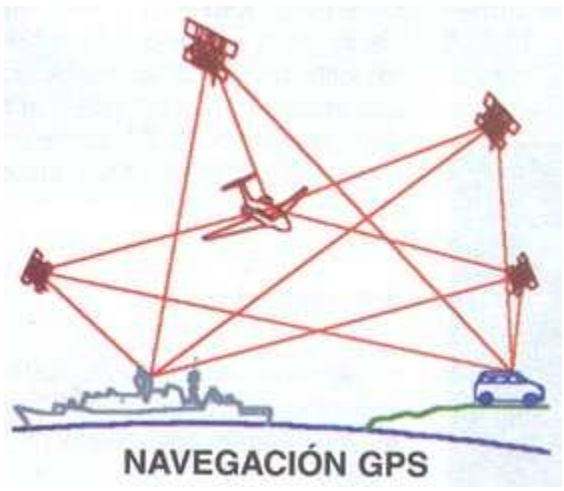
Figura 11. Estaciones de monitoreo localizadas alrededor del mundo.

Las estaciones de control miden las señales procedentes de los satélites, y son incorporadas en modelos orbitales para cada satélite. Los modelos calculan datos de ajuste de órbita (efemérides) y correcciones de los relojes de cada satélite. La estación maestra envía, a cada satélite, las efemérides y correcciones de reloj. Cada satélite envía, posteriormente, y mediante señales de radio, subconjuntos de estas informaciones a los receptores de GPS.

6.1.3 Sector Usuario. El segmento de usuario lo forman los receptores y la comunidad de usuarios. Los receptores convierten las señales recibidas de los satélites en posición, velocidad y tiempo estimados. Para el cálculo de la posición en cuatro dimensiones X, Y, Z y tiempo, se requieren cuatro satélites. Los receptores son utilizados para navegación, posicionamiento, estimaciones temporales y otras investigaciones.

Los receptores GPS pueden ser cargados en la mano o instalados en aeronaves, barcos, tanques, carros, submarinos, etc.. Como se aprecia en la figura 12.

Figura 12. Receptores GPS del sector usuario.



6.2 CÓMO FUNCIONA UN GPS

El receptor GPS funciona midiendo su distancia de los satélites, y usa esa información para computar su posición. Esta distancia se mide calculando el tiempo que la señal tarda en llegar a su posición, y basándose en el hecho de que la señal viaja a la velocidad de la luz, se puede calcular la distancia sabiendo la duración del viaje.

Cada satélite indica que el receptor se encuentra en un punto en la superficie de la esfera con centro en el propio satélite y de radio la distancia total hasta el receptor. Obteniendo información de dos satélites indica que el receptor se encuentra sobre la circunferencia que resulta cuando se interceptan las dos esferas. Si se adquiere la misma información de un tercer satélite notamos que la nueva esfera solo corta el círculo anterior en dos puntos.

Teniendo información de un el cuarto satélite, la cuarta esfera coincidirá con las tres anteriores en un único punto, y es en este momento cuando el receptor puede determinar una posición tridimensional, 3D (latitud, longitud y altitud).

Los receptores GPS pueden recibir, y habitualmente lo hacen, la señal de más de tres satélites para calcular su posición. En principio, cuantas más señales reciben, más exacto es el cálculo de posición.

6.2.1 Principios Básicos de Funcionamiento de un GPS

Los principios básicos de funcionamiento de un GPS son:

Triangulación. La base del GPS es la "triangulación" desde los satélites.

Distancias. Para "triangular", el receptor de GPS mide distancias utilizando **el tiempo de viaje de señales de radio.**

Tiempo. Para medir el tiempo de viaje de estas señales, el GPS necesita un control muy estricto del tiempo y lo logra con ciertos trucos.

Posición. Además de la distancia, el GPS necesita conocer exactamente donde se encuentran los satélites en el espacio. Orbitas de mucha altura y cuidadoso monitoreo, le permiten hacerlo.

Corrección. Finalmente el GPS debe corregir cualquier demora en el tiempo de viaje de la señal que esta pueda sufrir mientras atraviesa la atmósfera.
Se explicará cada uno de estos puntos en detalle.

La Triangulación desde los satélites. Aunque pueda parecer improbable, la idea general detrás del GPS es utilizar los satélites en el espacio como puntos de referencia para ubicaciones aquí en la tierra. Esto se logra mediante una muy, pero muy exacta, medición de nuestra distancia hacia al menos tres satélites, lo que nos permite "triangular" nuestra posición en cualquier parte de la tierra.

Supongamos que medimos nuestra distancia al primer satélite y resulta ser de 11.000 millas (20.000 Km.). Sabiendo que estamos a 11.000 millas de un satélite determinado, no podemos por lo tanto estar en cualquier punto del universo sino que esto limita nuestra posición a la superficie de una esfera que tiene como centro dicho satélite y cuyo radio es de 11.000 millas.

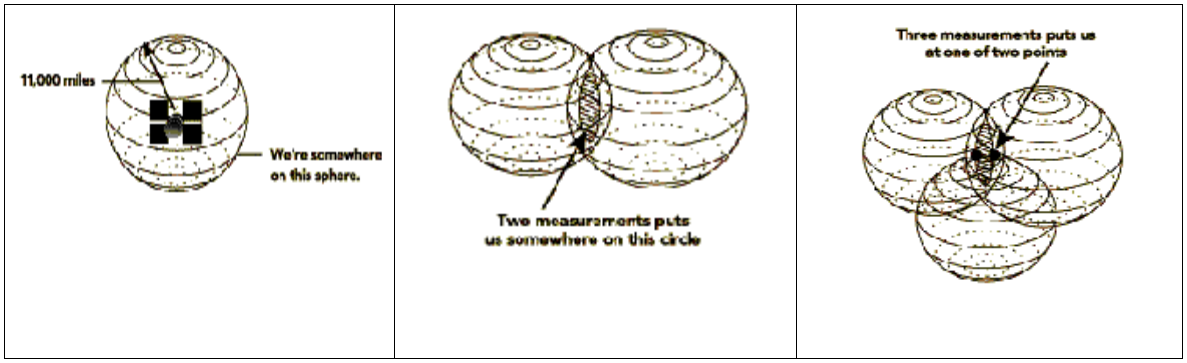
A continuación medimos nuestra distancia a un segundo satélite y estamos a 12.000 millas del mismo. Esto nos dice que no estamos solamente en la primera esfera, correspondiente al primer satélite, sino también sobre otra esfera que se encuentra a 12.000 millas del segundo satélite. En otras palabras, estamos en algún lugar de la circunferencia que resulta de la intersección de las dos esferas

Si ahora medimos nuestra distancia a un tercer satélite y estamos a 13.000 millas del mismo, esto limita nuestra posición aún más, a los dos puntos en los cuales la esfera de 13.000 millas corta la circunferencia que resulta de la intersección de las dos primeras esferas.

O sea, que midiendo nuestra distancia a tres satélites limitamos nuestro posicionamiento a solo dos puntos posibles. Para decidir cual de ellos es nuestra posición verdadera, podríamos efectuar una nueva medición a un cuarto satélite. Pero normalmente uno de los dos puntos posibles resulta ser muy improbable por su ubicación demasiado lejana de la superficie terrestre

y puede ser descartado sin necesidad de mediciones posteriores. Ver figura 13.

Figura 13. Triangulación de satélites



Medición de las Distancias a los satélites. En el caso del GPS estamos midiendo una señal de radio, que sabemos que viaja a la velocidad de la luz, alrededor de 300.000 Km. por segundo.

El problema de la medición de ese tiempo es complicado. Los tiempos son extremadamente cortos. Si el satélite estuviera justo sobre nuestras cabezas, a unos 20.000 Km. de altura, el tiempo total de viaje de la señal hacia nosotros sería de algo más de 0.06 segundos. Estamos necesitando relojes muy precisos.

Supongamos que nuestro GPS, por un lado, y el satélite, por otro, generan una señal auditiva en el mismo instante exacto. Supongamos también que nosotros, parados al lado de nuestro receptor de GPS, podamos oír ambas señales (Obviamente es imposible "oír" esas señales porque el sonido no se propaga en el vacío).

Oiríamos dos versiones de la señal. Una de ellas inmediatamente, la generada por nuestro receptor GPS y la otra con cierto atraso, la proveniente del satélite, porque tuvo que recorrer alrededor de 20.000 Km. para llegar hasta nosotros. Podemos decir que ambas señales no están sincronizadas.

Si quisiéramos saber cual es la magnitud de la demora de la señal proveniente del satélite podemos retardar la emisión de la señal de nuestro GPS hasta lograr la perfecta sincronización con la señal que viene del satélite.

El tiempo de retardo necesario para sincronizar ambas señales es igual al tiempo de viaje de la señal proveniente del satélite. Supongamos que sea de 0.06 segundos. Conociendo este tiempo, lo multiplicamos por la velocidad de la luz y ya obtenemos la distancia hasta el satélite.

Tiempo de retardo (0.06 seg) x Vel. de la luz (300.000 km/seg) = Dist. (18.000 km)

La señal emitida por nuestro GPS y por el satélite es algo llamado "Código Pseudo Aleatorio" (Pseudo Random Code). La palabra "Aleatorio" significa algo generado por el azar.

Este Código Pseudo Aleatorio es una parte fundamental del GPS. Físicamente solo se trata de una secuencia o código digital muy complicado. O sea una señal que contiene una sucesión muy complicada de pulsos "on" y "off", como se pueden ver en la figura 14:

Figura 14. Código Pseudo Aleatorio.

La señal es tan complicada que casi parece un ruido eléctrico generado por el azar. De allí su denominación de "Pseudo-Aleatorio". Hay varias y muy buenas razones para tal complejidad. La complejidad del código ayuda a asegurarnos que el receptor de GPS no se sintonice accidentalmente con alguna otra señal. Siendo el modelo tan complejo es altamente improbable que una señal cualquiera pueda tener exactamente la misma secuencia. Dado que cada uno de los satélites tiene su propio y único Código Pseudo Aleatorio, esta complejidad también garantiza que el receptor no se confunda accidentalmente de satélite. De esa manera, también es posible que todos los satélites trasmitan en la misma frecuencia sin interferirse mutuamente. Esto también complica a cualquiera que intente interferir el sistema desde el exterior al mismo. El Código Pseudo Aleatorio le da la posibilidad al Departamento de Defensa de EEUU de controlar el acceso al sistema GPS.
Pero hay otra razón para la complejidad del Código Pseudo Aleatorio, una razón que es crucial para conseguir un sistema GPS económico.

El código permite el uso de la "teoría de la información" para amplificar las señales de GPS. Por esa razón las débiles señales emitidas por los satélites pueden ser captadas por los receptores de GPS sin el uso de grandes antenas.

Control perfecto del Tiempo. Si la medición del tiempo de viaje de una señal de radio es clave para el GPS, los relojes que se emplean deben ser exactísimos, dado que si miden con un desvío de un milésimo de segundo, a la velocidad de la luz, ello se traduce en un error de 300 km. Por el lado de los satélites, el timing es casi perfecto porque llevan a bordo relojes atómicos de increíble precisión.

El secreto para obtener un timing tan perfecto es efectuar una medición satelital **adicional**. Resulta que si **tres mediciones perfectas** pueden posicionar un punto en un espacio tridimensional, **cuatro mediciones imperfectas** pueden lograr lo mismo.

Si todo fuera perfecto (es decir que los relojes de nuestros receptores GPS lo fueran), entonces todos los rangos (distancias) a los satélites se interceptarían en un único punto (que indica nuestra posición). Pero con relojes imperfectos, una cuarta medición efectuada como control cruzado, NO interceptará con los tres primeros. De esa manera la computadora de nuestro GPS detectará la discrepancia y atribuirá la diferencia a una sincronización imperfecta con la hora universal.

Dado que cualquier discrepancia con la hora universal afectará a las cuatro mediciones, el receptor buscará un factor de corrección único que siendo aplicado a sus mediciones de tiempo hará que los rangos coincidan en un solo punto. Dicha corrección permitirá al reloj del receptor ajustarse nuevamente a la hora universal. Una vez que el receptor de GPS aplica dicha corrección al resto de sus mediciones, obtenemos un posicionamiento preciso.

Una consecuencia de este principio es que cualquier GPS decente debe ser capaz de sintonizar al menos cuatro satélites de manera simultánea. En la práctica, casi todos los GPS en venta actualmente, acceden a más de 6, y hasta a 12, satélites simultáneamente. Ahora bien, con el Código Pseudo Aleatorgio como un pulso confiable para asegurar la medición correcta del tiempo de la señal y la medición adicional como elemento de sincronización con la hora universal, tenemos todo lo necesario para medir nuestra distancia a un satélite en el espacio.

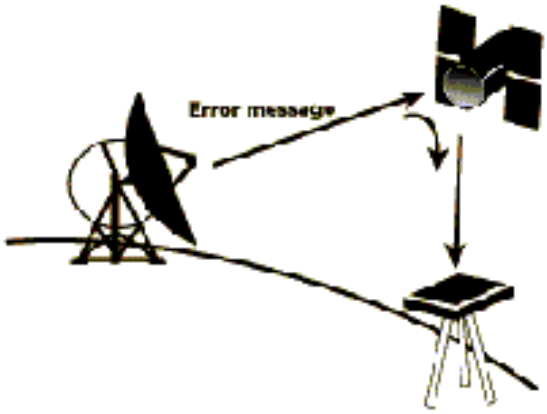
Posición de los satélites. La altura de 20.000 Km. es en realidad un gran beneficio para este caso, porque algo que está a esa altura, está bien despejado de la atmósfera. Eso significa que orbitará de manera regular y predecible mediante ecuaciones matemáticas sencillas. La Fuerza Aérea de los EEUU colocó cada satélite de GPS en una órbita muy precisa, de acuerdo al Plan Maestro de GPS. En tierra, todos los receptores de GPS tienen un almanaque programado en sus computadoras que les informan donde está cada satélite en el espacio, en cada momento. Las órbitas básicas son muy exactas pero con el fin de mantenerlas así, los satélites de GPS son monitoreados de manera constante por el Departamento de Defensa, ver figura 15.

Figura 15. Monitoreo permanente de satélites.

Ellos utilizan radares muy precisos para controlar constantemente la exacta altura, posición y velocidad de cada satélite. Los errores que ellos controlan son los llamados errores de efemérides, o sea evolución orbital de los satélites. Estos errores se generan por influencias gravitacionales del sol y de la luna y por la presión de la radiación solar sobre los satélites. Estos errores son generalmente muy sutiles pero si queremos una gran exactitud debemos tenerlos en cuenta.

Una vez que el Departamento de Defensa ha medido la posición exacta de un satélite, vuelven a enviar dicha información al propio satélite. De esa manera el satélite incluye su nueva posición corregida en la información que transmite a través de sus señales a los GPS.

Figura 16. Retroalimentación de la posición de un satélite.

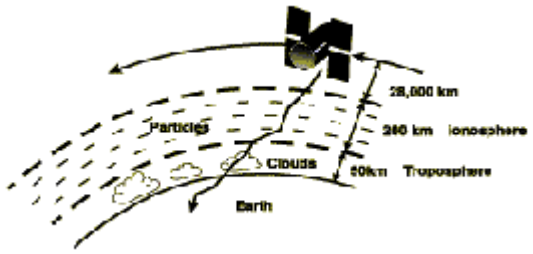


Esto significa que la señal que recibe un receptor de GPS no es solamente un Código Pseudo Aleatorio con fines de timing. También contiene un mensaje de navegación con información sobre la órbita exacta del satélite

Corrección de Errores. Hemos estado afirmando que podemos calcular la distancia a un satélite multiplicando el tiempo de viaje de su señal por la velocidad de la luz. Pero la velocidad de la luz sólo es constante en el vacío.

Una señal de GPS pasa a través de partículas cargadas en su paso por la ionosfera y luego al pasar a través de vapor de agua a la troposfera pierde algo de velocidad, creando el mismo efecto que un error de precisión en los relojes como se muestra en la figura 17.

Figura 17. Tránsito de la señal del GPS a través de la ionósfera.

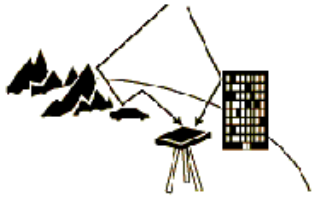


Hay un par de maneras de minimizar este tipo de error. Por un lado, podríamos predecir cual sería el error tipo de un día promedio. A esto se lo llama modelación y nos puede ayudar pero, por supuesto, las condiciones atmosféricas raramente se ajustan exactamente el promedio previsto.

Otra manera de manejar los errores inducidos por la atmósfera es comparar la velocidad relativa de dos señales diferentes. Esta medición de doble frecuencia es muy sofisticada y solo es posible en receptores GPS muy avanzados.

Los problemas para la señal de GPS no terminan cuando llega a la tierra. La señal puede rebotar varias veces debido a obstrucciones locales antes de ser captada por nuestro receptor GPS. Este error es similar al de las señales fantasma que podemos ver en la recepción de televisión. Los buenos receptores GPS utilizan sofisticados sistemas de rechazo para minimizar este problema. Ver figura 18.

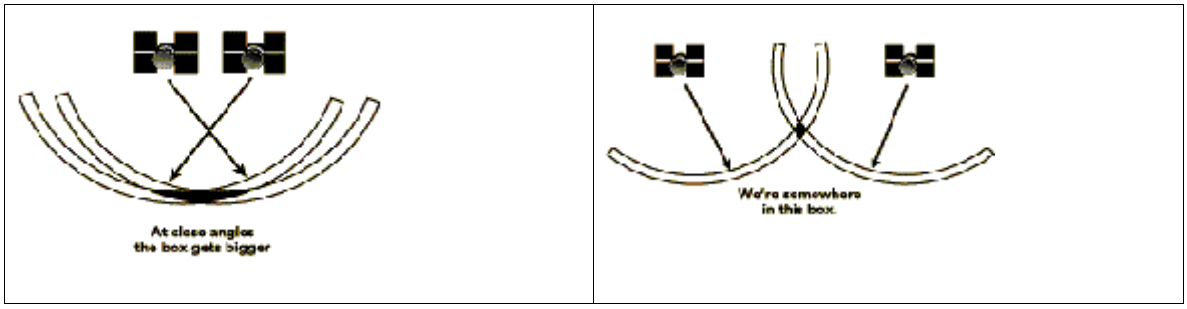
Figura 18. Rebote de la señal GPS.



Aún siendo los satélites muy sofisticados no tienen en cuenta minúsculos errores en el sistema. Los relojes atómicos que utilizan son muy, pero muy, precisos, pero no son perfectos. Pueden ocurrir minúsculas discrepancias que se transforman en errores de medición del tiempo de viaje de las señales. Y, aunque la posición de los satélites es controlada permanentemente, tampoco pueden ser controlados a cada segundo. De esa manera pequeñas variaciones de posición o de efemérides pueden ocurrir entre los tiempos de monitoreo.

La geometría básica por si misma puede magnificar estos errores mediante un principio denominado "Dilución Geométrica de la Precisión", o DGDP. En la realidad suele haber más satélites disponibles que los que el receptor GPS necesita para fijar una posición, de manera que el receptor toma algunos e ignora al resto. Si el receptor toma satélites que están muy juntos en el cielo, las circunferencias de intersección que definen la posición se cruzarán a ángulos con muy escasa diferencia entre sí. Esto incrementa el área gris o margen de error acerca de una posición. Si el receptor toma satélites que están ampliamente separados, las circunferencias interceptan a ángulos prácticamente rectos y ello minimiza el margen de error. Ver figura 19.

Figura 19. Errores por geometría.



Los buenos receptores son capaces de determinar cuales son los satélites que dan el menor error por Dilución Geométrica de la Precisión.

Aunque resulte difícil de creer, el mismo Gobierno que pudo gastar 12.000 Millones de dólares para desarrollar el sistema de navegación más exacto del mundo, está degradando intencionalmente su exactitud. Dicha política se denomina "Disponibilidad Selectiva" y pretende asegurar que ninguna fuerza hostil o grupo terrorista pueda utilizar el GPS para fabricar armas certeras.

Básicamente, el Departamento de Defensa introduce cierto "ruido" en los datos del reloj satelital, lo que a su vez se traduce en errores en los cálculos de posición. El Departamento de Defensa también puede enviar datos orbitales ligeramente erróneos a los satélites que estos reenvían a los receptores GPS como parte de la señal que emiten.

Estos errores en su conjunto son la mayor fuente unitaria de error del sistema GPS. Los receptores de uso militar utilizan una clave encriptada para eliminar la Disponibilidad Selectiva y son, por ello, mucho más exactos.

6.3 GPS DIFERENCIAL O DGPS

El DGPS o GPS diferencial se distingue del normal en que éste es mucho más preciso que es GPS. Es un sistema desarrollado por los fabricantes de receptores civiles para conseguir una precisión similar al sistema militar. Para ello, es necesario acoplar al receptor GPS otro tipo de receptor. Este "complemento" capta las señales emitidas por una red de radiobalizas situadas en estaciones costeras.

La utilización del sistema DGPS sólo es aplicable en la navegación marina, siendo especialmente útil en las maniobras de atraque con poca visibilidad.

Dentro de lo que llamamos DGPS hay dos sistemas que nos proporcionan una precisión aún mayor el WADGPS y el AUGPS. El WADGPS (DGPS de Área Extensa) es una mejora sobre el DGPS tradicional. Este sistema trata de eliminar la dependencia del error del usuario con respecto a la distancia a la estación de referencia mediante la medición del retardo ionosférico y las pseudodistancias a todos los satélites a la vista.

El mayor inconveniente de este sistema está en todo el procesado que debe efectuar el receptor para utilizar estos datos, además de ser capaz de recibir una señal proveniente de un satélite geoestacionario. El error oscila entre los 2-3 metros. El GPS Extendido (AUGPS) comprende el uso de estaciones monitoras del sistema, estaciones maestras para WADGPS, satélites geoestacionarios para la retransmisión de correcciones diferenciales y cualquier otro método que ayuda a mejorar la fiabilidad y precisión en tareas delicadas que así lo necesiten como, por ejemplo, los altímetros barométricos en los aviones.

6.4 APLICACIONES Y USOS DE LOS GPS

Los servicios de transporte utilizan GPS para realizar un seguimiento de su flota y acelerar las entregas. Las compañías de transporte equipan los buques cisterna y cargueros con GPS para su navegación, así como para registrar y controlar los movimientos de las embarcaciones.

Los pilotos civiles utilizan GPS para la navegación, fumigación aérea, topografía y fotografía aérea. Al utilizar la tecnología GPS para elaborar los planes de vuelo, las líneas aéreas ahorran millones de dólares. Los GPS se pueden utilizar para el aterrizaje instrumental, tanto en aeropuertos grandes como pequeños, y hacen posible la creación de nuevos sistemas de navegación aérea.

En los automóviles se están instalando GPS para que los conductores puedan saber dónde están y a la vez recibir indicaciones de dirección. En Japón, 500.000 automóviles ya incorporan un sistema de navegación basado en GPS.

En estudio de fenómenos atmosféricos. Cuando la señal GPS atraviesa la troposfera el vapor de agua, principal causante de los distintos fenómenos meteorológicos, modifica su velocidad de propagación. El posterior análisis de la señal GPS es de gran utilidad en la elaboración de modelos de predicción meteorológica.

En modelos geológicos y topográficos. Los geólogos comenzaron a aplicar el sistema GPS en los 80 para estudiar el movimiento lento y constante de las placas tectónicas, para la predicción de terremotos en regiones geológicamente activas. En topografía, el sistema GPS constituye una herramienta básica y fundamental para realizar el levantamiento de terrenos y los inventarios forestales y agrarios.

En el guiado de disminuidos físicos. Se están desarrollando sistemas GPS para ayuda en la navegación de invidentes por la ciudad [12]. En esta misma línea, la industria turística estudia la incorporación del sistema de localización en guiado de visitas turísticas a fin de optimizar los recorridos entre los distintos lugares de una ruta.

Navegación desasistida de vehículos. Se están incorporando sistemas DGPS como ayuda en barcos para maniobrar de forma precisa en zonas de intenso tráfico, en vehículos autónomos terrestres que realizan su actividad en entornos abiertos en tareas repetitivas, de vigilancia en medios hostiles (fuego, granadas, contaminación de cualquier tipo) y en todos aquellos móviles que realizan transporte de carga, tanto en agricultura como en minería o construcción. La alta precisión de las medidas ha permitido

importantes avances en el espacio en órbitas bajas y así tareas de alto riesgo de inspección, mantenimiento y ensamblaje de satélites artificiales pueden ahora realizarse mediante robots autónomos.

7. CONCLUSIONES

7.1 CONCLUSIONES GENERALES

Se cumplieron satisfactoriamente los objetivos propuestos en el plan de trabajo de monografía.

- Se obtuvo acceso a información sobre Sistemas de Posicionamiento Global, dispositivos receptores o GPS y sobre la norma NMEA 0183.
- Se consiguió una herramienta informática que se adaptara y cumpliera con los requisitos expuestos en los objetivos de este trabajo de monografía.
- Se elaboraron el documento de recopilación sobre la norma NMEA 0183, la documentación del código fuente de la aplicación NMEA PARSER DEMO y la traducción del paper sobre la misma.

7.2 CONCLUSIONES ESPECÍFICAS

El uso de sistemas globales de posicionamiento, es una alternativa reciente y llamativa por implementar en aplicaciones de robots móviles.

Pero es necesario tener el conocimiento y fundamentos teóricos antes de incursionar en el campo práctico.

Cualquier lector neófito al tener acceso al documento de recopilación de la norma NMEA 0183, adquirirá un conocimiento básico confiable sobre este protocolo de comunicación de datos y si tiene alguna experticia en programación se apoyará en la documentación de la aplicación NMEA

PARSER DEMO y fácilmente estará en capacidad de crear y diseñar aplicaciones en está área.

No obstante, estos son los primeros pasos en la iniciación de las líneas de investigación y desarrollo de aplicaciones con robots móviles, en donde su autonomía y notable “inteligencia” hacen que sean objeto de uso en múltiples aplicaciones industriales.

La universidad Industrial de Santander más específicamente el Grupo de Investigación en Mecatrónica (GIMKT), ha direccionado sus esfuerzos y objetivos en adquirir y desarrollar proyectos los cuales denoten que se encuentra a la vanguardia tecnológica.

8. RECOMENDACIONES

- Con el fin de desarrollar proyectos basados en robots autónomos, se ve necesario la adquisición de sensores ultrasónicos, infrarrojos, pda, gps diferencial y demás dispositivos.

- Es recomendable un estudio detallado del código fuente con el objetivo de se realice una mejor implementación del protocolo, cambiando los tipos de definición enum por estructuras o clases dejando en el NMEA Parser solo la descripción de los comandos, es decir, dejando todos los comandos como clases o estructuras independientes porque cada uno de esos comando tienen atributos multivalores propios , según los principios de programación orientado a objetos

- Se recomienda que el código se implemente en sistemas embebidos como agenda digital o PDA, celular de última tecnología que permiten mayor movilidad, son más livianos y económicos.

- Actualmente, esta tecnología se ha proyectado ampliamente a otros campos de aplicaciones como en la sincronización de señales, en sistemas de alarmas en transporte de sustancias de alto riesgo contaminante, en la guía de disminuidos físicos, en la agroindustria, entre otras aplicaciones, sería interesante orientar proyectos a estos campos.

BIBLIOGRAFÍA

BADDELEY Glenn. Sitio en la Web: http://GlennBaddeley-GPS-NMEAsentenceinformation.htm

BAUMBACH Torsten. Sitio en la Web: http://pandora.inf.uni-jena.de/ttbb/

BENNETT Peter. Sitio en la Web: http://vancouverwebpages.com/pub/peter/index.html

Garmin eTrex Summit. Manual de usuario y referencia. 2001

GOFAST MARINE. NMEA 0183. 2002

National Marine Electronics Association: http://www.nmea.org

NMEA 0183 Buffer. Actisense. 2003

PENÏN, Álvarez Miguel. Desarrollo de un sistema gestor de rutas de posicionamiento global por satélite. 2002

POZO-RUIZ A., RIBEIRO A., GARCIA-ALEGRE M.C., GARCÍA L., GUINEA D., SANDOVAL F. Sistema de posicionamiento global (gps): descripción, análisis de errores, aplicaciones y futuro.

VARIAKOJIS Monte. NMEA Parser Design. Visualgps.

ANEXOS

Anexo A. PROTOCOLO NMEA 0183

1. PROTOCOLO NMEA

La Asociación Electrónica Nacional Marina de Estados Unidos (NMEA, sigla en inglés) es una asociación sin ánimo de lucro de manufactureros, distribuidores, institutos educativos y personas interesados en equipos periféricos electrónicos marinos. Esta asociación ha desarrollado una especificación que define la interface entre varios equipos electrónicos marinos.

La norma permite equipos electrónicos de uso marino enviar información a computadores y a otros equipos. En 1980, un grupo de profesionales de la industrial se reunieron y desarrollaron un "lenguaje común" para aparatos marinos. El resultado fue la norma 0180 de la Asociación Electrónica Nacional Marina, (NMEA 0180), que fue direccionada a un problema que hizo que los lorans (sistemas de radio-navegación operada por la Guardia Costera Americana) y los auto-pilotos trabajasen juntos, y fue un éxito. En los siguientes años esta norma fue revisada, ampliada y actualizada (existiendo otra versión 0182) para un amplio rango de dispositivos electrónicos que son usados en la navegación dando como resultado la norma actual, NMEA 0183 que fue convirtiéndose lentamente en el método más común con el cual los aparatos eléctricos marítimos hablan entre sí. La norma especifica tanto las conexiones que conforman un sistema NMEA y el formato de las líneas de datos que llevan la información NMEA.

La norma NMEA 0183 es totalmente un esquema de transmisión de datos digital, usando unos (1) y ceros (0) en un formato binario para comunicar una representación digital de la información requerida como profundidad, velocidad, tiempo, etc… a un instrumento conectado.

2. INTERFASE ELÉCTRICA

Los aparatos NMEA 0183 son diseñados tanto como emisores o como receptores (algunos trabajan de las formas), empleando una interfase serial asincrónica con los siguientes parámetros:

Velocidad en Baudios	4800
Números de bits de datos	8 (bit 7 es cero)
Bits de parada	1 (ó más)
Paridad	Ninguna
Handshake	Ninguna

Los *emisores* son aquellos dispositivos fuentes de información que transmiten datos como son GPS, sonares de profundidad, etc. Los equipos que reciben estos datos como plotters son llamados *receptores*.

La norma permite un solo emisor y varios receptores en un circuito. Usted no puede conectar un número de emisores en una sola línea NMEA, como esto no esta permitido, ellos no están sincronizados y tratarán de “hablar” al mismo tiempo resultando en el daño de sus datos, y un posible desastre si su información es valiosa. La conexión de cableado recomendada es un par protegido, con tierra protegida solo en el emisor. La norma no específica el uso de un conector especial.

Nota: La nueva norma 0183 HS (HS = high speed) introducida en la versión 3.0 usa una interfase de 3 cables y una velocidad de 38400 baudios.

La interconexión normal requiere cable protegido de núcleo doble. Debe contener dos colores codificados y conductores de fibra. Este cable debe se conectado a la fuente de datos o el emisor usando conectores o terminales.

Mientras que los cables de datos son relativamente insensibles a las interferencias, evite ubicarse cerca de una antena de radio SSB, cables transductores de sonido, cargas de ignición de motores, cables principales de corriente alterna o suministros de corriente directa que manejen grandes cantidades de corriente y dispositivos que sean eléctricamente ruidosos como motores y luces fluorescentes.

Es recomendado que la salida del emisor cumpla con EIA RS-422, un sistema diferencial con dos líneas de señal A y B. Señales diferenciales no tienen referencia a tierra y son más inmunes al ruido. Sin embargo una línea con un solo final a un nivel TTL es aceptado. Los voltajes de la línea A corresponde a esos en el cable simple de TTL, mientras los voltajes de B son invertidos (cuando la salida A es +5 V, la salida B es 0 V, y viceversa. Esto es una operación RS-422 unipolar. En el modo bipolar son usados ±5 V). En cualquier caso, el circuito recomendado usa un opto acoplador con un adecuado circuito de protección para así reducir los cambios de interferencia y remover el problema del efecto de corrientes inversas en tierra.

La entrada debe ser aislada de la tierra del receptor. En la practica, el cable simple o el cable A del RS-422 puede ser directamente conectado a la entrada RS-232 del computador. De hecho, incluso muchos de los más recientes productos como receptores GPS de mano, no tienen una salida diferencial RS-422, sino solo una línea con TTL o un nivel de señal compatible con CMOS a 5 V.

Emisor: Bajo esta norma, los emisores son capaces de ser conductores diferenciales con EIA RS-422 (similar a RS-232, con la diferencia en que las señales diferenciales no tienen referencia a tierra y son más inmunes al ruido). En versiones anteriores de la norma NMEA 0183 permitía conductores con circuitos de final simple, por ejemplo 0 - +15 V DC. Por lo

tanto, todavía existen instrumentos que no tienen salidas de conductores diferenciales. Un simple emisor puede manejar múltiples receptores dentro de ciertas limitaciones de corriente.

Receptor: Estos receptores deben tener la entrada aislada de la tierra de los barcos. La opto acoplación (Convertir energía eléctrica a luz y retornarla a electricidad otra vez, eliminando el ruido), es generalmente usada para cumplir este requerimiento, limita la impedancia de la entrada. La norma especifica una resistencia mínima de entrada para receptores de 500 ohmios y la mayoría de los aparatos probablemente estarán cercanos a este valor.

3. CONECTANDO UN PC A SU SISTEMA NMEA 0183

La especificación NMEA usualmente usa el mismo enlace "serial" como se usa en los PC de uso diario. De hecho, aunque no es recomendado, la mayoría de los PC pueden leer NMEA directamente a través del puerto COM. Esto es peligroso, sin embargo, como picos de voltajes pueden presentarse debido a que las líneas pasan cerca de los motores de los radares, alternadores, etc., esto dañará el puerto de un PC portátil que ha sido solamente diseñados para uso domestico. Esto hace romper los requerimientos de la norma para la opto-acoplación entre el equipo, por lo tanto puede ser que las condiciones de garantía para este efecto sean solo para equipos de uso marítimo.

Para mantener segura la interfase a un sistema NMEA 0183 en un bote, lo mejor es obtener un convertidor opto-acoplado NMEA a RS232. Simplemente conecte el opto-acoplador al puerto del PC y conecte los cables al dispositivo que contiene NMEA 0183.

La norma RS232C, es una interfase aprobada por la Asociación de Industrias Electrónicas (EIA) para conectar dispositivos en forma serial. En 1987, la EIA liberó una nueva versión de la norma y cambio el nombre a EIA-232D y en 1991, la EIA se unió con la Asociación de Telecomunicaciones Industriales (TIA) y sacaron una nueva versión de la norma llamada EIA/TIA-232-E. Muchas personas, sin embargo, todavía se refieren a la norma como RS-232C o solamente RS-232.

La mayoría de los módems cumplen la norma EIA-232 y la mayoría de los computadores personales tiene un puerto EIA-232 para conectar un módem u otro aparato. La gente "normal" lo llama solamente puerto serial. Además de los módems, muchas pantallas, ratones e impresoras seriales son diseñados para conectarse a un puerto EIA-232.

Básicamente, una interfase 232 consiste de un cable de transmisión (Tx) y un cable de recepción (Rx) y algunos cables de señales de control. Los

datos son transmitidos por los cables como una serie de pulsos de código binario. Estos tienen valores eléctricos, por lo menos 4.0 V representa un cero (0) lógico y menos de 0.5 V representa un uno (1). De esta forma una cadena de 1 y 0 son enviados de ida y venida. Este envío es lento debido a que la comunicación tiene que ir en forma de serie (un bit después de otro) por el cable, de ahí el término “puerto serial”.

La norma EIA-232 soporta dos tipos de conectores, un conector tipo D de 25 pines (DB-25) y un de 9 pines (DB-9), los portátiles tendrán DB9 por cuestión de ahorro de espacio.

4. PROBANDO LA CONEXIÓN CON “HYPERTERMINAL” DE WINDOWS

Si usted no tiene todavía un software instalado, usted puede probar si los datos están entrando por el puerto serial, usando la aplicación HyperTerminal incluido en todos los sistemas operativos Windows desde la versión 95. Este es normalmente encontrado en la carpeta de Comunicaciones de la carpeta Accesorios.

Arranque la aplicación, y le preguntarán por una nueva conexión, seleccione un icono y dé un nombre a la comunicación. Seleccione “directo a Com x” en la caja de dialogo de configuración del puerto (donde x es el número del puerto com en el cual el dispositivo NMEA esta conectado) Luego seleccione 4800 bits por segundo, un bit para parada y otro para arrancada, sin paridad y ningún control de flujo en la siguiente ventana, luego haga click en Aceptar para conectar el puerto. Usted deberá ver cadenas NMEA mostrándose en la pantalla del HyperTerminal. Estos datos están en el formato propio de NMEA 0183 y es decodificado por programas de navegación para entregar al usuario todos los datos de navegación.

5. FORMATO GENERAL DE LA NORMA NMEA.

Todos los datos son transmitidos en forma de *sentencias*, solo es permitido en formato ASCII más CR (retorno del carro) y LF (final de línea). Cada sentencia comienza con un signo $ y finaliza con <CR><LF>. Los cinco caracteres inmediatamente después del $ son el campo de dirección. Este campo de dirección es interpretado según el tipo de sentencia (emisor, consulta o de propietario). Los caracteres dentro del campo de dirección están limitados por dígitos y letras mayúsculas. El campo de dirección no puede ser un campo nulo.

Múltiples campos de datos siguen a este campo de dirección y son delimitados por comas. Las sentencias no pueden ser más grandes de 80 caracteres de texto visible (incluyendo los finales de línea). Si algún dato

para un campo determinado no esta disponible, el campo simplemente es omitido, pero las comas que lo delimitarían son todavía enviadas sin espacio entre ellas.

El campo opcional checksum consiste de un "*" y dos dígitos hexagesimales que representan la O exclusiva de todos los caracteres entre los signos "$" y "*" sin incluirlos. Un checksum es requerido en algunas sentencias. Los datos pueden variar en la cantidad de precisión contenida en el mensaje. Por ejemplo el tiempo puede ser indicado en cifras decimales de un segundo o la posición puede ser mostrada con 3 o incluso con 4 dígitos después del punto decimal. Un ejemplo del formato es el siguiente:

```
$GPRMB,A,4.08,L,EGLL,EGLM,5130.02,N,00046.34,W,004.6,213.9,122.9,A*3D<CR><LF>
       1  2   3  4    5       6      7      8     9   10   11   12  13 14
```

GPRMB: Recomended minimum navigation information (información mínima de navegación recomendada)

N. de campo	Dato	Significado
1	A	Validity (Validación)
2	4.08	Off track (Fuera de travesía)
3	L	Steer left (L/R) (Guía de izquierda)
4	EGLL	Last waypoint (Último punto de camino)
5	EGLM	Next waypoint (Siguente punto de camino)
6	5130.02	Latitude of next waypoint (Latitud del siguiente punto de camino)
7	N	North/South (Norte/Sur)
8	00046.34	Longitude of next waypoint (Longitud del siguiente punto de camino)
9	W	East/West (Este/Oeste)
10	004.6	Range (Rango)
11	213.9	Bearing to waypt (Marcación del punto de camino)
12	122.9	Closing velocity (Velocidad de aproximación)
13	A	Validity (Validación)
14	*3D	Checksum

Hay tres clases básicas de sentencias: sentencias del emisor, sentencias propietarias y sentencias de consulta.

Sentencias del emisor: El formato general para estas sentencias es:

```
$ttsss,d1,d2,….<CR><LF>
```

Las dos primeras letras que suceden al símbolo $ son el identificador del emisor. Los siguientes tres caracteres (sss) son el identificador de la

sentencia, seguido por un número de campos de datos separados por comas, seguidos por un checksum opcional, y terminando con <CR><LF>. Los campos de datos son definidos únicamente por cada tipo de sentencia. Un ejemplo sería:

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

Donde,
"HC" especifica el emisor que ha sido un compás o brújula magnética.
"HDM" especifica el encabezamiento del mensaje.
"238" es el valor de encabezamiento, y
"M" designa el valor de encabezamiento como magnético.

Algunos ejemplos de nemónicos son:

Tabla 1. Nemónicos de identificación de dispositivos emisores.

AG	Autopilot - General	(Autopiloto general)
AP	Autopilot – Magnetic	(Autopiloto magnético)
CD	Communications – Digital Selective Calling (DSC)	(Comunicaciones – llamado digital selectivo)
CR	Communications – Receiver / Beacon Receiver	(Comunicaciones – receptor/receptor de guía)
CS	Communications – Satellite	(Comunicaciones – satélite)
CT	Communications – Radio-Telephone (MF/HF)	(Comunicaciones – Radio-teléfono banda (MF/HF))
CV	Communications – Radio-Telephone (VHF)	(Comunicaciones – Radio teléfono banda VHF)
CX	Communications – Scanning Receiver	(Comunicaciones – Receptor escaneador)
DE	DECCA Navigation (Navegación DECCA)	DECCA Navigation (Navegación DECCA)
DF	Direction Finder	(Buscador de dirección)
EC	Electronic Chart Display & Inforamation System (ECDIS)	(Sistema de información y muestra de cartas electrónicas)
EP	Emergency Position Indicating Beacon (EPIRB)	(Escaneador indicador de posición de emergencia)
ER	Engineroom Monitoring Systems	(Sistemas de monitoreo del salón de máquinas)
GP	Global Positioning System (GPS)	(Sistema de posicionamiento global)
HC	Heading Sensors, Compass, Magnetic	(Encabezado de sensores, brújulas magnéticos)
HE	Heading Sensors, Gyro, North Seeking	(Encabezado de sensores, giroscopio, buscador del Norte)
HN	Heading Sensors, Gyro, Non-north Seeking	(Encabezado de sensores, giroscopio, no buscador del Norte)
II	Integrated Instrumentation	(Instrumentación integrada)
IN	Integrated Navigation	(Navegación integrada)
LA	Loran A	(Loran A)
LC	Loran C	(Loran C)

OM	Omega Navigation receiver	(Receptor de navegación Omega)
P	Proprietary Code	(Código de propietario)
RA	RADAR and/or ARPA	(RADAR y/o ARPA)
SD	Sounder, Depth	(Profundidad)
SN	Electronic Positioning System, other/general	(Sistema de posicionamiento electrónico, otro/general)
SS	Sounder, Scanning	(escaneador, sonar)
TI	Turn Rate Indicator	(Indicador de velocidad)
TR	Transit Navigation System	(Sistema de tránsito de navegación)
VD	Velocity Sensor, Doppler, other/general	(Sensor de velocidad, Doppler, otros)
VM	Velocity Sensor, Speed Log, Water, Magnetic	(Sensor de velocidad, registro de velocidad, agua, Magnético)
VW	Velocity Sensor, Speed Log, Water, Mechanical	(Sensor de velocidad, registro de velocidad, agua, Mecánico)
WI	Weather Instruments	(Instrumentos de clima)
YX	Transducer	(Transductor)
ZA	Timekeepers, Time/Date, Atomic Clock	(Mantenedores de tiempo, tiempo/fecha, reloj atómico)
ZC	Timekeepers, Time/Date, Chronometer	(Mantenedores de tiempo, tiempo/fecha, cronómetro)
ZQ	Timekeepers, Time/Date, Quartz	(Mantenedores de tiempo, tiempo/fecha, reloj de cuarzo)
ZV	Timekeepers, Time/Date, Radio Update, WWV or WWVH	(Mantenedores de tiempo, tiempo/fecha, actualización por radio, WWW o WWWH)

Sentencias propietarias: La norma permite a los manufactureros individuales definir formatos de sentencias que no están predefinidos en la norma NMEA0183. El formato general de una sentencia propietaria es:

```
$PmmmA,df1,df2,…,<CR><LF>
```

Estas sentencias comienzas con "$P", luego los siguientes tres caracteres (mmm) que son la identificación del manufacturero. El quinto carácter en el campo de dirección es una letra (A-Z) el cual define el tipo específico del mensaje, seguido por los datos que el manufacturero desee, siguiendo el formato general de las sentencias. Un ejemplo sería:

```
$PGRME,15.0,M,45.0,M,25.0,M*22<CR><LF>
```

Donde,
"P" significa que es una sentencia propietaria
"GRM" es el ID del manufactor, en este caso Corporación Garmin
"E" es el tipo específico de esta sentencia: Estimated Position Error (Error estimado de posición)

15.0,M Error estimado en la posición horizontal (HPE) en metros (M)
45.0,M Error estimado en la posición vertical (VPE) en metros (M)
25.0,M Error global en la posición equivalente esférica en metros (M)

Algunos códigos de manufacturadores son:

Tabla 2. Códigos de manufactureros.

AAR	Asian American Resources
ACE	Auto-Comm Engineering Corporation
ACR	ACR Electronics, Inc.
ACS.	Arco Solar, Inc
ACT	Advanced Control Technology
AGI	Airguide Instrument Company
AHA	Autohelm of America
AIP	Aiphone Corporation
ALD.	Alden Electronics, Inc
AMR	AMR Systems
AMT	Airmar Technology
ANS	Antenna Specialists
ANX	Analytyx Electronic Systems
ANZ	Anschutz of America
APC	Apelco
APN	American Pioneer, Inc.
APX	Amperex, Inc.
AQC	Aqua-Chem, Inc.
AQD	Aquadynamics, Inc.
AQM	Aqua Meter Instrument Company
ASP	American Solar Power
ATE	Aetna Engineering
ATM	Atlantic Marketing Company, Inc.
ATR	Airtron
CCC	Coastal Communications Company
CCL	Coastal Climate Company
CCM	Coastal Communications
CDC	Cordic Company
CEC	Ceco Communications, Inc.
CHI	Charles Industries, Limited
CKM	Cinkel Marine Electronics Industries
CMA	Societe Nouvelle D'Equiment du Calvados
CMC	Coe Manufacturing Company
CME	Cushman Electronics, Inc.
CMP	C-Map, s.r.l.

ATV	Activation, Inc.
AVN	Advanced Navigation, Inc.
AWA	Awa New Zealand, Limited
BBL	BBL Industries, Inc.
BBR	BBR and Associates
BDV	Brisson Development, Inc.
BEC	Boat Electric Company
BGS	Barringer Geoservice
BGT	Brookes and Gatehouse, Inc.
BHE	BH Electronics
BHR	Bahr Technologies, Inc.
BLB	Bay Laboratories
BNI	Neil Brown Instrument Systems
BNS	Bowditch Navigation Systems
BRM	Mel Barr Company
BRY	Byrd Industries
BTH	Benthos, Inc.
BTK	Baltek Corporation
BTS	Boat Sentry, Inc.
BXA	Bendix-Avalex, Inc.
CAT	Catel
CBN	Cybernet Marine Products
CCA	Copal Corporation of America
DGC	Digicourse, Inc.
DME	Digital Marine Electronics Corp.
DMI	Datamarine International, Inc.
DNS	Dornier System GmbH
DNT	Del Norte Technology, Inc.
DPS	Danaplus, Inc.
DRL	R.L. Drake Company
DSC	Dynascan Corporation
DYN	Dynamote Corporation
DYT	Dytek Laboratories, Inc.
ESA	European Space Agency

CMS	Coastal Marine Sales Company
CMV	CourseMaster USA,
CNI	Continental Instruments
CNV	Coastal Navigator
CNX	Cynex Manufactoring Company
CPL	Computrol, Inc.
CPN	Compunav
CPS	Columbus Positioning, Inc.
CPT	CPT, Inc.
CRE	Crystal Electronics, Limited
CRO	The Caro Group
CRY	Crystek Crystals Corporation
CSI	Communication Systems International, Inc.
CSM	Comsat Maritime Services
CST	Cast, Inc.
CSV	Combined Services
CTA	Current Alternatives
CTB	Cetec Benmar
CTC	Cell-tech Communications
CTE	Castle Electronics
CTL	C-Tech, Limited
CWD	Cubic Western Data
CWV	Celwave R.F., Inc.
CYZ	cYz, Inc.
DCC	Dolphin Components Corporation
DEB	Debeg GmbH
DFI	Defender Industries, Inc.
GPI	Global Positioning Instrument Corporation
GRM	Garmin Corporation
GSC	Gold Star Company, Limited
GTO	Gro Electronics
GVE	Guest Corporation
GVT	Great Valley Technology
HAL	HAL Communications Corporation
HAR	Harris Corporation
HIG	Hy-Gain
HIT	Hi-Tec
HPK	Hewlett-Packard
HRC	Harco Manufacturing Company
HRT	Hart Systems, Inc.
HTI	Heart Interface, Inc.

EBC	Emergency Beacon Corporation
ECT	Echotec, Inc.
EFC	Efcom Communication Systems
EEV	EEV, Inc.
ELD	Electronic Devices, Inc.
EMC	Electric Motion Company
EMS	Electro Marine Systems, Inc.
ENA	Energy Analysts, Inc.
ENC	Encron, Inc.
EPM	Epsco Marine
EPT	Eastprint, Inc.
ERC	The Ericsson Corporation
FDN	Fluiddyne
FEC	Furuno Electric Company (??)
FHE	Fish Hawk Electronics
FJN	Jon Fluke Company
FMM	First Mate Marine Autopilots
FNT	Franklin Net and Twine, Limited
FRC	The Fredericks Company
FTG	T.G. Faria Corporation
FUJ	Fujitsu Ten Corporation of America
FUR	Furuno USA, Inc.
GAM	GRE America, Inc.
GCA	Gulf Cellular Associates
GES	Geostar Corporation
GFC	Graphic Controls Corporation
GIS	Galax Integrated Systems
KBE	KB Electronics, Ld.
KBM	Kennebec Marine Company
KLA	Klein Associates, Inc.
KMR	King Marine Radio Corporation
KNG	King Radio Corporation
KOD	Koden Electronics Company, Ltd.
KRP	Krupp International, Inc.
KVH	KVH Company
KYI	Kyocera International, Inc.
LAT	Latitude Corporation
LEC	Lorain Electronics Corporation
LMM	Lamarche Manufacturing Company
LRD	Lorad
LSE	Littlemore Scientific Engineering

HUL	Hull Electronics Company
HWM	Honeywell Marine Systems
ICO	Icom of America, Inc.
IFD	International Fishing Devices
IFI	Instruments for Industry
IME	Imperial Marine Equipment
IMI	I.M.I.
IMM	ITT MacKay Marine
IMP	Impulse Manufacturing, Inc.
IMT	International Marketing and Trading, Inc.
INM	Inmar Electro. and Sales, Inc.
INT	Intech, Inc.
IRT	Intera Technologies, Ltd.
IST	Innerspace Technology, Inc.
ITM	Intermarine Electronics, Inc.
ITR	Itera, Limited
JAN	Jan Crystals
JFR	Ray Jefferson
JMT	Japan Marine Telecommunications
JRC	Japan Radio Company, Inc.
JRI	J-R Industries, Inc.
JTC	J-Tech Associates, Inc.
JTR	Jotron Radiosearch, Ltd.
MOT	Motorola
MPN	Memphis Net and Twine Company, Inc.
MQS	Marquis Industries, Inc.
MRC	Marinecomp, Inc.
MRE	Morad Electronics Corporation
MRP	Mooring Products of New England
MRR	II Morrow, Inc.
MRS	Marine Radio Service
MSB	Mitsubishi Electric Company, Ltd.
MSE	Master Electronics
MSM	Master Mariner, Inc.
MST	Mesotech Systems, Ltd.
MTA	Marine Technical Associates
MTG	Marine Technical Assistance Group

LSP	Laser Plot, Inc.
LTF	Littlefuse, Inc.
LWR	Lowrance Electronics Corportation
MCL	Micrologic, Inc.
MDL	Medallion Instruments, Inc.
MEC	Marine Engine Center, Inc.
MEG	Maritec Engineering GmbH
MFR	Modern Products, Ltd
MFW	Frank W. Murphy Manufacturing
MGN	Magellan Corporation
MGS	MG Electronic Sales Corporation
MIE	Mieco, Inc.
MIM	Marconi International Marine Company
MLE	Martha Lake Electronics
MLN	Matlin Company
MLP	Marlin Products
MLT	Miller Technologies
MMB	Marsh-McBirney, Inc.
MME	Marks Marine Engineering
MMP	Metal Marine Pilot, Inc.
MMS	Mars Marine Systems
MNI	Micro-Now Instrument Company
MNT	Marine Technology
MNX	Marinex
ODN	Odin Electronics, Inc.
OIN	Ocean instruments, Inc.
OKI	Oki Electronic Industry Company
OLY	Navstar Limited (Polytechnic Electronics)
OMN	Omnetics
ORE	Ocean Research
OTK	Ocean Technology
PCE	Pace
PDM	Prodelco Marine Systems
PLA	Plath, C. Division of Litton
PLI	Pilot Instruments
PMI	Pernicka Marine Products
PMP	Pacific Marine Products
PRK	Perko, Inc.
PSM	Pearce-Simpson

MTK	Martech, Inc.
MTR	Mitre Corporation, Inc.
MTS	Mets, Inc.
MUR	Murata Erie North America
MVX	Magnavox Advanced Products and Systems Company
MXX	Maxxima Marine
MES	Marine Electronics Service, Inc.
NAT	Nautech, Limited
NEF	New England Fishing Gear, Inc.
NMR	Newmar
NGS	Navigation Sciences, Inc.
NOM	Nav-Com, Inc.
NOV	NovAtel Communications, Ltd.
NSM	Northstar Marine
NTK	Novatech Designs, Ltd.
NVC	Navico
NVO	Navionics, s.p.a.
NVS	Navstar
OAR	O.A.R. Corporation
ODE	Ocean Data Equipment Corporation
RTN	Robertson Tritech Nyaskaien A/S
SAI	SAIT, Inc.
SBR	Sea-Bird electronics, Inc.
SCR	Signalcrafters, Inc.
SEA	SEA
SEC	Sercel Electronics of Canada
SEP	Steel and Engine Products, Inc.
SFN	Seafarer Navigation International
SGC	SGC, Inc.
SIG	Signet, Inc.
SIM	Simrad,Inc
SKA	Skantek Corporation
SKP	Skipper Electronics A/S
SLI	Starlink, Inc.
SME	Shakespeare Marine Electronics
SMF	Seattle Marine and Fishing Supply Co.
SML	Simerl Instruments
SMI	Sperry Marine, Inc.

PTC	Petro-Com
PTG	P.T.I./Guest
PTH	Pathcom, Inc.
RAC	Racal Marine, Inc.
RAE	RCA Astro-Electronics
RAY	Raytheon Marine Company
RCA	RCA Service Company
RCH	Roach Engineering
RCI	Rochester Instruments, Inc.
RDI	Radar Devices
RDM	Ray-Dar Manufacturing Company
REC	Ross Engineering Company
RFP	Rolfite Products, Inc.
RGC	RCS Global Communications, Inc.
RGY	Regency Electronics, Inc.
RMR	RCA Missile and Surface Radar
RSL	Ross Laboratories, Inc.
RSM	Robertson-Shipmate, USA
RWI	Rockwell International
RME	Racal Marine Electronics
TCN	Trade Commission of Norway (THE)
TDL	Tideland Signal
THR	Thrane and Thrane A/A
TLS	Telesystems
TMT	Tamtech, Ltd.
TNL	Trimble Navigation
TRC	Tracor, Inc.
TSI	Techsonic Industries, Inc.
TTK	Talon Technology Corporation
TTS	Transtector Systems
TWC	Transworld Communications, Inc.
TXI	Texas Instruments, Inc.
UME	Umec
UNI	Uniden Corporation of America
UNP	Unipas, Inc.
UNF	Uniforce Electronics Company
VAN	Vanner, Inc.
VAR	Varian Eimac Associates

SNV	Starnav Corporation
SOM	Sound Marine Electronics, Inc.
SOV	Sell Overseas America
SPL	Spelmar
SPT	Sound Powered Telephone
SRD	SRD Labs
SRS	Scientific Radio Systems, Inc.
SRT	Standard Radio and Telefon AB
SSI	Sea Scout Industries
STC	Standard Communications
STI	Sea-Temp Instrument Corporation
STM	Si-Tex Marine Electronics
SVY	Savoy Electronics
SWI	Swoffer Marine Instruments, Inc.
SRS	Shipmate, Rauff & Sorensen, A/S
TBB	Thompson Brothers Boat Manufacturing Company
VCM	Videocom
VEX	Vexillar
VIS	Vessel Information Systems, Inc.
VMR	Vast Marketing Corporation
WAL	Walport USA
WBG	Westberg Manufacturing, Inc.
WEC	Westinghouse Electric Corporation
WHA	W-H Autopilots
WMM	Wait Manufacturing and Marine Sales Company
WMR	Wesmar Electronics
WNG	Winegard Company
WSE	Wilson Electronics Corporation
WTC	Watercom
WST	West Electronics Ltd.
YAS	Yaesu Electronics

Algunos ejemplos de nemónicos son:

Tabla 3. Sentencias propietarias de Garmin y Starlink Inc.

Sentencias propietarias de Garmin		
PGRMB	DGPS Beacon Information	Información del guía DGPS
PGRMC	Sensor configuration information	Información de la configuración del sensor
PGRMCE	Sensor Configuration Information Enquiry	Solicitud de información de la configuración del sensor
PGRMC1	Additional Sensor Configuration Information	Información adicional de la configuración del sensor
PGRMC1E	Additional Sensor Configuration Information Enquiry	Solicitud de la información adicional de la configuración del sensor
PGRME	Estimated Position Error	Error estimado de posición
PGRMF	Position Fix Sentence	Sentencia de posición fija
PGRMI	Sensor Initialization Information	Información de la inicialización del sensor
PGRMIE	Sensor Initialization Information Enquiry	Solicitud de la información de inicialización del sensor
PGRMM	Map Datum	Map Datum (Geiode)
PGRMO	Output Sentence Enable/Disable	Activación/Desactivación de la sentencia de salida
PGRMT	Sensor Status Information	Información del estado del sensor

PGRMV	3D Velocity	Velocidad en 3D
PGRMZ	Altitude Information	Información de altitud

Sentencias propietarias de Starlink Inc.		
PSLIB	Differental GPS Beacon Receiver Control	Control del receptor en un GPS diferencial

Sentencias de consulta: Esta sentencia es utilizada por un receptor para solicitar una sentencia particular de un emisor. El formato general es:

$ttllQ,sss, <CR> <LF>

Los dos primeros caracteres del campo de dirección son el identificador del emisor de la solicitud y los siguientes dos caracteres son el identificador del emisor del dispositivo que es sido consultado (receptor). El quinto carácter es siempre una “Q” que define el mensaje como una consulta. El siguiente campo (sss) contiene el nemónico de tres letras de la sentencia que sea solicitada. Un ejemplo es:

$CCGPQ,GGA<CR><LF>

Donde,
El dispositivo “CC” (computador) está solicitando la sentencia “GGA” de un dispositivo “GP” (una unidad GPS). El GPS entonces transmitirá esta sentencia una vez por segundo hasta que una consulta diferente es solicitada.

Los nemónicos de identificación del emisor se pueden observar en la tabla 1.

6. Formatos e identificadores de sentencias

3. **AAM** Waypoint Arrival Alarm (*Alarma de arribo en el punto de camino*)

$--AAM,A,A,x.x,N,c - - c*hh<CR><LF>

1 Estado booleano (A)
A = Registro del círculo de arribo
2 Estado booleano (A)
A = Perpendicular pasada en el punto de camino
3 Radio del círculo de arribo (x.x)
4 Unidad del radio, millas naúticas (N)
5 ID del punto de camino (c - - c)

6 Checksum (*hh)

4. **ALM** GPS Almanac Data (*Datos de almanaque del GPS*)

```
$--ALM,x.x,x.x,xx,x.x,hh,hhhh,hh,hhhh,hhhh,hhhhhh,hhhhhh,hhhhhh,hhhhhh,hhh,hhh*hh<CR><LF>
```

1 Número total de sentencias (x.x)
2 Número del mensaje (x.x)
3 Número PRN de satélite (01 al 32) (xx)
4 Número de semana en el GPS: fecha y tiempo en el GPS está computada como número de semanas. (x.x)
5 Profundidad SV, bits 17 – 24 de cada página del almanaque (hh)
6 Excentricidad (hhhh)
7 Tiempo de referencia del almanaque (hh)
8 Ángulo de inclinación (hhhh)
9 Rata de ascensión en línea recta (hhhh)
10 Ruta del eje semi mayor (hhhhhh)
11 Argumento de perilunio (omega) (hhhhhh)
12 Longitud del nodo de ascensión (hhhhhh)
13 Medida anómala (hhhhhh)
14 Parámetro de reloj F0 (hhh)
15 Parámetro de reloj F1 (hhh)
16 Checksum (*hh)

5. **APA** Autopilot Sentence “A” (*Sentencia de autopiloto “A”*)

```
$--APA,A,A,x.xx,L,N,A,A,xxx,M,c---c*hh<CR><LF>
```

1 Estado (A)
V = Destello en Loran - C o advertencia SNR
A = Bandera de advertencia general u otro sistema de navegación cuando una posición fiable no esta disponible
2 Estado (A)
V = Bandera de advertencia de bloqueo del ciclo del Loran – C
A = OK o no es usado
3 Magnitud del error de Travesía (x.xx)
4 Dirección de guía, Derecha o Izquierda (L o R) (a)
5 Unidades de travesía(Millas náuticas o Kilómetros) (N)
6 Estado (A)
A = Registro del círculo de arribo

7 Estado (A)
A = Perpendicular pasada por el punto de camino
8 Marcación del origen al destino (xxx)
9 M = Magnético, T = verdadero (T)
10 ID del punto de camino del destino (c - - c)
11 Checksum (*hh)

6. **APB** Autopilot Sentence “B” (*Sentencia de autopiloto “B”*)

$GPAPB,A,A,x.x,a,N,A,A,x.x,a,c - - c,x.x,a,x.x,a*hh<CR><LF>

1 Estado (A)
V = Destello en Loran - C o advertencia SNR
A = Bandera de advertencia general u otro sistema de navegación cuando una posición fiable no esta disponible
2 Estado (A)
V = Bandera de advertencia de bloqueo del ciclo del Loran – C
A = OK o no es usado
3 Magnitud del error de travesía (x.x)
4 Dirección de guía, Derecha o Izquierda (R o L) (a)
5 Unidades de travesía (Millas náuticas o Kilómetros) (N)
6 Estado (A)
A = Registro del círculo de arribo
7 Estado
A = Perpendicular pasada por el punto de camino
8 Marcación del origen al destino (x.x)
9 M = Magnético, T = verdadero (M) (a)
10 ID del punto de camino del destino (c - - c)
11 Marcación, de la posición actual al destino (x.x)
12 M = Magnético, T = verdadero (M) (a)
13 Encabezamiento para guía al punto de camino del destino (x.x)
14 M = Magnético, T = verdadero (M) (a)
15 Checksum (*hh)

7. **ASD** Autopilot System Data (*Datos del sistema de autopiloto*)

Formato desconocido

8. **BEC** Bearing & Distance to Waypoint – Dead Reckoning (*Marcación y distancia al punto de camino – cálculo muerto*)

$--BEC,hhmmss.ss,llll.ll,a,yyyyy.yy,a,x.x,T,x.x,M,x.x,N,c--c*hh<CR><LF>

1 Tiempo (UTC) (hhmmss.ss)
2 Latitud del punto de camino (llll.ll)
3 N = Norte , S = Sur (a)
4 Longitud del punto de camino (yyyyy.yy)
5 E = Este , W = Oeste (a)
6 Marcación, Verdadero (x)
7 T = verdadero (T)
8 Marcación, Magnético (x.x)
9 M = Magnético (M)
10 Millas náuticas (x.x)
11 N = Millas náuticas (N)
12 ID del punto de camino (c - - c)
13 Checksum (*hh)

2. **BOD** Bearing – Waypoint to Waypoint (*Marcación de punto de camino a punto de camino*)

$--BOD,x.x,T,x.x,M,c--c,c--c*hh<CR><LF>

1 Grados de marcación, VERDADERO (x.x)
2 T = Verdadero (T)
3 Grados de marcación, Magnético (x.x)
4 M = Magnético (M)
5 Al punto de camino (c - - c)
6 Del punto de camino (c - - c)
7 Checksum (*hh)

9. **BWC** Bearing and Distance to Waypoint – Latitude, N/S, Longitude, E/W, UTC, Status (*Marcación y distancia al punto de camino – Estado de Latitud, N/S, Longitud; E/W, UTC*)

$--BWC,hhmmss.ss,llll.ll,a,yyyyy.yy,a,x.x,T,x.x,M,x.x,N,c--c*hh<CR><LF>

1 Tiempo (UTC) (hhmmss.ss)
2 Latitud del punto de camino (llll.ll)
3 N = Norte, S = Sur (a)

4 Longitud del punto de camino (yyyyy.yy)
5 E = Este, W = Oeste (a)
6 Marcación, Verdadero (x.x)
7 T = Verdadero (T)
8 Marcación, Magnético (x.x)
9 M = Magnético (M)
10 Millas náuticas (x.x)
11 N = Millas náutica (x.x)
12 ID del punto de camino (c - - c)
13 Checksum (*hh)

10. **BWR** Bearing and Distance to Waypoint – Rhumb Line Latitude, N/S, Longitude, E/W,UTC, Status (*Marcación y distancia al punto de camino – Estado de latitud, N/S, Longitud; E/W, UTC de la línea de rumbo*)

$--BWR,hhmmss.ss,llll.ll,a,yyyyy.yy,a,x.x,T,x.x,M,x.x,N,c--c*hh<CR><LF>

1 Tiempo (UTC) (hhmmss.ss)
2 Latitud del punto de camino (llll.ll)
3 N = Norte, S = Sur (a)
4 Longitud del punto de camino (yyyyy.yy)
5 E = Este, W = Oeste (a)
6 Marcación, Verdadero (x.x)
7 T = Verdadero (T)
8 Marcación, Magnético (x.x)
9 M = Magnético (M)
10 Millas náuticas (x.x)
11 N = Millas náutica (x.x)
12 ID del punto de camino (c - - c)
13 Checksum (*hh)

3. **BWW** Bearing - Waypoint to Waypoint (*Marcación – Punto de camino a Punto de camino*)

$--BWW,x.x,T,x.x,M,c--c,c--c*hh<CR><LF>

1 Grados de marcación, Verdadero (x.x)
2 T = Verdadero (T)
3 Grados de marcación, Magnético (x.x)

4 M = Magnético (M)
5 Al punto de camino (c - - c)
6 Del punto de camino (c - - c)
7 Checksum (*hh)

4. **DBK** Depth Below Keel (*Profundidad bajo quilla*)

```
$--DBK,x.x,f,x.x,M,x.x,F*hh<CR><LF>
```

1 Profundidad en pies (x.x)

2 f = pies (f)

3 Profundidad en metros (x.x)

4 M = metros (M)

5 Profundidad en brazadas (x.x)

6 F = brazadas (F)

7 Checksume (*hh)

5. **DBS** Depth Below Surface (*Profundidad bajo superficie*)

```
$--DBS,x.x,f,x.x,M,x.x,F*hh<CR><LF>
```

1 Profundidad en pies (x.x)

2

f = pies (f)

3

Profundidad en metros (x.x)

4

M = metros (M)

5

Profundidad en brazadas (x.x)

6

F = brazadas (F)

7

Checksume (*hh)

6. **DBT** Depth below transducer (*Profundidad bajo transductor*)

$--DBT,x.x,f,x.x,M,x.x,F*hh<CR><LF>

1

Profundidad en pies (x.x)

2

f = pies (f)

3

Profundidad en metros (x.x)

4

M = metros (M)

5

Profundidad en brazadas (x.x)

6

F = brazadas (F)

7 Checksume (*hh)

7. **DCN** Decca Position (*Posición Decca*)

$--DCN,xx,cc,x.x,A,cc,x.x,A,cc,x.x,A,A,A,A,x.x,N,x*hh<CR><LF>

1 Identificador de cadena Decca (xx9
2 Identificador de zona roja (cc)
3 Posición de línea roja (x.x)
4 Estado de línea maestra roja (A)
5 Identificador de línea verde (cc)
6 Posición de línea verde (x.x)
7 Estado de línea maestra verde (A)
8 Identificador de línea púrpura (cc)
9 Posición de línea púrpura (x.x)
10 Estado de línea maestra púrpura (A)
11 Uso de navegación de la línea roja (A)
12 Uso de navegación de la línea verde (A)
13 Uso de navegación de la línea púrpura (A)
14 Posición con incertidumbre (x.x)
15 N = millas nautícas (N)
16 Base de datos fija (x)
1 = Patrón normal
2 = Patrón de identificación de ruta
3 = Transmisiones de identificación de ruta
17 Checksum (*hh)

8. **DPT** Heading - Deviation & Variation (*Encabezado – Desviación y variación*)

$--DPT,x.x,x.x*hh<CR><LF>

1 Profundidad en metros (x.x)
2 Offset del transductor
- Positivo significa distancia desde el transductor a la línea de agua
- Negativo significa distancia desde el transductor a la quilla
3 Checksum

8. **DSC** Digital Selective Calling Information (*Información de llamado digital selectivo*)

Formato desconocido

9. **DSE** Extended DSC (*DSC extendido*)

Formato desconocido

10. **DSI** DSC Transponder Initiate (*Inicialización de un transponder DSC*)

Formato desconocido

11. **DSR** DSC transponder Response (*Respuesta de un transponder DSC*)

Formato desconocido

12. **DTM** Datum Reference (*Referencia de Datum*)

Formato desconocido

FSI Frequency Set Information (*Información del conjunto de frecuencia*)

$--FSI,xxxxxx,xxxxxx,c,x*hh<CR><LF>

1 Frecuencia transmitida (xxxxxx)
2 Frecuencia recibida (xxxxxx)
3 Modo de comunicaciones (sintaxis 2 del NMEA) (c)
4 Nivel de potencia (x)
5 Checksum (*hh)

13. **GBS** GPS Satellite Fault Detection (*Detección por defecto de satélite del GPS*)

Formato desconocido

9. **GGA** Global Positioning System Fix Data, Time, Position and fix related data for a GPS receiver. (*Datos de localización del sistema de posicionamiento global, datos relacionados de tiempo, posición y localización para un receptor GPS*)

```
$--GGA,hhmmss.ss,llll.ll,a,yyyyy.yy,a,x,xx,x.x,x.x,M,x.x,M,x.x,xxxx*hh<CR><LF>
```

1 Tiempo (UTC) (hhmmss.ss)
2 Latitiud (llll.ll)
3 N o S (Norte o Sur) (a)
4 Longitud (yyyyy.yy)
5 E o W (Este u Oeste) (a)
6 Indicador de calidad del GPS (x)
0 = Localización no disponible
1 = Localización del GPS
2 = Localización de GPS diferencial
7 Número de satélites visibles, 00 – 12 (xx)
8 Dilución de precisión horizontal (x.x)
9 Altitud de antena por encima/por debajo del nivel promedio del mar (geoid) (x.x)
10 Unidades de altitud de antena, en metros (M)
11 Separación geodésica, la diferencia entre la elipsoide terrestre WSG-84 y el nivel promedio del mar (geoide), “-“ significa nivel promedio del mar debajo del elipsoide (x.x)
12 Unidades de separación geodésica, en metros (M)
13 Edad de datos de GPS diferencial, tiempo en segundos desde la última actualización tipo 1 o 9 de SC104 , campo nulo cuando un DGPS no es usado (x.x)
14 ID de la estación de referencia diferencial, 0000-1023 (xxxx)
15 Checksum (*hh)

10. **GLC** Geographic Position, Loran-C (Posición geográfica, Loran-C)

```
$--GLC,xxxx,x.x,a,x.x,a,x.x,a,x.x,a,x.x,a,x.x,a*hh<CR><LF>
```

1 GRI microsegundo/10 (xxxx)
2 Microsegundos TOA maestro (x.x)
3 Estado de señal TOA maestro (a)
4 Diferencia de tiempo 1 microsegundo (x.x)
5 Estado de señal de diferencia de tiempo 1 (a)
6 Diferencia de tiempo 2 microsegundos (x.x)
7 Estado de señal de diferencia de tiempo 2 (a)
8 Diferencia de tiempo 3 microsegundos (x.x)
9 Estado de señal de diferencia de tiempo 3 (a)
10 Diferencia de tiempo 4 microsegundos (x.x)
11 Estado de señal de diferencia de tiempo 4 (a)
12 Diferencia de tiempo 5 microsegundos (x.x)
13 Estado de señal de diferencia de tiempo 5 (a)
14 Checksum (*hh)

11. **GLL** Geographic Position - Latitude/Longitude (*Posición geográfica – Latitiud/Longitud*)

$--GLL,llll.ll,a,yyyyy.yy,a,hhmmss.ss,A*hh<CR><LF>

1 Latitud (llll.ll)
2 N o S (Norte o Sur) (a)
3 Longitud (yyyyy.yy)
4 E o W (Este u Oeste)
5 Tiempo (UTC) (hhmmss.ss)
6 Estado A – Dato válido (A)
Estado V – Dato inválido
7 Checksum (*hh)

12. **GRS** GPS Range Residuals (*Rangos residuales de GPS*)

$--GRS,hhmmss.ss,x,x.x,x.x,x.x,.....,*hh<CR><LF>

1 Tiempo (UTC) (hhmmss.ss)
2 Modo (x)
0 = Residuales usados en GGA
1 = Residuales calculados después de GGA
3 Residuo del satélite 1 (metros) (x.x)
4 Residuo del satélite 2 (metros), el orden corresponde a los número PRN en la sentencia GSA (x.x)
5 Residuo del satélite 3 (metros) (x.x)
6 - 14 Entradas sin uso

15 Checksum (*hh)

13. **GSA** GPS DOP and active satellites (*GPS DOP y satélites activos*)

$--GSA,a,a,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x.x,x.x,x.x*hh<CR><LF>

1 Selección del modo (a) Autómatico (A) o manual (M)
2 Modo (a)
3 ID del satélite 1 usado para localización (x)
4 ID del satélite 2 usado para localización (x)
5 - 13 IDs de los satélites 5 – 12 para localización (x)
14 ID del satélite 12 usado para localización (x)
15 PDOP en metros (x.x)
16 HDOP en metros (x.x)
17 VDOP en metros (x.x)
18 Checksum (*hh)

14. **GST** Pseudorange Noise Statistics (*Estadísticas de ruido seudorango*)

$--GST,hhmmss.ss,x.x,x.x,x.x,xx.x,x.x,x.x,xx.x*hh<CR><LF>

1 Time (UTC) (hhmmss.ss)
2 Desviación RMS (x.x)
3 Desviación del eje semi mayor de la elipse de error, en metros (x.x)
4 Desviación del eje semi menor de la elipse de error, en metros (x.x)
5 Orientación de eje semi mayor de la elipse de error, en grados del norte verdadero (xx.x)
6 Desviación del error de latitud en metros (x.x)
7 Desviación del error de longitud en metros (x.x)
8 Desviación del error de altitud en metros (xx.x)
9 Checksum (*hh)

15. **GSV** Satellites in view (*Satélites vistos*)

$--GSV,x,x,x,x,x,x,x,...*hh<CR><LF>

1 Total de número de mensajes que contiene la información de los satelites vistos (x), cada sentencia da la información de 4 satélites.
2 Número de mensaje (x), que indica el número de mensaje dentro del ciclo o paquete de sentencias GSV que dan la información cabalmente.
3 Número total de satélites vistos (x)
4 ID de satélite (PRN) (x)
5 Elevación en grados (x)
6 Azimuth en grados desde el norte verdadero (x)
7 SNR en dB (x)
Información del segundo, tercer y cuarto satélites como el número y el SNR, campos 4 - 7
n Checksum (*hh)

14. **GTD** Geographic Location in Time Differences (*Localización geográfica en diferencias de tiempo*)

```
$--GTD,x.x,x.x,x.x,x.x,x.x*hh<CR><LF>
```

1 Diferencia de tiempo (x.x)
2 Diferencia de tiempo (x.x)
3 Diferencia de tiempo (x.x)
4 Diferencia de tiempo (x.x)
5 Diferencia de tiempo (x.x)
6 Checksum (*hh)

15. **GXA** TRANSIT Position - Latitude/Longitude - Location and time of TRANSIT fix at waypoint (*Posición de tránsito – Latitud/Longitud – Localización y tiempo de localización de tránsito en el punto de camino*)

```
$--GXA,hhmmss.ss,llll.ll,a,yyyyy.yy,a,c--c,x*hh<CR><LF>
```

1 Tiempo (UTC) de localización de posición (hhmmss.ss)
2 Latitud (llll.ll)
3 Este u Oeste (a)
4 Longitud (yyyyy.yy)
5 Norte o Sur (a)
6 ID del punto de camino (c - - c)

7 Número del satélite (x)
8 Checksum (*hh)

16. **HDG** Heading - Deviation & Variation (*Encabezado – Desviación y variación*)

$--HDG,x.x,x.x,a,x.x,a*hh<CR><LF>

1 Encabezado del sensor magnético en grados (x.x)
2 Desviación magnética en grados (x.x)
3 Dirección de la desviación magnética, (a)
E = proveniente del Este
W = proveniente del Oeste
4 Grados de la variación magnética (x.x)
5 Dirección de la variación magnética (a)
E = proveniente del Este
W = proveniente del Oeste
6 Checksum (*hh)

17. **HDM** Heading - Magnetic (*Encabezado – Magnético*)

$--HDM,x.x,M*hh<CR><LF>

1 Grados de encabezado, magnético (x.x)
2 M = magnético (M)
3 Checksum (*hh)

18. **HDT** Heading - True (*Encabezado – Verdadero*)

$--HDT,x.x,T*hh<CR><LF>

1 Grados de encabezamiento, verdadero (x.x)
2 T = verdadero (T)
3 Checksum (*hh)

19. **HSC** Heading Steering Command (*Comando de guía de encabezado*)

$--HSC,x.x,T,x.x,M,*hh<CR><LF>

1 Grados del encabezado, Verdadero (x.x)
2 T = verdadero (T)
3 Grados del encabezado, Magnético (x.x)
4 M = Magnético (M)
5 Checksum (*hh)

20. **LCD** Loran-C Signal Data (*Datos de señal Loran-C*)

$--LCD,xxxx,xxx,xxx,xxx,xxx,xxx,xxx,xxx,xxx,xxx,xxx,xxx,xxx*hh<CR><LF>

1 Microsegundos/10 GRI (xxxx)
2 SNR maestro relativo (xxx)
3 ECD maestro relativo (xxx)
4 Diferencia de tiempo 1 microsegundos (xxx)
5 Estado de señal de diferencia de tiempo 1 (xxx)
6 Diferencia de tiempo 2 microsegundos (xxx)
7 Estado de señal de diferencia de tiempo 2 (xxx)
8 Diferencia de tiempo 3 microsegundos (xxx)
9 Estado de señal de diferencia de tiempo 3 (xxx)
10 Diferencia de tiempo 4 microsegundos (xxx)
11 Estado de señal de diferencia de tiempo 4 (xxx)
12 Diferencia de tiempo 5 microsegundos (xxx)
13 Estado de señal de diferencia de tiempo 5 (xxx)
14 Checksum (*hh)

16. **MSK** MSK Receiver Interface (for DGPS Beacon Receivers) (*Interface de receptor de MSK (para receptores DGPS)*)

$--MSK,xxx.x,xx,xxx,xx,N*hh<CR><LF>

1 Frecuencia en kHz (283.5 a 325.0) (xxx.x)
2 Selección de frecuencia (xx)
M1 = Manual
A1 = Automática (cuando el campo 1 esta vacío)
3 Rata de bits MSK (100 a 200) (xxx)

4 Selección de la rata de bits (xx)
M2 = Manual
A2 = Automática (cuando el campo 3 esta vacío)
5 Período de salida del mensaje de estado de ejecución, 0 a 100 ($CRMSS) (N)
6 Checksum (*hh)

17. **MSS** MSK Receiver Signal Status (*Estado de señal del receptor MSK*)

$--MSS,x.x,x.x,x.x,xx.x,*hh<CR><LF>

1 Fuerza de señal (dB 1□V) (x.x)
2 SNR (dB) (x.x)
3 Frecuencia del receptor kHz (x.x)
4 Rata de datos (BPS) (xx)
5 Campo desconocido enviado por el receptor usado para prueba (x)
6 Checksum (*hh)

18. **MWD** Wind Direction & Speed (*Velocidad y dirección del viento*)

Formato desconocido

21. **MTW** Water Temperature (*Temperatura del agua*)

$--MTW,x.x,C*hh<CR><LF>

1 Grados (x.x)
2 Unidad de medida, Celsius (C)
3 Checksum (*hh)

22. **MWV** Wind Speed and Angle (*Ángulo y velocidad del viento*)

$--MWV,x.x,a,x.x,a,a*hh<CR><LF>

1 Ángulo del viento, de 0 a 360 grados (x.x)

2 Referencia (a)
R = Relativo
T = Verdadero
3 Velocidad del viento (x.x)
4 Unidades de la velocidad del viento K/M/N (a)
5 Estado (a)
A = Dato válido
6 Checksum (*hh)

19. **OLN** Omega Lane Numbers (*Números de ruta Omega*)

$--OLN,aa,xxx,xxx,aa,xxx,xxx,aa,xxx,xxx*hh<CR><LF>

1 Par 1 Omega (aa,xxx,xxx)
2 Par 1 Omega (aa,xxx,xxx)
3 Par 1 Omega (aa,xxx,xxx)
4 Checksum (*hh)

23. **OSD** Own Ship Data (*Datos propios del barco*)

$--OSD,x.x,A,x.x,a,x.x,a,x.x,x.x,a*hh<CR><LF>

1 Encabezado, grados verdaderos (x.x)
2 Estado (A)
A = Dato válido
3 Trayectoria de la embarcación, grados verdaderos (x.x)
4 Referencia de trayectoria (a)
5 Rapidez de la embarcación (x.x)
6 Referencia de la rapidez (a)
7 Inclinación de la embarcación, en grados verdaderos (x.x)
8 Desviación del rumbo de la embarcación (rapidez) (x.x)
9 Unidades de rapidez (a)
10 Checksum (*hh)

24. **R00** Waypoints in active route (*Puntos de camino en ruta activa*)

$--R00,c---c,c---c,....*hh<CR><LF>

1...n ID de puntos de camino (c- - c)
2 Checksum (*hh)

25. **RMA** Recommended Minimum Navigation Information (*Información mínima recomendada de navegación*)

$--RMA,A,llll.ll,a,yyyyy.yy,a,x.x,x.x,x.x,x.x,x.x,a*hh<CR><LF>

1 Advertencia de destello (A)
2 Latitud (llll.ll)
3 Norte o Sur (a)
4 Longitud (yyyyy.yy)
5 Este u Oeste (a)
6 Diferencia de tiempo A, □S (x.x)
7 Diferencia de tiempo B, □S (x.x)
8 Velocidad en tierra, en nudos (x.x)
9 Trayectoria bien hecha, en grados verdaderos (x.x)
10 Variación magnética, grados (x.x)
11 Este u Oeste (a)
12 Checksum(*hh)

26. **RMB** Recommended Minimum Navigation Information (*Información mínima recomendada de navegación*)

$--RMB,A,x.x,a,c--c,c--c,llll.ll,a,yyyyy.yy,a,x.x,x.x,x.x,A*hh<CR><LF>

1 Estado de los datos
A = Ok
V = Advertencia del receptor de navegación
2 Error a través de la trayectoria en millas náuticas (x.x)
3 Dirección para la guía, Izquierda o derecha (a)
4 ID del punto de camino TO (c - - c)
5 ID del punto de camino FROM (c - - c)
6 Latitud del punto de camino de destino (llll.ll)
7 Norte o Sur (a)
8 Longitud del punto de camino de destino (yyyyy.yy)
9 Este u Oeste (a)
10 Rango al destino en millas náuticas (x.x)
11 Marcación del destino en grados verdaderos (x.x)
12 Velocidad de aproximación al destino en nudos (x.x)

13 Estado de arribo (A)
A = Entrando al circulo de arribo
V = No se ha entrado al circulo de arribo
14 Checksum (*hh)

27. **RMC** Recommended Minimum Navigation Information (*Información mínima recomendada de navegación*)

$--RMC,hhmmss.ss,A,llll.ll,a,yyyyy.yy,a,x.x,x.x,xxxx,x.x,a*hh<CR><LF>

1 Tiempo de la localización (UTC) (hhmmss.ss)
2 Estado
A = Posición válida
V = advertencia del receptor de navegación
3 Latitud (llll.ll)
4 Norte o Sur (a)
5 Longitud (yyyyy.yy)
6 Este u Oeste (a)
7 Velocidad sobre tierra en nudos (x.x)
8 Trayectoria bien hecha en grados verdaderos (x.x)
9 Fecha de la localización, ddmmyy (xxxx)
10 Variación magnética en grados (x.x)
11 Este u Oeste (a)
12 Checksum (*hh)

28. **ROT** Rate Of Turn (*Rata de giro*)

$--ROT,x.x,A*hh<CR><LF>

1 Rata de giro, en grado por minuto, “-“ significa proa gira a babor (x.x)
2 Estado (A)
A = dato válido
3 Checksum (*hh)

29. **RPM** Revolutions (*Revoluciones*)

$--RPM,a,x,x.x,x.x,A*hh<CR><LF>

1 Fuente (a)
S = eje
E = Motor
2 Número de eje o motor (x)
3 Velocidad, revoluciones por minuto (x.x)
4 Paso de la hélice, % de máximo, “-“ significa a popa (x.x)
5 Estado (A)
A = dato válido
6 Checksum (*hh)

30. **RSA** Rudder Sensor Angle (*Ángulo sensor del timón*)

$--RSA,x.x,A,x.x,A*hh<CR><LF>

1 Sensor de timón a estribor o sencillo, “-“ significa girar a babor (x.x)
2 Estado (A)
A = dato válido
3 Sensor de timo de babor (x.x)
4 Estado (A)
A = dato válido
5 Checksum (*hh)

31. **RSD** RADAR System Data (*Datos del sistema RADAR*)

$--RSD,x.x,x.x,x.x,x.x,x.x,x.x,x.x,x.x,x.x,x.x,x.x,a,a*hh<CR><LF>

9 Rango de cursor propio de la nave (x.x)
10 Grados de marcación del cursor horario desde cero (x.x)
11 Escala de rango (x.x)
12 Unidades de rango (a)
14 Checksum (*hh)

32. **RTE** Routes (*Rutas*)

$--RTE,x.x,x.x,a,c--c,c--c, ... c--c*hh<CR><LF>

1 Número total de mensajes que han sido transmitido (x.x)

2 Número de mensaje (x.x)
3 Modo de mensaje(a)
c = ruta completa, todos los puntos de camino
w = Ruta de trabajo, el punto de camino que justo dejó, el punto de camino encabeza a el resto
4 ID de puntos de camino (c - - c)
5 … Más puntos de camino (c - - c)
n Checksum (*hh)

33. **SFI** Scanning Frequency Information (*Información de la frecuencia de escaneo*)

$--SFI,x.x,x.x,xxxxxx,c ..... xxxxxx,c*hh<CR><LF>

1 Número total de mensajes (x.x)
2 Número de mensaje (x.x)
3 Frecuencia 1 (xxxxxx)
4 Modo 1 (c)
n Checksum (*hh)

34. **STN** Multiple Data ID (*ID de datos múltiples*)

$--STN,x.x,*hh<CR><LF>

1 Número ID del emisor (x.x)
2 Checksum (*hh)

20. **TLL** Target Latitude and Longitude (*Latitud y longitud del objetivo*)

Formato desconocido

35. **TRF** TRANSIT Fix Data (*Datos de localización de tránsito*)

$--TRF,hhmmss.ss,xxxxxx,llll.ll,a,yyyyy.yy,a,x.x,x.x,x.x,x.x,xxx,A*hh<CR><LF>

1 Tiempo (UTC) (hhmmss.ss)

2 Fecha, ddmmyy (xxxxxx)
3 Latitud (llll.ll)
4 Norte o Sur (a)
5 Longitud (yyyyy.yy)
6 Este u Oeste (a)
7 Ángulo de elevación (x.x)
8 Número de iteraciones (x.x)
9 Número de intervalos de Doppler (x.x)
10 Distancia actualizada en millas náuticas (x.x)
11 ID del satélite (xxx)
12 Dato válido (A)
13 Checksum (*hh)

36. **TTM** Tracked Target Message (*Mensaje del objetivo recorrido*)

$--TTM,xx,x.x,x.x,a,x.x,x.x,a,x.x,x.x,a,c--c,a,a*hh<CR><LF>

1 Número del objetivo (xx)
2 Distancia del objetivo (x.x)
3 Marcación desde el barco (x.x)
4 Unidades de marcación (a)
5 Velocidad del objetivo (x.x)
6 Curso del objetivo (x.x)
7 Unidades del curso (a)
8 Distancia del punto de aproximación más cercano (x.x)
9 Tiempo antes del punto de aproximación más cercano "-" significa aumento (x.x)
11 Nombre del objetivo (c - - c)
12 Estado del objetivo (a)
13 Objetivo de referencia (a)
14 Checksum (*hh)

37. **VBW** Dual Ground/Water Speed (*Velocidad dual tierra/agua*)

$--VBW,x.x,x.x,A,x.x,x.x,A*hh<CR><LF>

1 Velocidad longitudinal en agua, "-" significa a proa (x.x)
2 Velocidad transversal en agua, "-" significa a babor (x.x)
3 Estado (A)
A = Dato válido
4 Velocidad longitudinal en tierra, "-" significa a proa (x.x)
5 Velocidad transversal en tierra, "-" significa a babor (x.x)

6 Estado (A)
A = Dato válido
7 Checksum (*hh)

38. **VDR** Set and Drift (*Posición y desvio*)

$--VDR,x.x,T,x.x,M,x.x,N*hh<CR><LF>

1 Grados verdaderos (x.x)
2 T = Verdadero (T)
3 Grados magnéticos (x.x)
4 M = Magnético (M)
5 Nudos (velocidad de corriente) (x.x)
6 N = Nudos (N)
7 Checksum (*hh)

39. **VHW** Water speed and heading (*Encabezado y velocidad del agua*)

$--VHW,x.x,T,x.x,M,x.x,N,x.x,K*hh<CR><LF>

1 Grados verdaderos (x.x)
2 T = Verdadero (T)
3 Grados mágneticos (x.x)
4 M = Mágnetico (M)
5 Nudos (velocidad de la embarcación relativa al agua) (x.x)
6 N = Nudos (N)
7 Kilómetros (velocidad de la embarcación relativa al agua) (x.x)
8 K = Kilómetros (K)
9 Checksum (*hh)

21. **VLW** Distance Traveled through Water (*Distancia viajada a través del agua*)

$--VLW,x.x,N,x.x,N*hh<CR><LF>

1 Distancia total acumulada (x.x)
2 N = Millas náuticas (N)

3 Distancia desde el reinicio (x.x)
4 N = Millas náuticas (N)
5 Checksum (*hh)

22. **VPW** Speed - Measured Parallel to Wind (*Velocidad medida paralelamente al viento*)

$--VPW,x.x,N,x.x,M*hh<CR><LF>

1 Velocidad, "-" significa a favor del viento (x.x)
2 N = Nudos (N)
3 Velocidad, "-" significa a favor del viento (x.x)
4 M = Metros por segundo (M)
5 Checksum (*hh)

23. **VTG** Track made good and Ground speed (*Trayectoria bien hecha y velocidad en tierra*)

$--VTG,x.x,T,x.x,M,x.x,N,x.x,K*hh<CR><LF>

1 Grados de trayectoria (x.x)
2 T = Verdadero (T)
3 Grados de trayectoria (x.x)
4 M = Mágnetico (M)
5 Velocidad en nudos (x.x)
6 N = Nudos (N)
7 Velocidad en Kilómetros por hora (x.x)
8 K = Kilómetros por hora (K)
9 Checksum (*hh)

24. **VWR** Relative Wind Speed and Angle (*Ángulo y velocidad relativos del viento*)

$--VWR,x.x,a,x.x,N,x.x,M,x.x,K*hh<CR><LF>

1 Magnitud de la dirección del viento en grados (x.x)
2 Dirección del viento, izquierda/derecha de proa (a)
3 Velocidad (x.x)
4 N = Nudos (N)

5 Velocidad (x.x)
6 M = Metros por segundo (M)
7 Velocidad (x.x)
8 K = Kilómetros por hora (K)
9 Checksum (*hh)

25. **WCV** Waypoint Closure Velocity (*Velocidad de fin de punto de camino*)

```
$--WCV,x.x,N,c--c*hh<CR><LF>
```

1 Velocidad (x.x)
2 N = Nudos (N)
3 ID del punto de camino (c - - c)
4 Checksum (*hh)

26. **WNC** Distance - Waypoint to Waypoint (*Distancia entre puntos de camino*)

```
$--WNC,x.x,N,x.x,K,c--c,c--c*hh<CR><LF>
```

1 Distancia en millas náuticas (x.x)
2 N = Millas náuticas (N)
3 Distancia en kilómetros (x.x)
4 K = Kilómetros (K)
5 Punto de camino TO (c - - c)
6 Punto de camino FROM (c - - c)
7 Checksum (*hh)

27. **WDC** Distance to Waypoint – Great Circle (*Distancia al punto de camino – Gran circulo*)

Formato desconocido

28. **WDR** Distance to Waypoint – Rhumb Line (*Distancia al punto de camino – Línea de rumbo*)

Formato desconocido

29. **WPL** Waypoint Location (*Localización del punto de camino*)

$--WPL,llll.ll,a,yyyyy.yy,a,c--c*hh<CR><LF>

1 Latitud (llll.ll)
2 Norte o Sur (a)
3 Longitud (yyyyy.yy)
4 Este u Oeste (a)
5 Nombre del punto de camino (c - - c)
6 Checksum (*hh)

30. **XDR** Cross Track Error - Dead Reckoning (*Error de travesía – Cálculo muerto*)

$--XDR,a,x.x,a,c--c, ... *hh<CR><LF>

1 Tipo de transductor (a)
2 Dato de medida (x.x)
3 Unidades de medida (a)
4 - x Nombre de los transductores (c - - c)
n Checksum (*hh)

31. **XTE** Cross-Track Error, Measured (*Error de travesía, medido*)

$--XTE,A,A,x.x,a,N,*hh<CR><LF>

1 Estado (A)
V = destello del LORAN-C o advertencia SNR
V = Bandera de advertencia general u otros sistemas de navegación cuando la posición exacta no esta disponible
2 Estado (A)
V = Bandera de advertencia de bloqueo de ciclo de Loran-C
A = OK o no es usado
3 Magnitud del error de travesía (x.x)
4 Dirección de guía (L o R) (Derecha o Izquierda) (a)

5 Unidades de travesía, N = Millas náuticas (N)
6 Checksum (*hh)

32. **XTR** Cross Track Error - Dead Reckoning (Error de travesía – Cálculo muerto)

$--XTR,x.x,a,N*hh<CR><LF>

1 Magnitud del error de travesía (x.x)
2 Dirección de guía L o R (Izquierda o derecha) (a)
3 Unidades, N = Millas náuticas (N)
4 Checksum (*hh)

33. **ZDA** Time & Date - UTC, day, month, year and local time zone (*Tiempo y fecha – UTC, día, mes, año y tiempo de zona local*)

$--ZDA,hhmmss.ss,xx,xx,xxxx,xx,xx*hh<CR><LF>

1 Tiempo universal coordinado (UTC) (hhmmss.ss)
2 Día, 01 al 31 (xx)
3 Mes, 01 al 12 (xx)
4 Año (xxxx)
5 Descripción de la zona horaria local en horas, 00 a ± 13 horas (xx)
6 Descripción de la zona horaria local en minutos, mismo signo de las horas locales (xx)
7 Checksum (*hh)

34. **ZDL** Time and Distance to Variable Point (*Tiempo y distancia para punto variable*)

Formato desconocido

35. **ZFO** UTC & Time from origin Waypoint (*UTC y tiempo desde un punto de camino de origen*)

$--ZFO,hhmmss.ss,hhmmss.ss,c--c*hh<CR><LF>

1 Tiempo universal coordinado (UTC) (hhmmss.ss)
2 Tiempo transcurrido (hhmmss.ss)
3 ID del punto de camino de origen (c - - c)
4 Checksume (*hh)

36. **ZTG** UTC & Time to Destination Waypoint (*UTC y tiempo al punto de camino de destino*)

```
$--ZTG,hhmmss.ss,hhmmss.ss,c--c*hh<CR><LF>
```

1 Tiempo universal coordinado (UTC) (hhmmss.ss)
2 Tiempo sobrante (hhmmss.ss)
3 ID del punto de camino de destino (c - - c)
4 Checksume (*hh)

7. Formato de algunas sentencias propietarias

37. **PGRMB** DGPS Beacon Information (*Información del DGPS guía*)

```
$PGRMB,xxx.x,x,x,xx,xx,K,a,a,a*hh<CR><LF>
```

1 Frecuencia sintonizada (283.3 – 322.0 en 0.5 pasos) (xxx.x)
2 Rata de bits, bits por segundo (0,25,50,100,200) (x)
3 SNR, 0 – 31 (x)
4 Calidad de los datos 0 - 100 (xx)
5 Distancia a la estación referencia de guía (xx)
6 Unidad de distancia (K = Kilómetros) (K)
7 Estado de comunicación del receptor (a)
  0 = Chequeo de instalación eléctrica
  1 = Sin señal
  2 = Sintonía
  3 = Recepción
  4 = Escaneo
8 Fuente de localización (a)
  R = RTCM
  W = WAAS,
  N = No hay localización de DPGS

9 Modo DGPS (a)
A = Automático
W = WAAS solamente
R= RTCM solamente
N = Ninguno, DGPS deshabilitado
10 Checksum (*hh)

38. **$PGRMC** Sensor Configuration Information (*Información de la configuración del sensor*)

$PGRMC,A,x.x,hh,x.x,x.x,x.x,x.x,x.x,x.x,c,c,c,c,x,ss*hh<CR><LF>

1 Modo de localización (A)
A = Autómatico
2 = 2D exclusivamente, el sistema patrón debe suministrar la altitud
3 = 3D exclusivamente
2 Altitud por encima/debajo del nivel del mar promedio en metros (x.x)
3 Índice del datum de la tierra. Si el índice del datum del usuario está especificado (96), los campos 4 al 8 deben contener valores válidos, de lo contrario deben estar en blanco (hh)
4 Eje semi – mayor en metros con resolución de 0.001 metros (x.x)
5 Factor de aplanamiento inverso, de 285 a 310, resolución de 10e-9 (x.x)
6 Coordenada centrada Delta X de la tierra en metros, -5000 a 5000 con un metro de resolución (x.x)
7 Coordenada centrada Delta Y de la tierra en metros, -5000 a 5000 con un metro de resolución (x.x)
8 Coordenada centrada Delta Z de la tierra en metros, -5000 a 5000 con un metro de resolución (x.x)
9 Modo diferencial (c)
A = Automático, localizaciones de salidas de DGPS cuando están disponibles de otro modo no se encuentran DGPS
D = Solamente localizaciones de salidas diferenciales
10 Rata de baudios de NMEA 0183 (c)
1 = 1200
2 = 2400
3 = 4800
4 = 9600
5 = 19200
6 = 300
7 = 600

11 Filtro de velocidad en segundos (c)
0 = Ninguno
1 = Automático
2 – 225 = Filtro constante
12 Modo PPS (c)
1 = Ninguno
2 = 1 Hertz
13 Longitud del pulso PPS (x)
N = 0 a 48
Longitud (milisegundos) = (N + 1)*20
14 Tiempo de validación del cálculo muerto, 1 a 30 segundos (ss)
15 Checksum (*hh)

39. **$PGRMCE** Sensor Configuration Information Enquiry (*Solicitud de Información de configuración del sensor*)

$PGRMCE*hh<CR><LF>

La unidad responderá transmitiendo una sentencia $PGRMC que contenga los valores por defecto actuales.

1 Checksum (*hh)

40. **$PGRMC1** Additional Sensor Configuration Information (*Información adicional de la configuración del sensor*)

$PGRMC1*hh<CR><LF>

1 Tiempo de salida NMEA0183 en segundos, 1 – 900 (No aplicable al GPS16A) (xxx)
2 Datos de salida en binario (c)
1 = off
2 = on
3 Definiendo posición (c)
1 = off
2 = on
4 Frecuencia del guía DGPS en KiloHertz, 283.5 a 325.0 en 0.5 pasos (xx)
5 Rata de bits del guía DGPS (0,25,50,100,200) (x)

6 Escaneo de guía DGPS (a)
1 = off
2 = on
7 Indicador de modo NMEA0183 versión 3 (a)
1 = off
2 = on
8 Modo DGPS (a)
A = Automático
W = WAAS solamente
R= RTCM solamente
N = Ninguno, DGPS deshabilitado
9 Modo de ahorro de potencia (c)
P = Activo
N = Normal
10 Checksum (*hh)

41. **$PGRMC1E** Additional Sensor Configuration Information Enquiry (*Solicitud de la información adicional de la configuración del sensor*)

$PGRMC1E*hh<CR><LF>

La unidad responderá transmitiendo una sentencia $PGRMC1 que contenga los valores por defecto actuales.

1 Checksum (*hh)

42. **$PGRME** Estimated Position Error (*Error de posición estimada*)

$PGRME,x.x,M,x.x,M,x.x,M*hh<CR><LF>

1 Error horizontal estimado (HPE) (x.x)
2 Unidades, metros (M)
3 Error vertical estimado (VPE) (x.x)
4 Unidades, metros (M)
5 Error de posición equivalente esférica global (x.x)
6 Unidades, metros (M)
7 Checksum (*hh)

43. $PGRMF Position Fix Sentence (Sentencia de localización de posición)

```
$PGRMF,x.x,x.x,ddmmyy,hhmmss,x.x,ddmm.mmmm,c,dddmm.mmmm,c,c,c,x.x,x.x,c,c*hh<CR><LF>
```

1 Número de semana GPS (0 – 1023) (x.x)
2 Segundos GPS (0 – 604799) (x.x)
3 Localización de posición de fecha UTC (ddmmyy)
4 Localización de posición del tiempo UTC (hhmmss.ss)
5 Cuenta de segundos saltados GPS (x.x)
6 Latitud (ddmm.mmmm)
7 Norte o Sur (c)
8 Longitud (ddmm.mmmm)
9 Este u Oeste (c)
10 Modo (c)
M = Manual
A = Automático
11 Tipo de Localización (c)
0 = Sin localización
1 = Localización en 2D
2 = Localización en 3D
12 Velocidad sobre tierra, 0 – 999 kilómetros por hora (x.x)
13 Trayectoria sobre tierra, 0 – 359 grados, verdadero (x.x)
14 Dilución del posición de precisión 0 – 9 (redondeado al entero más cercano) (c)
15 Dilución del tiempo de precisión 0 – 9 (redondeado al entero más cercano) (c)
16 Checksum (*hh)

44. **$PGRMI** Sensor Initialization Information (*Información de inicialización del sensor*)

```
$PGRMI, ddmm.mmmm,N,ddmm.mm,E,ddmmyy,hhmmss*hh<CR><LF>
```

1 Latitud (ddmm.mmm)
2 Norte o sur (N)
3 Longitud (ddmm.mmm)
4 Este u Oeste (E)
5 Fecha actual UTC (ddmmyy)
6 Tiempo actual UTC (hhmmss)
7 Checksum (*hh)

45. **$PGRMIE** Sensor Initialization Information Enquiry (*Solicitud de información de inicialización del sensor*)

$PGRMIE*hh<CR><LF>

La unidad responderá transmitiendo una sentencia $PGRMI que contenga los valores actuales por defecto.

1 Checksum (*hh)

46. **$PGRMM** Map Datum (*Datum*)

$PGRMM, c - - c*hh<CR><LF>

1 Datum horizontal activo actualmente (WGS – 84, NAD27 Canadá, ED50, a.s.o) (c - - c)
2 Checksum (*hh)

47. **$PGRMO** Output Sentence Enable/Disable (*Habilita/deshabilita sentencia de salida*)

$PGRMO,ccccc,c*hh<CR><LF>

1 Descripción de sentencia objetivo, por ejemplo PGRMT; GPGSV, etc.. (ccccc)
2 Modo de sentencia objetivo (c)
0 = Deshabilita sentencia específica
1 = Habilita sentencia específica
2 = Deshabilita toda
3 = Habilita todas las sentencias de salida (excepto GPALM)
3 Checksum (*hh)

48. **$PGRMT** Sensor Status Information (*Información del estado del sensor*)

$PGRMT,c…c,c,c,c,c,c,cx.x,c*hh<CR><LF>

1 Producto, modelo y versión del software por ejemplo “GPS23VEE1” (c…c)
2 Prueba del checksum Rom (c)
P = Pasa
F = Fallo
3 Falla discreta del receptor (c)
P = Pasa
F = Fallo
4 Pérdida de datos almacenados (c)
R = Retenido
L = Perdido
5 Pérdida del tiempo real del reloj (c)
R = Retenido
L = Perdido
6 Desviación discreta del oscilador (c)
P = Pasa
F = Fallo
7 Recolección discreta de datos (c)
C = Recolectando
8 Temperatura de tablero en grados centígrados (x.x)
9 Datos de configuración del tablero (c)
R = Retenido
L = Perdido
10 Checksum (*hh)

49. **$PGRMV** 3D Velocity (*Velocidad 3D*)

$PGRMV,x.x,x.x,x.x*hh<CR><LF>

1 Velocidad de Este verdadero, -999.9 a 9999.9 metros/segundo (x.x)
2 Velocidad de Norte verdadero, -999.9 a 9999.9 metros/segundo (x.x)
3 Velocidad de ascenso, -999.9 a 9999.9 metros/segundo (x.x)
4 Checksum (*hh)

50. **$PGRMZ** Altitude Information (*Información de altitud*)

$PGRMZ,x.x,f,h*hh<CR><LF>

1 Altitud (x.x)
2 Unidades en pies (f)
3 Dimensiones de localización de posición (h)
2 = Altitud del usuario
3 = Altitud del GPS
4 Checksum (*hh)

40. **$PSLIB** Differential GPS Beacon Receiver Control (*Control del receptor guía GPS diferencial*)

SPSLIB,x.x,x.x,c*hh<CR><LF>

1 Frecuencia (x.x)
2 Rata de bits (x.x)
3 Tipo de petición (c)
J = Petición de estado
K = Petición de configuración
4 Checksum (*hh)

Anexo B. DOCUMENTACIÓN DE LA APLICACIÓN NMEAParserDemo

La aplicación NMEAParserDemo.exe se construye con base en 17 archivos entre archivos fuentes, de encabezamiento, de recursos y de dependencias externas. Los más significativos serán explicados a continuación:

1. <u>CommSettingDlg.h: Archivo de encabezamiento</u>

Class CCommSettingDlg : public Cdialog { public: int m_nBaud; int m_nPort;	Se declaran dos variables de tipo entero para capturar los valores de puerto y tasa de transferencia.
*CCommSettingsDlg(CWnd *pParent = NULL);*	Constructor estándar.

Este archivo tiene como finalidad determinar las pautas necesarias para que se cree un cuadro de dialogo en donde se pueda configurar las dos variables anteriormente declaradas.
El asistente de clases del Visual C++ genera funciones, miembros, variables, declaraciones de los mismos e incluso declaraciones adicionales que sean necesarias para el adecuado funcionamiento de la aplicación.

_-
_-_-_-

2. <u>CommSettingsDlg.cpp: Archivo de implementación</u>

*CCommSettingsDlg::CCommSettingsDlg(CWnd *pParent /*=NULL*/):CDialog(CCommSettingsDlg::IDD, pParent)
{
m_nBaud=4800;
m_nPort=1;
}*
	Se llama la función responsable del manejo de la ventana de dialogo y se asignan valores por defecto a las variables m_nBaud y m_nPort.

void CCommsettingsDlg::DoDataExchange(CdataExchange pDX)*	Es llamado por Visual C++ para hacer intercambio y validación de los datos que

```
{
Cdialog::DoDataExchange(pDX);
}
```

se tienen en las ventanas de diálogos.

```
void CCommSettingsDlg::OnOk()
{
m_nBaud = GetItemInt(IDC_BAUD);
m__nPort = GetItemInt(IDC_PORT);
Cdialog ::OnOk() ;
}
```

Se llama la función OnOk cuando se hace clic en el botón OK del cuadro de dialogo y los valores que se obtienen de los static text son asignados a las dos variables en forma de enteros.

```
BOOL CCommSettingsDlg::OnInitDialog()
{
CDialog::OnInitDialog();
SetDlgItemInt(IDC_BAUD, m_nBaud);
SetDlgItemInt(IDC_PORT, m_nPort);
return TRUE ;
}
```

Por último se encuentra la función que nos indica si se ha colocado el foco en uno de los controles de la caja de dialogo y se realiza conversión del texto de los controles en una representación de cadena de números enteros.

_-
_-_-_-

3. <u>NMEAParserDemo.h: Archivo de encabezamiento principal para la aplicación NMEAPARSERDEMO</u>

```
class CNMEAParserDemoApp : public CWinApp
{
public:
```

Se crea la clase CNMEAParserDemoApp, que es derivada de la clase CWinApp, que nos permite crear aplicaciones en Windows.

```
CNMEAParserDemoApp();
```

Se enuncia la función constructora de la aplicación.

```
virtual BOOL InitInstance();
```

Se nombra la función para la inicialización

```
DECLARE_MESSAGE_MAP()
}
```

Forma parte de la implementación que el asistente de clases de Visual C++ utiliza cuando se realiza una aplicación en ambientes Windows.

_-
_-_-_-

4. NMEAParserDemo.cpp. Define los comportamientos de la clase para la aplicación

BEGIN_MESSAGE_MAP(CNMEAParserDemoApp, CWinApp)	Es un macro que se usa para iniciar el mapa de mensaje que atañe al hecho de usar una clase de tipo aplicación windows.
CNMEAParserDemoApp::CNMEAParserDemoApp() *{* *}*	Se crea la clase CNMEAParserDemoApp que tiene un único objeto derivado de CWinApp.
BOOL CNMEAParserDemoApp::InitInstance()	Inicialización estándar de la clase.

El resto de código hace referencia al hecho de trabajar con ventanas de dialogo.

---

5. sio.h: Archivo de encabezamiento

class CSio *{* *int m_bErrorFlag;*	Esta clase cuenta con una bandera de error que es una variable entera.
	Se declaran las siguientes variables públicas,
HANDLE m_hFile;	: Manejador del puerto
CString m_sFileName;	: Guarda cadenas de caracteres del puerto
int m_nBaud;	: Tasa de transferencia de datos
int m_nport;	: Número del puerto
CSio(); *~CSio();*	Se define el constructor y destructor

Se declaran las funciones para las operaciones usadas en el manejo del puerto de comunicaciones,

IsConnectError();	: Dice si existe conexión con el puerto
InitComm();	: Inicializa el puerto
Close();	: Cierra el puerto

Open_file()	: Abre un archivo para guardar las sentencias o información recibidas por el puerto
DWORDWrite();	: Se escribe por el buffer de datos
DWORDRead();	: Lee por el buffer de datos

_-

6. sio.cpp : Archivo de implementación

CSio::Csio() *{* *m_hFile=INVALID_HANDLE_VALUE;* *m_sFile=_T("Untitled.txt");* *}*	En el constructor de la clase CSio se asignan valores iniciales a las variables m_hFile y m_sFilename.
CSio::~CSio() *{* *closeHandle(m_hFile)* *}*	Se crea el destructor; es una función especial que en este caso elimina el objeto m_hFie, para liberar la memoria de dicho objeto.
BOOL CSio::InitComm(int sport, int nBaud, int nParity, int nData, int nStop, BOOL bNoError)	Se determina la función de inicialización de puerto

Esta función es de tipo booleano, y devuelve un valor verdadero si se dio inicio a la comunicación. Esta función recibe los siguientes parámetros:

nPort	:Número de puerto (1 – 256)
nBaud	: Bits por segundo, tasa de transferencia (300 – 115200)
nParity	: Almacena la paridad que puede ser: NOPARITY → Paridad ninguna ODDPARITY → Paridad impar EVENPARITY → Paridad par MARKPARITY → Marca SPACEPARITY → Espacio
nData	: Bits de datos que puede ser 7 u 8
nStop	: Bits de parada que pueden ser: ONESTOPBIT = 1 ONESSTOPBITS = 1.5 TWOSTOPBITS = 2
BOOL bNoError	: Para mostrar mensajes de error
char S[100];	: Declaramos un arreglo o una cadena de caracteres

int err;	: Error
DCB dcb;	: Declaramos un objeto (dcb) de la clase DCB (Device – Control Block); esta estructura define la configuración de control para un dispositivo con comunicación serial. Llenando esta estructura con los valores requeridos, se pude cambiar los parámetros de conexión o aquellos necesitados en ese momento.
m_nBaud = nBaud; *m_nPort = nPort;*	A nuestras variables, le asignamos los parámetros pasados por referencia.
m_bErrorFlag = False;	Error de advertencia o bandera que se inicializa inactivo.
Sprintf(s, "COM%d", nport);	Función que permite guardar o escribir en un archivo las sentencias en este formato.

La función CreateFile crea o abre un archivo, directorio, volumen, buffer de consola, etc. Esta función retorna un manejador que puede ser usado para el acceso al objeto

m_hFile= CreateFile(

s,	Apuntador al nombre del archivo
GENERIC_READ|GENERIC_WRITE,	Este comando nos permite leer o escribir sobre el dispositivo.
0,	Esto permite que el objeto accese el dispositivo.
NULL,	Apuntador a los atributos de seguridad que impiden que este archivo sea heredado.
OPEN_EXISTING	Abrir archivo y si este no existe se genera error.
FILE_ATRIBUTE_NORMAL	Atributos normales de un archivo, existe también:
NULL,	Manejador de archivos con atributos de copia.

if (m_hFile == INVALID_HANDLE_VALUE)
{

Si el valor del manejador es invalido, se presentan los siguientes errores:

m_bErrorFlag = TRUE;	La bandera de error se activa
DWORD dwLastError = GetLastError;	Se crea una variable tipo DWORD donde se almacena el tipo de error que retorna de la función GetLastError
CString sError;	Se crea una cadena de

	caracteres para almacenar el mensaje de falla.
switch(dwLastError) *{*	Compara dwLastError con los casos de error ya definidos y selecciona cual es el caso que se presenta.
ERROR_ACCESS_DENIED *ERROR_FILE_NOT_FOUND*	Dependiendo del caso, la cadena de caracteres sError toma el valor de diferentes mensajes predefinidos.
ltoa (dwLastError,s,10)	Si no se cumple ninguno de los anteriores casos se muestra el error presente en dwLastError usando la función ltoa que convierte un entero largo en una cadena de caracteres. dwLastError → Es el valor para ser representado en una cadena. s → Buffer donde se almacena la cadena resultante. 10 → Base en el cual el valor va a ser presentado
If (bNoError == FALSE) *AfxMenssageBox(sError,MB_ICONSTOP|MB_OK);*	Esto es la condición de una doble negación, que consulta si es falso que no haya error, en otras palabras, ¿Hay error? Entonces despliega el mensaje guardado en sError.

El estado del puerto COM se obtiene con los datos en m_hFile y la dirección del objeto dcb de la función GetCommState. Estos datos son almacenados en la variable err

err = GetCommState(m_hFile,&dcb);

if (err<0)
{

Si se presenta que err<0 se generan las siguientes asignaciones:

m_bErrorflag=TRUE	Toma valor verdadero
bNoError==FALSE	Se despliega una advertencia sonora y una ventana de mensaje de error.

Se realiza asignaciones a los campos del objeto dcb con los siguientes parámetros:

dcb.BaudRate (DWORD)nBaud;	=	Tasa de comunicación del dispositivo.
dcb.Parity = (BYTE)nParity;		Paridad del puerto.
dcb.StopBits = (BYTE)nStop;		Bits de parada.
dcb.ByteSize = (BYTE)nData;		Bits de datos.
dcb.fbinary = TRUE;		Especifica si el modo binario esta disponible, valor por defecto.
dcb.fOutxCtsFlow = FALSE;		Especifica que el control de flujo en la salida es CTS (Clear to Send).
dcb.fOutxDsrFlow = FALSE;		Control de flujo de salida DSR
dcb.fDtrControl = DTR_CONTROL_ENABLE;		Control de flujo DTR (Data – Terminal - Ready)
dcb.fDsrSensitivy = FALSE;		El driver de comunicación es sensible al estado de señal DSR
dcb.fOutX = FALSE;		Especifica si el control de flujo Xon/Xoff es usado durante la transmisión
dcb.fInX = FALSE;		Especifica si el control de flujo Xon/Xoff es usado durante la recepción
dcb.fNull = FALSE;		Especifica si los bytes nulos son descartados.
dcb.fRtsControl = RTS_CONTROL_ENABLE;		Especifica el control de flujo RTS (Request to Send).
Dcb.fAbortOnError = FALSE		Especifica si las operaciones de lectura y escritura son canceladas si un error ocurre.

La nueva configuración del puerto se realiza con la función SetCommState, que es de tipo booleano, si entrega cero, el puerto no esta activo y se debe generan errores:

err = SetComnState(m_hFile,&dcb);

Si la variable err devuelve un valor cero ocurre lo siguiente,

m_bErrorFlag= TRUE;	La bandera de error se activa
bNoError == FALSE	Se muestra una alarma sonora y un mensaje de error : “Could not set communication state”

La función CommTimeOuts es la encargada de las interrupciones cuando se manipula comunicaciones seriales.
Se llama el objeto CommTimeOuts de la clase CommTimeOuts. Este objeto tiene los siguientes campos que son configurados de la siguiente manera:

CommTimeOuts.ReadIntervalTimeout = MAXDWORD;	Se especifica el máximo número de milisegundos que pueden transcurrir entre dos caracteres sin que ocurra una interrupción.
CommTimeOuts.ReadTotalTimeoutMultiplier = 0;	Para cada operación de lectura, este número es multiplicado por el número de bytes que la operación relectura espera recibir.
CommTimeOuts.ReadTotalTimeoutsConstant = 0;	Es el número de milisegundos agregados al resultado de multiplicar el número total de bytes a leer por el campo de multiplicador.
CommTimeOuts.WriteTotalTimeoutMultiplier = 0;	Para cada operación de escritura, este número es multiplicado por el número de bytes que la operación relectura espera recibir.
CommTimeOuts.WriteTotalTimeoutConstant = 1000;	Para cada operación de escritura, este número es multiplicado por el número de bytes que la operación de escritura espera recibir.

SetCommTimeouts(m_hFile,&CommTimeOuts);

Se ejecuta la función SetCommTimeouts (encargada de activar las interrupciones) y se le pasa el archivo manejador del puerto m_hFile y la dirección donde se encuentra almacenado el objeto.

ClearCommError (m_hFile,&dwErrors,&ComStat);

Esta función recupera la información acerca de errores de comunicación y reporta el estado actual del dispositivo de comunicación. Esta función es llamada cuando un error de comunicación ocurre y limpia la bandera de error del dispositivo para habilitarlo para operaciones adicionales de entrada y salida.

SetupComm(m_hFile, 16384, 16384)

Esta función inicializa los parámetros de comunicación para un dispositivo específico.

m_hFile	Manejador del dispositivo de comunicación.

16384	Especifica el tamaño recomendado en bytes del buffer interno de entrada.
16384	Especifica el tamaño recomendado en bytes del buffer interno de salida.

Se determina la función de cierre del puerto que es de tipo booleano.

BOOL CSio::Close()

Se realizan asignaciones necesarias,

BOOL bVal = CloseHandle (m_hFile);		Si esta variable esta activa, el manejador del puerto se cerró exitosamente.
m_hFile	=	Se asigna este valor al manejador.
INVALID_HANDLE_VALUE,		
sleep(1000)		Este comando se usa para detener en un tiempo determinado la ejecución de la siguiente línea o salida de la función en que se encuentra.

void CSio::OpenFile(CString sFileName)

Se llama la función OpenFile que abre un archivo, para esto, la función recibe una cadena de caracteres con el nombre del mismo. Devuelve un manejador que puede ser usado para el acceso al objeto.

Los campos de la función OpenFile son configurados de la siguiente manera,

sFileName;	Puntero al nombre del archivo.
GENERIC_READ,	Solo es permitida la operación de lectura.
0,	Este valor indica que el objeto puede acceder al dispositivo.
NULL,	Apuntador a los atributos de seguridad que impiden que este archivo sea heredado.
OPEN_EXISTING,	Abrir archivo, si este no existe se genera error.
FILE_ATRIBUTE_READONLY,	Atributo del archivo para que sea solo de lectura.
NULL	Manejador del archivo con atributos de copia.

Si m_hFile presenta un valor de manejo inválido se despliega un mensaje de error de este tipo: “CSio:File Error”.

La función Read de tipo DWORD es solicitada en la clase CSio con dos parámetros, un apuntador *pData de tipo byte y un archivo dwLen de tipo DWORD, además un archivo dwBytesRead de tipo DWORD.

Esta función lee los datos de un archivo comenzando en la posición indicada por el puntero, después de terminar la lectura el apuntador es ajustado por el número de bytes actualmente leídos.

Si no se presenta algún valor inválido en el manejador, se especifica:

m_hFile,	Manejador del archivo a leer.
pData,	Dirección del buffer que recibe los datos.
dwLen,	Número de bytes a leer.
&dwBytesRead,	Dirección del número de bytes leídos.
NULL	Dirección de la estructura de datos.

DWORD dwLastError = GetLastError();

Se revisa el estado de errores ocurridos y se almacenan en dwLastError.

ClearCommError(m_hFile,$dwErrors,&ComStat);

Esta función recupera la información acerca de errores de comunicación y reporta el estado actual del dispositivo de comunicación. Esta función es llamada cuando un error de comunicación ocurre y limpia la bandera de error del dispositivo para habilitarlo para operaciones adicionales de entrada y salida.

Al final del proceso de lectura se tiene el archivo dwBytesRead con los bytes que han sido leídos.

Se llama la función de escritura de la clase CSio, esta función transfiere los datos a un dispositivo o archivo. Comienza escribiendo datos en la posición indicada por el apuntador del archivo, al final este apuntador se ubica en el número de bytes actualmente escritos.

Se crea dwBytesWrittern, donde se encuentran los datos o información escrita. Se devuelve un valor booleano verdadero si todos los campos de la estructura son válidos,

m_hFile,	Manejador del archivo a ser escrito.
pBuff,	Puntero de los datos a escribir en el archivo.
dwLen,	Número de bytes a escribir.
&dwBytesWrittern,	Apuntador al número de bytes escritos.
NULL	Dirección de la estructura necesitada para

sobrecarga de entradas y salidas.

Al final de la operación de escritura se tiene el archivo dwBytesWrittern con los datos que serán usados por el analizador de sentencias NMEA.

_-

7. <u>NMEAParser.h: Interfase para la clase CNMEAParser</u>

Nota: Sentencia o mensajes NMEA analizadas:
GPGGA, GPGSA, GPGSV, GPRMB, GPRMC, GPZDA

enum NP_STATE{	Es una estructura que nos permite etiquetar diferentes campos de la sentencia.
NP_STATE_SOM = 0,	Busca el campo 0 en el comienzo de la sentencia.
NP_STATE_CMD,	Trae el comando (GGA, GSA…)
NP_STATE_DATA,	Trae los datos.
NP_STATE_CHECKSUM_1	Trae el valor del primer carácter del checksum.
NP_STATE_CHECKSUM_2	Trae el valor del segundo carácter del cheksum.
}	

#define NP_MAX_CMD_LEN	8	Longitud máxima delos comandos.
#define NP_MAX_DATA_LEN	256	Longitud máxima de los datos.
#define NP_MAX_CHAN	36	Número máximo de canales
#define NP_WAYPOINT_ID_LEN	32	Longitud máxima de la cadena de puntos de camino (waypoint)

Class CNPSatInfo
{
public:

Esta clase maneja la información de todos los satélites vistos por el GPS. Se declaran las siguientes variables.

WORD m_wPRN;	Identificador único de cada satélite.
WORD m_wSignalQuality;	Calidad de la señal del satélite.
BOOL m_bUsedInSolution;	Variable booleana que se activa si es usada por el GPS, para determinar la solución de la posición.
WORD m_wAzimut;	Ángulo en el plano horizontal del satélite con respecto al norte.
WORD m_wElevation;	Ángulo en el plano vertical del satélite con respecto al horizonte.
}	

Class CNMEAParser
{

En esta clase se declaran las variables globales y específicas, y las funciones necesarias para el análisis de las sentencias, como sigue:

NP_STATE m_nState;	Estado actual del analizador del protocolo esta activo.
BYTE m_btChecksum;	Checksum calculado de la sentencia NMEA.
BYTE m_btReceivedChecksum;	Checksum recibido de la sentencia NMEA (si existe).
WORD m_wIndex;	Índice usado para comando y datos.
BYTE m_pCommand[NP_MAX_CMD_LEN];	Comando NMEA. (Indicamos su longitud máxima)
BYTE m_pData[NP_MAX_DATA_LEN];	Dato NMEA. (Indicamos su longitud máxima.
DWORD m_dwCommandCount;	Número de los comandos NMEA recibidos (procesadas o no).

Variables utilizadas para los datos GPGGA

BYTE m_btGGAHour;	Hora.
BYTE m_btGGAMinute;	Minutos.
BYTE m_btGGASecond;	Segundos.
double m_dGGALatitude	<0 = Sur, >0 = Norte.
double m_dGGALongitude;	<0 = Occidente, >0 = Oriente.
BYTE m_btGGAGPSQuality;	0 = Localización no disponible. 1 = Modo sps del GPS. 2 = GPS diferencial, modo sps, localización válida. 3 = Modo pps del GPS, localización válida.
BYTE m_btGGANumOfSatsInUse;	Número de satélites en uso.
double m_dGGAHDOP;	Dilución de posición horizontal.
double m_dGGAAltitude;	Altitud (metros), altura promedio del nivel del mar (geoide)
DWORD m_dwGGACount;	Registro de las sentencias GGA leídas.
int m_nGGAOldVSpeedSeconds;	
double m_dGGAOldVSpeedAlt;	
double m_dGGAVertSpeed;	

Variables utilizadas para los datos GPGSA

BYTE m_btGSAMode;	M = Manual A = Automático 2D/3D
BYTE m_btGSAFixMode;	Modo de la localización 1 = Localización no disponible 2 = 2D 3 = 3D
WORD m_wGSASatsInSolution[NP_MAX_CHAN]	Identificador de los satélites usados en la solución.
double m_dGSAPDOP;	Dilución de posición en precisión.
double m_dGSAHDOP;	Dilución de posición horizontal.
double m_dGSAVDOP;	Dilución de posición vertical.
DWORD m_dwGSACount;	Registro de las sentencias GSA leídas.

Variables utilizadas para los datos GPGSV

BYTE m_btGSVTotalNumofMsg;	Número total de mensajes de la sentencia.
WORD m_wGSVTotalNumSatsInView;	Número total de satélites vistos.
CNPSatInfo m_GSVSatInfo[NP_MAX_CHAN]	Información del satélite como PRN, elevación, azimut y SNR
DWORD SVCount;	Registro de las sentencias GSV leídas.

Variables utilizadas para los datos GPRMB

BYTE m_btRMBDataStatus;	A = Dato válido V = Advertencia de navegación del receptor.
double m_dRMBCrosstrackError;	Error de travesía en millas náuticas.
BYTE m_bRMBDirectionToSteer;	Dirección guía, Derecha/Izquierda.
CHAR m_lpszRMBOrignWaypoint[NP_WAYPOINT_ID_LEN];	Identificador del punto de camino de origen.
CHAR m_lpszRMBDestWaypoint[NP_WAYPOINT_ID_LEN];	Identificador del punto de camino de destino.
double m_dRMBDestLatitude;	Latitud del punto de camino de destino.
double m_dRMBDestLongitude;	Longitud del punto de camino de destino.
double m_dRMBRangeToDest;	Rango al destino en millas náuticas.
double m_dRMBBearingToDest;	Marcación al destino en

	grados verdaderos.
double m_dRMBDestClosingVelocity;	Velocidad de aproximación al destino en nudos.
BYTE m_btRMBArrivalStatus;	A = Entrando al circulo de arribo V = Fuera del circulo de arribo.
DWORD m_dwRMBCount;	Registro de las sentencias RMB leídas.

Variables utilizadas para los datos GPRMC

BYTE m_bRMCHour;	Hora.
BYTE m_bRMCMinute;	Minuto.
BYTE m_bRCMSecond;	Segundos.
BYTE m_btRMCDataValid;	A = Dato Válido V = Advertencia de navegación del receptor.
double m_dRMCLatitude;	Latitud actual.
double m_dRMCLongitude;	Longitud actual.
double m_dRMCGroundSpeed;	Velocidad en tierra en nudos.
double m_dRMCCourse;	Curso en tierra en grados verdaderos.
BYTE m_btRMCDay;	Día.
BYTE m_btRMCMonth;	Mes.
BYTE m_wRMCYear;	Año.
double m_dRMCMagVar;	Variación magnética en grados, Este (+) y Oeste (-).
DWORD m_dwRMCCount;	Registro de sentencias RMC leídas.

Variables utilizadas para los datos GPZDA

BYTE m_bZDAHour;	Hora.
BYTE m_btZDAMinute;	Minuto.
BYTE m_btZDASecond;	Segundos.
BYTE m_btZDADay;	Día (1 – 31)
BYTE m_btZDAMonth;	Mes (1 – 12)
WORD m_wZDAYear;	Año.
BYTE m_btZDALocalZoneHour;	Hora de zona local (0 a +/- 13)
BYTE m_btZDALocalZoneMinute;	Minuto de zonal local (0 a 59)
DWORD m_dwZDACount;	Registro de sentencias ZDA leídas.

*void ProcessGPZDA(BYTE *pData);*	Procesa los datos que trae el comando; para esto recibe un apuntador o puntero a los datos. Esta función también existe para los demás

	comandos.
BOOL IsSatUsedInSolution (WORD wSatID);	Función que determina si el satélite ayuda a definir la posición.
void Reset ();	Todas las variables vuelven a sus datos iniciales.
*BOOL GetField(BYTE *pData, BYTE *pField, int nFieldNum, int nMaxFieldLen);*	Esta función se encarga de analizar de forma específica los datos, para esto recibe un puntero a ellos, un apuntador para recorrer los campos, una variable tipo entero que trae el valor del campo, una variable que trae la longitud máxima del campo, esto corresponde a los parámetros de esta función descritos de izquierda a derecha.
*BOOL ProcessCommand (BYTE *pCommand, BYTE *pData);*	Recibe el comando y los datos que trae la sentencia.
void ProcessNMEA (BYTE btData);	Describe gramaticalmente cada bit de datos.
*BOOL ParserBuffer (BYTE *pBuff, DWORD dwLen);*	Recibe un puntero al buffer y la longitud de la sentencia.
CNMEAParser(); *virtual ~CNMEAParser();*	Constructor y destructor de la clase CNMEAParser().

_-

8. <u>NMEAParser.ccp : Implementación de la clase CNMEAParser</u>

#define MAXFIELD 25	Máxima longitud del campo de los datos.
CNMEAParser::CNMEAParser() *{*	Constructor
m_nState = NP_STATE_SUM;	Se inicializa m_nstate con lo que haya en el comienzo del mensaje.
m_dwCommandCount = 0;	Se inicializa el contador de sentencias.
Reset(); *}*	
*BOOL CNMEAParser::ParseBuffer(BYTE *pBuff, DWORD dwLen)* *{* *for (DWORD i = 0, i<dwLen; i++)*	Este es un for que hace uso del contador i para recorrer las posiciones de Buffer y del dwLen para saber cual es la longitud del Buffer y así saber

	cuando pasar.
{ *ProcessNMEA(pBuff[i]);*	Ejecuta la función ProcessNMEA(); y le envía cada carácter del Buffer.
} *return TRUE;* *}*	Retorna un valor positivo o verdadero cada vez que se compila esta función.
void CNMEAParser::ProcessNMEA(BYTE btData) *{* *switch(m_nstate)*	Revisa en que estado esta m_nstate.
{ *case NP_STATE_SOM;*	En caso que se encuentre al inicio de la sentencia se realizan los siguientes pasos.
if(btData==’$’) *{*	Compara si el bit a analizar es el que encabeza todas las sentencias NMEA. Si es así entonces:
m_btChecksum=0;	Resetea el checksum.
m_wIndex=0;	Resetea el índice
m_nState=NP_STATE_CMD; *}*	La variable m_nState cambia al estado de comando.
break;	Salida de la función.
case NP_STATE_CMD;	Si se encuentra el comando entonces:
if(btData!=’,’&&btData!=’’)*	Compara si el bit de datos actual es diferente de “,” y “*” entonces realiza las siguientes operaciones.
m_pCommand[m_wIndex++]=btData;	Se le asigna al bit en cuestión una posición en la variable m_pCommand[]. Por ejemplo: GPGGA m_pCommand[1]=”G” m_pCommand[2]=”P” m_pCommand[3]=”G” m_pCommand[4]=”G” m_pCommand[5]=”A” G P G G A m_pCommand
m_bChecksum1=btData;	Chequeo para evitar desbordamiento del comando.
if(m_wIndex>=NP_MAX_CMD_LEN) *{* *m_nState=NP_STATE_SOM;* *}*	Si el índice es mayor o igual que la longitud máxima entonces haga esta asignación para continuar con

	el análisis de la sentencia.
else	El comando es terminado.
{	
m_pCommand[m_wIndex]='\0'	
m_btChecksum ^=btData;	Compara btData y m_btChecksum, si ambos son 1 o 0 se guarda en checksum 0 si son diferentes se guarda en checksum 1.
m_wIndex=0;	Esta línea existe porque se debe reinicializar el índice cuando comenzamos otro estado u otra parte de la sentencia.
m_nState=NP_STATE_DATA;	Esta asignación es para pasar al estado de datos.
}	
break;	Salir del switch.
case NP_STATE_DATA	Si estamos en el estado de datos entonces se hace una comparación.
if(btData=='')*	Bandera del checksum.
{	
m_pdata[m_wIndex]='\0';	Comando terminado.
m_nState=NP_STATE_CHECKSUM1;	Paso al estado de checksum1.
}	
else	No hay bandera, entonces almacena datos.
{	
if(btData=='\r')	Si btData es igual '\r', CR Retorno de línea.
{	
m_pData[m_wIndex]='\0';	Termina comando.
ProcessCommand(m_pCommand,m_pData);	Envía el comando y los datos para que sean procesados.
m_nState=NP_STATESOM;	Va al comienzo del mensaje.
return;	Salir de la función.
}	
Se almacena datos y calcula checksum.	
m_btChecksum^=btData;	Almacena datos y calcula checksum. Si son iguales la sentencia ha sido transferida exitosamente.
m_pData[m_wIndex]=btData	Esta asignación sirve para recorrer y llenar los campos del arreglo de datos con los bits que están llegando.
if(++m_wIndex)=NP_MAX_DATA_LEN;	Chequeo para evitar

{	desbordamiento del buffer.
m_nState=NP_STATE_SOM;	
}	
}	
break;	
case NP_STATE_CHECKSUM_1:	
{	
if((btData-‘0’)≤a)	
{	
m_btReceivedChecksum=(btData-‘0’)<<4;	
}	
else	
{	
m_btReceivedChecksum=(btData-	
‘A’+10)<<4;	
}	
m_nState=NP_STATE_CHECKSUM_2;	Pasa al estado etiquetado como checksum2.
break;	Sale de switch.
case NP_STATE_CHECKSUM_2:	
if((btData-‘0’)<=9)	
{	
m_btReceivedChecksum1=(btData-‘0’);	
}	
else	
{	
m_btReceivedChecksum1=(btData-‘A’+10);	
}	
if(m_btChecksum==m_btReceivedChecksum)	
{	
ProcessCommand(m_pCommand,m_pData);	
}	
m_nState=NP_STATE_SOM;	
break;	
default : m_nState=NP_STATE_SOM;	

Procesamiento de las sentencias

*BOOL CNMEAParser::ProcessCommand(BYTE *pCommand, BYTE *pData)*	Se llama la función que realiza el procesamiento de los comandos de las sentencias.
*if(strcmp(char *)pCommand;”GPGGA”==NULL)*	Compara caracter a caracter dos cadenas, y toma los siguientes valores dependiendo del caso. Cad1>Cad2→ El resultado es positivo (>0).

	Cad1<Cad2→ El resultado es negativo (<0). Cad1=Cad2→ El resultado es cero (0, NULL)

Si el comando pasado por medio del apuntador pCommand es GPGGA entonces,

ProcessGPGGA(pData); *}*	Se pasan los datos por medio del puntero pData para que se realice a estos el proceso pertinente.
else *if…*	Sino sigue comparando para ver, a que comando pertenecen los datos para realizarles el proceso adecuado.
m_dwCommandCount++,	Esta variable cuenta las sentencias procesadas.
return TRUE; *}*	
*BOOL CNMEAParser::GetField(BYTE *pData, BYTE *pField, int nFieldNum, int nMAxFieldLen)* *{*	Esta función obtiene el campo específico en una cadena o sentencia NMEA. - *pData : Puntero a la cadena NMEA - *pField : Puntero al campo retornado nfieldNum : Comienzo del campo a conseguir. - nMaxFieldLen : Cantidad máxima de bytes que pField puede manejar.

Validación de Parámetros

if(pData==NULL||pField==NULL||nMaxFieldLen<=0	Esta condición revisa si se terminó la cadena de caracteres para salir de la función con:
return False; *}*	

Las siguientes inicializaciones nos llevan al principio del campo seleccionado.

int i=0; *int nField=0;* *while (nField!=nFieldNum&&pData[i])* *{*	Crea un ciclo mientras el número de campo sea diferente de la posición de los datos.
if(pData[i]=',') *{* *nField++*	Si el caracter en la posición i de la cadena de datos es igual a uno como ',' entonces sume 1 al número

}	de campo para continuar con el proceso.
i++	Al igual hay que sumar 1 para seguir avanzando posiciones en los datos.
if(pData[i]==NULL)	Esto se presenta cuando se termina la cadena de caracteres.
{	
pField[0]='\0';	Para determinar el primer campo.
return False;	Sale de la función y retorna un valor negando que existan o retornen datos para procesar.
}	
}	
if(pData[i]==',' || pData[i]=='')*	Esta condición es para enmarcar los datos dentro de los caracteres reservados ',' y '*'.
{	
pField[0]='\0';	
return FALSE;	

Las siguientes instrucciones copian los datos en un campo específico.

int i2=0;	Inicializa el contador del field.
while (pData[i]!=','&&pData[i]!=''&&pData[i])*	Mientras esta condición se cumpla tendremos solo datos.
{	
pField[i2]=pData[i];	Copia de los datos campo a campo e incremento de los contadores.
i2++; i++;	
if(i2>=nMaxFieldLen)	Chequea y corrige si el campo es más grande que el parámetro.
{	
i2=nMaxFieldLen-1;	
break;	Salida del if.
}	
pField[i2]='\0';	Da visto bueno para procesar datos.
return TRUE;	

Reset. Resetea todos los datos NMEA, asignando a todas las variables sus valores iniciales o por defecto.

Ahora se chequea si los IDs de los satélites suministrados son usados en la solución en el GPS.

BOOL CNMEAParser::IsSatUsedInSolution(WORD wSatID)	Esta función recibe la identificación de un satélite.
{	
if(wSatID==0)	Se dispone así porque no existe el satélite 0.
return FALSE;	
for(int i00; i<12;i++)	El programa tiene capacidad para 36 satélites, gracias a los cales que puede manejar, pero
{	

	por cuestiones estéticas de la interfase gráfica de usuario se visualizan solo 12.
if(wSatID==m_GSASatsInSolution[i]) *{* *return TRUE;* *}* *}*	Compara si el satélite pasado hace parte de aquellos que resuelven la posición, si esto es así, entonces retorna un valor positivo o verdadero y sale de la función.
} *return FALSE;* *}*	Esto ocurre cuando el satélite no hace parte del a solución.

Sentencia GPGGA

Global Positioning System Fix Data, Time, Position and fix related data for a GPS receiver. (*Datos de localización del sistema de posicionamiento global, datos relacionados de tiempo, posición y localización para un receptor GPS*)

$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47

Donde,

GPGGA	Comando de la sentencia
123519	Localización fue hecha a las 12:35:19 UTC.
4807.038,N	Latitud 48º 07.038’ N
01131.000,E	Longitud 11º 31.000’ E
1	Calidad de la localización (señal) 0 → Valor inválido 1 → Localización en modo SPS del GPS. 2 →Localización en modo SPS del GPS diferencial. 3 → Localización en modo PPS del GPS.
08	Número de satélites que han sido rastreados.
0.9	Dilución de posición horizontal.
545.4,M	Altitud en metros, por encima del nivel promedio del mar.
46.9,M	Altura del geoide (nivel promedio del mar) por encima de la elipsoide WGS84.
Campo vacío	Tiempo en segundos desde la última actualización de un GPS diferencial.
Campo vacío	Numero ID (identificación) de la estación GPS diferencial.
*47	Dato del checksum, siempre comienza con *.

*void CNMEAParser::ProcessGPGGA(BYTE *pData)*	Se llama la función que procesará la sentencia GPGGA.
BYTE pField[MAXFIELD];	Se crea un apuntador para campos de la sentencia.
CHAR pBuff[10];	Un buffer para datos.
if(GetField(pData,pField,0,MAXFIELD))	Le pasamos a esta función un puntero a los datos, otro a los campos, el número del campo en este caso 0 (porque el tiempo es el primer dato que tiene este comando) y la máxima longitud del campo y esperamos de ella el visto bueno para procesar los campos
{	
//Hora	Subcampo 0 del campo 0.
pBuff[0]=pField[0];	
pBuff[1]=pField[1];	Subcampo 1 del campo 1
pBuff[2]='\0';	Termino del buffer para hora.
m_btGGAHour=atoi(pBuff);	Convierte caracteres ASCII a enteros.
//Minutos	En este caso asigna al buffer los subcampos 2 y 3 del campo 0 correspondientes a los minutos, para su debida conversión y para guardarlo en una variable que se pueda visualizar como m_btGGAMinute.
pBuff[0]=pField[2];	
pBuff[1]=pField[3];	
pBuff[2]='\0';	
m_btGGAMinute=atoi(pBuff);	
//Segundos	Es el mismo procedimiento que con la hora y los minutos.
//Latitud	Ahora pasamos un apuntador a los datos y a los subcampos del campo 1 que pertenece a la latitud.
if(GetField(pData,pField,1,MAXFIELD))	
{	
m_dGGALatitude=atof((CHAR)pField+2)/60.0;*	Conversión de ASCII a □lota, de los dos primeros

	subcampos, o sea 0 y 1; y se divide por 60.0 para mostrarlos adecuadamente.
pField[2]='\0';	Fin de campo.
m_dGGALatitude+=atof((CHAR)pField);*	Actualiza la variable m_dGGALatitude.
}	
if(GetField(pData,pField,2,MAXFIELD))	Entramos al campo 2 que nos dice si la latitud es Norte o Sur
{	
if(pField[0]=='S')	Traemos el Subcampo 0, que puede ser N o S. Si es N dejamos positivo el valor de la latitud. Si es S entonces:
{	
m_dGGALatitude=-m_dGGALatitude;	Anteponemos un signo negativo al valor de la latitud.
}	
}	
//Longitud	La longitud se encuentra en el campo 3.
if(GetField(pData,pField,3,MAXFIELD))	
{	
m_dGGALongitude=atof((CHAR)pField+3)/60.0;*	Conversión de ASCII a lota, de los 3 primeros subcampos. Es decir, pField[0], pField[1], pField[2].
pField[3]='\0';	Termino del Subcampo.
m_dGGALongitude+=atof((CHAR)pField);*	Actualiza la variable m_dGGALongitude.
}	
if(GetField(pData,pField,4,MAXFIELD))	Esta línea nos trae el Este (E) o el Oeste (W).
{	
if(pField[0])='W')	Si estamos en Colombia estaremos en el occidente y por lo consiguiente entraremos en esta condición que muestra de forma negativa la longitud.
{	
m_dGGALongitude=-m_dGGALongitude;	
}	
}	
//Calidad de la señal	Aquí encontramos los diferentes tipos de calidad de señal, es un dígito entre 0 y 3.
if(GetField(pData,pField,5,MAXFIELD))	
m_dGGAGPSQuality=pField[0] – '0';	
//Satélites en uso	Número de satélites usado que puede ir de 00 a 12.
if(GetField(pData,pField,6,MAXFIELD))	
{	
pBuff[0]=pField[0];	De 10 satélites vistos no

pBuff[1]=pField[1]; *pBuff[2]='\0';*	necesariamente debe todos ser parte de la solución. En pruebas el número máximo de satélites fueron 6.
m_btGGANumSatsinUse=atoi(pBuff);	Conversión del buffer de formato ASCII a entero (int).
//HDOP	Dilución de precisión horizontal, es un factor de corrección de posición debido a la geometría del satélite. Este factor se utiliza para ganar posición.
if(GetField(pData,pField,7,MAXFIELD)) *{* *m_dGGAHDOP=atof((CHAR*)pField);*	Esta conversión es debido a que el HDOP tiene la siguiente forma, x.x y viene dado en metros
//Altitud *if(GetField(pData,pField,0,MAXFIELD))* *{* *m_dGGAAltitude=atof((CHAR*)pField);*	Esta es nuestra posición en referencia al nivel del mar por medio de la elipsoide WGS-84 y por sus características es una variable float

A continuación calcularemos la velocidad de ascenso o descenso por medio de las diferencias en segundos y la altitud de las diferentes actualizaciones. Ejemplo:

$$\Delta t = t_{anterior} - t_{actual}$$

$$V = (Altitud_{anterior} – Altitud_{actual})/\Delta t$$

```
Int
nSeconds=(int)m_bGGAMinute*60+(int)m_btGGASecond;
if(nSeconds>m_nGGAOldVSpeedSeconds)
{
double dDiff=(double)(m_nGGAOldVSpeedAl t- nSeconds);
double dVal=dDiff/60.0;
if(dVal != 0.0)
{
m_dGGAVertSpeed=(m_dGGAOldVSpeedA- m_dGGAAltitude)/dVal;
}
}
m_dGGAOldVSpeedAlt=m_dGGAAltitude;
m_nGGAOldVSpeddSecond=nSeconds;
```

m_dwGGACount++;	Registro de las

} sentencias GGA leídas.

Sentencia GPGSA

GPS DOP and active satellites (*GPS DOP y satélites activos*)

$GPGSA,A,3,01,20,19,13,,,,,,,,,40.4,24.4,32.2*0A

Donde,

GPGSA	Comando de la sentencia
A	Modo Automático o Manual.
3	Modo de localización 1 → Sin localización 2 → Localización 2D 3 → Localización 3D
01	ID del satélite usado en el canal 1
20	ID del satélite usado en el canal 2
19	ID del satélite usado en el canal 3
13	ID del satélite usado en el canal 4
,,,,,,	ID de los satélites usados en los canales 5 – 12. Se usa valor nulo para los satélites no usados.
40.4	Dilución de precisión en posición dado en metros
24.4	Dilución de precisión horizontal dado en metros
32.2	Dilución de precisión vertical dado en metros
*0A	Dato del Checksum

```
void (NMEAParser::ProcessGPGSA(BYTE *pData)
{
BYTE pField[MAXFIELD];
CHAR pBuff[10];
//Modo
```

Que puede ser manual o automático.

```
if(GetField(pData,pField,0,MAXFIELD)
{
m_btGSAMode=pField[0];
}
```

Hasta aquí m_btGSAMode ha recibido un byte con una M o una A que el NMEAParserDemoDlg.ccp convertirá en Manual o Automático para un mejor entendimiento.

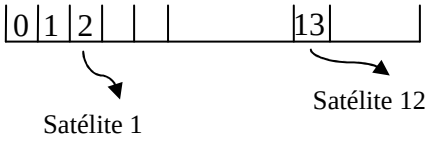
```
//Modo de localización
if (GetField(pData,pField,1,MAXFIELD))
{
m_btGSAFixMode=pField[0]-'0';
```

El campo 1 del comando GSA solo hace uso de un subcampo, toda vez que su parámetro es sencillamente un 0, 1, 2 ó 3 y la interpretación de

```
}
//Satélites activos o vistos
for(int i=0;i<12;i++)
{
if(GetField(pData,pField,2+i,MAXFIELD))
{
```

este valor se hace en el Dlg.cpp.
Son 12 los satélites de los cuales vamos a desplegar información.

Aquí pasamos un puntero a los datos, un apuntador de campo, la máxima longitud del campo y 2+i significa el campo a asignado a cada uno de los satélites, empezando desde el campo 2 (pues 0 es modo A o M y 1 es el modo de localización) y terminando en el campo 13.




```
pBuff[0]=pField[0];
pBuff[1]=pField[1];
pBuff[2]='\0';
m_wGSASatsInSolution[i]=atoi(pBuff);
}
else
m_wGSASatsInSolution[i]=0;


//PDOP
if(GetField(pData,pField,14,MAXFIELD))
{
m_dGSAPDOP=atof((CHAR*)pField)
}
else
{
m_dGSAPDOP=0.0


m_dwGSACount++;
}
```

Ejemplo: Satélites 8 y 24
0 2
8 4
Conversión de ASCII a entero.

No hay satélites a la vista, esto es porque nos encontramos en dentro de un recinto, generalmente se debe salir a campo abierto.

Dilución de precisión de posición
Convierte este valor en un float y lo guarda en el subcampo 14 del comando GSA.

No existe señal, no se tiene ese dato. Se procede de la misma manera con HDOP (dilución de precisión horizontal) y VDOP (dilución de precisión vertical).

Registro de las sentencias GSA leídas.

Sentencia GPGSV

Satellites in view (*Satélites vistos*)

El propósito de esta sentencia es mostrar los datos más significativos de los satélites que forman parte de la solución. Cada sentencia muestra los datos de 4 satélites y dependiendo del número de satélites percibidos por el GPS, se necesitarán más sentencias para recibir la información completa.

$GPGSV,3,1,11,03,03,111,00,04,15,270,00,06,01,010,00,13,06,292,00*74
$GPGSV,3,2,11,14,25,170,00,16,57,208,39,18,67,296,40,19,40,246,00*74
$GPGSV,3,3,11,22,42,067,42,24,14,311,43,27,05,244,00,,,,*4D

Donde,

GPGSV	Comando de la sentencia
3	Número total de mensajes de este tipo en este ciclo. En este ejemplo es 3, es decir que se esperan 3 sentencias GSV consecutivamente.
1	Número del mensaje. Indica cual es el mensaje que se esta transmitiendo dentro del ciclo.
11	Número total de satélites vistos.
03	ID del primer satélite (PRN)
03	Elevación en grados (0 – 90)
111	Azimut, dado en grados desde el norte verdadero. (0-359)
00	SNR en dB (0 – 99), Es nulo cuando el satélite no es rastreado.
04	ID del segundo satélite (PRN)
15	Elevación en grados (0 – 90)
270	Azimut, dado en grados desde el norte verdadero. (0-359)
00	SNR en dB (0 – 99), Es nulo cuando el satélite no es rastreado.
06	ID del tercer satélite (PRN)
01	Elevación en grados (0 – 90)
010	Azimut, dado en grados desde el norte verdadero. (0-359)
00	SNR en dB (0 – 99), Es nulo cuando el satélite no es rastreado.
13	ID del cuarto satélite (PRN)
06	Elevación en grados (0 – 90)
292	Azimut, dado en grados desde el norte verdadero. (0-359)
00	SNR en dB (0 – 99), Es nulo cuando el satélite no es rastreado.
*74	Dato del Checksum

En este caso se necesitaron 3 sentencias, para mostrar los datos de los 11 satélites vistos (Solo fue explicada la primera de las tres sentencias del ciclo).

*void CNMEAParser::ProcessGPGSV(BYTE *pData)*	Se llama la función que procesará la sentencia GSV.
{	
INT nTotalNumOfMsg,nMsgNum;	Número total de mensajes y número de mensaje.
BYTE pField[MAXFIELD]	
{	
*nTotalNumOfMsg=atoi((CHAR *)pField);*	Se convierte una secuencia ASCII en un entero.

if(nTtalNumOfMsg>9 || nTotalNumOfMsg<0) *return;*	A continuación nos aseguramos que el número total de mensajes es válido. Esto es usado para calcular índices dentro del mismo arreglo (sentencia).

Si el número de mensajes es mayor que 9 o menor que 0 no es válido y debe salir de la función.
¿Por qué no es válido? Porque solo se conoce la existencia de 32 satélites que pasándolos de 4 en 4 por sentencia se necesitaría a lo más 8 sentencias para tener información de ellos. La sentencia y el programa no acepta por integridad que hayan por ejemplo -2 satélites por esto si se cumple esta condición estamos frente a un error.

*if(nTotalNumOfMsg<1||nTotalNumOfMsg*4>=NP_MAX_CHAN)* *{* *return;* *}*	Esta condición cumple la misma función que la anterior. Aquí se toma en cuenta el error que sea 0, pues no habría señal y el hecho que tengamos más de 35 satélites vistos.
//Número de mensaje	Que puede ir de 1 a 4.
if(GetField(pData,pField,1,MAXFIELD)) *{* *nMsgNum=atoi((CHAR*)pField);* *{*	Se obtiene el valor del campo y se convierte de ASCII a entero.
if(nMsgNum>9 || nMsgNum<0) *return;* *}*	Ahora se asegura que el número de mensaje es válido.
//Número total de satélites vistos *if(GetField(pData,pField,2,MAXFIELD))* *{* *m_wGSVTotalNumSatsInView=atoi((CHAR*)pField);* *}*	Convertimos a entero porque nunca visualizaremos medios satélites.
//Datos de los satélites *for(int i=0; i<4;i++)* *{*	Es un ciclo que se repite 4 veces porque pasamos información de 4 satélites por sentencia.
//ID del satélite (PRN) *if(GetField(pData,pField,3+4*i,MAXFIELD))* *{*	Esta sentencia recibe el ID o PNR del primer satélite y lo ubica en la posición 3, y los

	demás los va ubicando cada 4 espacios, pues hay que dejar campo para el azimut, la elevación y la señal de cada satélite.
*m_GSVSatInfo[i+(nMsgNum-1)*4].m_wPRN=atoi((CHAR*)pField);* *}*	Guardamos la identificación del satélite, en el atributo m_wPRN del objeto m_GSVSatInfo, que es una instancia de la clase CNPSatInfo encargada de manipular la información de los satélites
else *{* *m_GSVSatInfo[i+(nMsgNum-1)*4.m_wPRN=0;* *}*	Si no se pudieron procesar los campos se asigna a este atributo un valor NULL.
	Este procedimiento se le practica a los datos de cada satélite, que son: - Identificador del satélite (PRN) - Elevación - Azimut - SNR Lo único es que el espaciado lo iniciaran desde la posición 4,5 y 6 respectivamente.
*m_GSVSatInfo[i+(nMsgNum-1)*4.m_bUsedInSolution= IsSatUsedInSolution(m_GSVSatInfo[i+(nMsgNum-1)*4.m_wPRN);* *}* *}*	Se le pregunta a la función IsSatUsedInsolution, si el satélite del cual estamos clasificando la información hace parte de los que no son usados para determinar nuestra posición, ella nos

	responde verdadero o falso y lo guardamos en otro atributo del satélite.
m_dwGSVCount++;	Registro de las sentencias GSV leídas.

Sentencia GPRMB

Recommended Minimum Navigation Information (*Información mínima recomendada de navegación*)

$GPRMB,A,0.66,L,003,004,4917.24,N,12309.57,W,001.3,052.5,000.5,V*0B

Donde,

GPRMB	Comando de la sentencia
A	Estado de datos A = Ok V = Advertencia
0.66	Error en la trayectoria en millas náuticas
L	Dirección para la guía, Left o Rigth
003	ID del waypoint de origen
004	ID del waypoint de destino
4917.24	Latitud del waypoint de destino, 49 grados 17.24 minutos.
N	Norte o Sur
12309.57	Longitud del waypoint de destino, 123 grados 09.57 minutos
W	Oeste o Este
001.3	Rango al destino, en millas náuticas
052.5	Marcación del destino en grados verdaderos
000.5	Velocidad de aproximación al destino en nudos.
V	Estado de arribo A = Entrando al circulo de arribo V = No se ha entrado al circulo de arribo
*0B	Datos del Checksum

*void CNMEAPARSER::ProcessGPRMB(BYTE *pData)* *{* *BYTE pField[MaXFIELD];* *//Estado de los datos* *if(GetField(pData,pField,0,MAXIFIELD))* *{*	Se llama la función que procesará la sentencia RMB. Si GetField retoma un valor positivo, es porque se puede continuar con la

m_btRMBDataStatus=pField[0]; *}*	asignación de datos de estado a la variable m_btRMBStatus.
else *{* *m_btRMBDataStatus='V';* *}*	Esto implica la asignación de una advertencia de cuidado.

De forma similar, se procede con el error de travesía, la dirección para guía, el identificador del waypoint de origen, de destino, el rango al destino, la orientación al destino, velocidad de aproximación y estado de arribo. Si la función GetField los pudo procesar se empiezan a hacer asignaciones a variables que se crearon con el asistente y que son susceptibles de ser desplegadas, sino a esas variables se les asigna valores por defecto o iniciales.

Para entender la forma del proceso de la latitud y longitud de destino, puede observar la función processGPGGA.

Sentencia GPRMC

Recommended Minimum Navigation Information (*Información mínima recomendada de navegación*)

$GPRMC,225446,A,4916.45,N,12311.12,W,000.5,054.7,191194,020.3,E*68

Donde,

GPRMC	Comando de la sentencia
225446	Tiempo de la localización 22:54:46 UTC
A	Estado A = Posición válida V = Advertencia
4916.45	Latitud 49 grados 16.45 minutos
N	Norte o Sur
12311.12	Longitud 123 grados. 11.12 minutos
W	Oeste u Este
000.5	Velocidad en tierra, en nudos
054.7	Trayectoria bien hecha en grados verdaderos
191194	Fecha de la localización (19 de Noviembre de 1994)
020.3	Variación magnética en grados
E	Este u Oeste
*68	Datos del Checksum

*void CNMEAParser::ProcessGPRMC(BYTE *pData)* *{* *CHAR pBuff[10];* *BYTE pField[MAXFIELD];*	Se llama la función que procesará la sentencia RMC.

Para observar el procesamiento del tiempo, la latitud, longitud, observe el procedimiento realizado con el comando GPGGA. En cuanto a la validación de datos, curso en tierra y velocidad son similares a los datos procesados por el comando GPRMB. Este comando además, tiene información sobre la fecha:

if(GetField(pData,pField,8,MAXFIELD)) *{*	Primero pasamos al buffer los campos 0 y 1 que corresponden al día. Acompañados del caracter '\0' para indicar que termina el buffer para este dato. Luego llenamos el buffer con los campos 2 y 3 que son los que traen información, del mes y se realiza el mismo procedimiento de cerrado de buffer para este dato. Por último se ubica en el buffer los campos 4 y 5 del año. Se pasa a hacer la conversión a enteros de los caracteres ASCII pasados por el buffer.
m_wRMCYear +=2000	Como la sentencia solo nos trae los últimos dos dígitos del año, por ejemplo: 04, tenemos que realizar un tipo de "calibración" sumando 2000 para que se pueda visualizar de forma entendible el año.

<u>Sentencia GPZDA</u>

Time & Date - UTC, day, month, year and local time zone (*Tiempo y fecha – UTC, día, mes, año y tiempo de zona local*)

$GPZDA, 024611.08,25,03,2002,00,00*6A

Donde,

GPZDA	Comando de la sentencia
024611.08	Tiempo (UTC) 02:46:11.08
25	Día (01 – 31)

03	Mes (01 – 12)
2002	Año (Formato de 4 dígitos)
00	Descripción de la zona horaria local en horas (00 a ± 13)
00	Descripción de la zona horaria local en minutos (00 a 59)
*6A	Datos del Checksum

*void CNMEAParser::ProcessGPZDA(BYTE *pData)* *{*	Se llama la función que procesará la sentencia ZDA.

El tiempo que viene en este comando es procesado de la misma forma que el comando GGA. El día, mes y año tienen un tratamiento diferente al RMC, pues presentan un diferente formato porque cada uno viene en un subcampo distinto y el año no necesita conversión pues vienen los 4 dígitos.

Los campos 4 y 5 de este comando traen en formato de hora y de minutos, la zona horaria local donde estamos ubicados.

_-

9. <u>NMEAParserDemoDlg.h: Archivo de encabezamiento</u>

#include "Sio.h" *#include "NMEAParser.h"*	Incluye las definiciones hechas sobre el puerto (en Sio.h) y sobre el procesamiento de sentencias NMEA (en NMEAParser.h)
class CNMEAParserDemoDlg:Public CDialog *CSio m_Sio;* *CMEAParser m_NMEAParser;*	Se crean los objetos para manipular las funciones de puerto y descriptores.
public: *void UpdateText();*	Función para actualizar los static text que muestran el valor de las variables. Ejemplo: Longitud.
CNMEAParserDemoDlg(CWnd pParent=NULL)*	Constructor estándar
protected: *HICON m_hIcon;*	Se crea un objeto para manipular lo respecto al icono.
BOOL OnIntDialog();	Esta función inicializa el cuadro de dialogo y las principales funciones de nuestra aplicación.

void OnSysCommand(UINT nID, LPARAM iParam);	Esta función es llamada cuando el usuario ejecuta un comando de menú de control o selecciona los botones minimizar y maximizar. En otras palabras esta función es la que hace sensible el cuadro de dialogo a las acciones del usuario.
void OnPaint();	Esta función implica y se encarga de la interfase gráfica de la aplicación o sea de imágenes, tamaños, etc.
HCURSOR OnQueryDragIcon();	Esta función se utiliza para solicitar información de posición, etc. cuando el icono es arrastrado.
OnTimer(UINT nIDEvent);	Esta es una función que realiza unos procesos periódicos.
OnResetData();	Se usa para volver a los valores iniciales o por defecto.
OnSetCom();	Función que es usada para la configuración del puerto.

_-
_-_-_-

10. <u>NMEAParserDemoDlg.Cpp: Define los comportamientos de la clase para la aplicación</u>

Se crea un objeto (m_Sio) para luego poder manipular las funciones de la clase que definimos para comunicación serial.
Se crea una instancia (m_NMEAParser) de la clase CNMEAParser que analiza gramaticalmente los campos del protocolo.
Actualiza los static text que son los cuadros que permiten visualizar información, ejemplo: longitud, latitud, etc… sin poderlos manipular
Se define de forma estándar el constructor del cuadro de dialogo.
Se determinan las funciones de contexto del cuadro de dialogo.

#include "Stdafx.h"	Este archivo se incluye para el despliegue de la interfaz por ventanas o cuadros de dialogo.
#include "NMEAParserDemo.h"	Se definen las funciones públicas para la construcción e inicialización de la aplicación.

#include “NMEAParserDemoDlg.h”	Encontramos la librería para comunicación por puerto serial y para interpretar el protocolo NMEA 0183.
#include “CommSettingsDlg.h”	Se declaran dos variables de tipo entero para capturar los valores de puerto y tasa de transferencia.
enum TIMER_ID *{*	Para definir timers generales o estándares para drivers de dispositivos.
TIMER_ID_POLL_COMM=1;	Contador para el puerto com.
TIMER_ID_UPDATE_TEXT;	Para actualizar datos.
BOOL CNMEAParserDemoDlg::OnInitDialog() *{*	Se llama la función de OnInitDialog.
cDialog::OnInitDialog();	Inicializa el cuadro de dialogo.
SetIcon(m_hIcon,TRUE);	Se configuran las operaciones del icono del cuadro de dialogo.
	Configura icono grande.
SetIcon(m_hIcon,FALSE);	Configura icono pequeño.
m_Sio.InitComm(1,4800,NOPARITY,8,ONESTOPBIT ;FALSE);	El objeto m_Sio llama a la función InitComm() para inicializar el puerto serial número 1, se le configura una tasa de transferencia de 4800 bits/segundos, que no se va a manejar paridad, que los bits de datos van a ser 8, el bit de parada es 1, y el error de bandera se inicializa inactivo.
SetTimer(TIMER_ID_POLL_COMM,100,NULL);	Se configura el reloj de conteo del puerto com cada cien milisegundos.
SetTimer(TIMER_ID_UPDATE_TEXT_1000,NULL);	Para configurar el cronómetro que actualiza los static text cada segundo.
return TRUE; *}*	Retorna un valor verdad--ero o positivo luego que se ha podido inicializar el cuadro de dialogo.
void CNMEAParserDemoDlg::OnPaint() *{* *}*	Este código lo necesitará para desplegar la ventana o en otras palabras dibujar el cuadro de dialogo. Usted lo podrá escribir o Visual C se lo

	genera si usted selecciona el modelo o arquitectura de soporte de documentos/vistas en una aplicación MFC.
HCURSOR CNMEAParserDemoDlg::OnQueryDragIcon() *{* *return (HCURSOR) m_hIcon;*	Esta función es llamada por el sistema cuando el cursor arrastra la ventana.
void CNMEAParserDemoDlg::Ontimer(UINT nIDEvent) *{* *switch(nIDEvent)* *{*	Este switch se utiliza para seleccionar el tipo de evento a realizar en el tiempo t. En este programa tenemos dos eventos a realizar periódicamente.
case TIMER_ID_POLL_COMM: *BYTE pBuff[256];*	El primero es leer los datos del puerto serial y enviarlos al analizador. Creamos el buffer para recibir los bytes por el puerto serial.
DWORD dwBytesRead;	Definimos una variable para recibir los bits leídos.
dwBytesRead=m_Sio.Read(pBuff,255);	Lo leído del buffer de datos lo almacena en dwBytesRead.
m_NMEAParser.ParseBuffer(pBuff,dwBytesRead);	Pasamos el analizador de sentencias NMEA, el buffer de datos y la variable dwBytesRead.
break;	Salir del switch
case TIMER_ID_UPDATE_TEXT: *UpdateText();* *break;* *}*	El segundo evento es actualizar los static text que muestra los valores de latitud, longitud, etc. al usuario.
cDialog::OnTimer(nIDEvent); *}*	Se llama así misa en forma de ciclo
void CNMEAParserDemoDlg::UpdateText() *{*	Esta función es llamada o ejecutada constantemente desde el timer.
CString str; *CString str2;*	Creamos cadenas de caracteres.
SetDlgItemInt(IDC_NMEA_RX_COUNT,m_NMEAParser.m_dwCommandCount);	Despliega y actualiza el contador de sentencias, que es un atributo del analizador o descriptor NMEA.

Para mostrar los principales datos que nos llegan por medio del comando GGA (latitud/longitud/altitud), damos las siguientes instrucciones:

str.Format("%f",m_NMEAParser.m_dGGAlatitud); *SetDlgItemText(IDC_LAT,str);*	Recibimos en la cadena de caracteres con formato "str" lo que hay en el atributo m_dGGALatitutde del analizador y luego pasamos a mostrar esa cadena en un static text que denominamos IDC_LAT. De igual forma hacemos con la longitud y la altitud.

En cuanto a la calidad de la señal del GPS, que también recibimos por el comando GGA se siguen los siguientes pasos:

switch(m_NMEAParser.m_btGGAGPSQuality) *{* *case 0 : str = _T("Fix not available"); break;*	Este switch revisa la calida de señal del GPS por medio del atributo m_btGGAGPSQuality del analizador de sentencias, y si este atributo esta en 0 es porque no se ha encontrado localización.
case 1 : str = _T("GPS sps mode"); break;	En caso que sea 1 es porque la localización se logró en modo sps y debe desplegar un mensaje con esta información.

Para el caso 2 se ha logrado una localización válida en modo sps para un GPS diferencial y para el caso 3 es una localización válida en modo pps. Si por algún motivo existiera un valor diferente a los mencionados anteriormente, se desplegaría un mensaje "Unknown" (desconocido). Los anteriores mensajes se muestran por medio del static text SetDlgItemText(IDC_GPS_QUAL,str).

Por otra parte el comando GSA nos trae información del modo de localización y el DOP (dilución de precisión).

switch (m_NMEAParser.m_btGSAMode) *{* *...* *}* *SetDlgItemText (IDC_GSA_MODE, str);*	Para poder visualizar la información del modo de localización, implementamos un selector de casos que nos dirá si es manual si recibe una M,

	automático si recibe una A o nos mostrará un "?" por defecto si el comando no trae esa información. Cualquiera que sea la selección se mostrará en el static text IDC_GSA_MODE.
switch(m_NMEAParser.m_btGSAFixMode) *{* *...* *}* *SetDlgItemText (IDC_GSA_FIX_MODE, str);*	De igual forma que la localización, estos pasos se implementan para el tipo de localización. En este caso si viene un 1 en el atributo m_btGSAFixMode, se mostrará el mensaje "Fix not available"; si es un 2, "2D Fix"; un 3 implicaría "3D Fix"; o por defecto sería "Unknown" (desconocido). Esto se muestra por el static text IDC_GSV_FIX_MODE
str.Format(_T("%.022f"), m_NMEAParser.m_dGSAVDOP); *SetDlgItemText(IDC_VDOP, str) ;*	En cuanto a la dilución de precisión vertical se mostrará en el static tex IDC_VDOP. De la misma forma se hace con el HDOP y PDOP.

Por otra parte la aplicación muestra información sobre los satélites vistos, por el GPS, gracias al comando GSA.

str=_T(" ");	Esto es una asignación de un espacio a la variable str.
for(int i=0;i<12;i++) *{* *str2.Format(2%02d", m_NMEAParser.m_wGSASatsinSolution[i]);*	Aquí se recorre los 12 campos del a variable m_wGSASatsInSolution, para extraer la identificación de los satélites usados en la solución.
str+=str2; *}*	str=str+str2; → Esto se realiza para dejar espacio entre el ID de un satélite y el otro.
SetDlgItemtext(IDC_SATS_USED_IN_SOL,str	Que muestre en el static text IDC_SATS_USED_IN_SOL lo que hay en la variable str.

También por el comando GSV podemos recibir información sobre el número total de satélites vistos, elevación, azimut, calidad de la señal.

SetDlgItemInt(IDC_SAT_IN_VIEW ,	Muestra cuanto son

m_NMEAParser.m_wGSVTotalMunSatInView) ;	los satélites vistos.
str=_TC(“ “); *for(i=0;i<m_NMEAParser.m_wGSVTotalSatInView;i++)* *{* *str2.Format(_T”%d\t”,m_NMEAParser.GSVSatInfo{[i].m_wPRN);* *str+=str2;* *}*	Se crea un ciclo para guardar en str2 el PRN de todos los satélites vistos. Se le adicionan espacios entre PRN de un satélite y el otro.
SetDlgItemText(IDC_GSV_PRN,str);	Se usa para mostrar los PRNs en el IDC_GSV_PRN.

De igual forma se hace con azimut, elevación y calidad de señal.

void CNMEAParserDemoDLG ::OnResetData() *{* *n_NMEAParser.Reset() ;* *}*	Esta función lo que hace es tomar todas las variables del programa y llevarlas a sus valores iniciales o por defecto.
void CNMEAParserDemoDlg::OnSetComm() *{* *CCommSettingsDlg dlg;*	Creamos un objeto dlg para manejar las variables del cuadro de dialogo para configurar puerto.
dlg.m_nBaud=m_Sio.m_nBaud;	Se muestra un checklist para que el usuario escoja la tasa de transferencia que puede ser 4800, 9600, 19200, 38400 ó 115200.
dlg.m_nPort=m_Sio.m_nPort;	En la variable m_nPort del cuadro de dialogo desplegamos las opciones de puerto que puede ir desde 1 hasta 16.
if(dlg.DoModal()==IDOK) *{* *m_Sio.Close();* *m_Sio.InitComm(dlg.m_nPort, dlg.m_nBaud, NONPARITY, 8, ONESTOPBIT, FALSE);* *}* *}*	Cuando el usuario dé click en el botón OK, se reinicializará el puerto COM con los valores seleccionados por el usuario en el cuadro de dialogo de configuración del puerto.

NMEA Parser Design

Monte Variakojis
monte@visualgps.net
Noviembre 1, 2002

La norma NMEA 0183 para la interfase de dispositivos electrónicos marítimos específica la estructura de la sentencia de datos NMEA así como las definiciones generales de las sentencias reconocidas. Sin embargo, la especificación no cubre la implementación ni el diseño. Este artículo aclarará sobre las tareas de diseño necesitadas para analizar sentencias NMEA. Se tratará de mostrar técnicas en análisis y chequeo de integridad de los datos. El artículo asume que el lector tiene conocimiento en diseño de software y tiene cierta experiencia en lenguaje de programación. Este artículo referenciará la especificación NMEA 0183 y se recomienda que la especificación NMEA 0183 sea adquirida como referencia.

Este artículo no hace asunción del medio como los datos NMEA son adquiridos. Las tecnologías aquí pueden ser aplicadas en datos recibidos de una capa de abstracción superior. Un simple ejemplo escrito en C++ demuestra este diseño del analizador.

Protocolo NMEA

Los datos NMEA son enviados en formato ASCII de 8 bits cuando el MSB es fijado en cero (0). La especificación también tiene un conjunto de caracteres reservados. Estos caracteres colaboran en el formato de la cadena de datos NMEA. La especificación también declara los caracteres válidos y da una tabla de estos caracteres que van desde HEX 20 a HEX 7E.

Como se establece en la especificación NMEA 0183 versión 3.01, el máximo número de caracteres deberán ser 82, consistiendo de un máximo de 79 caracteres entre el comienzo del mensaje “$” o “!” y el delimitador de final <CR><LF> (HEX 0D y 0A). El mínimo número de campos es uno (1).

Formato básico de la sentencia:

$aaccc,c--c*hh<CR><LF>

$ □ Comienzo de la sentencia

aaccc Campo de dirección / Comando

“,” Delimitador de campo (Hex 2C)

c--c Bloque de datos de la sentencia

* Delimitador del Checksum (HEX 2A)

hh Campo del Checksum (el valor hexadecimal representado en ASCII)

<CR><LF> Fin de la sentencia (HEX OD OA)

Analizador NMEA

La mayoría de los protocolos tienen una máquina de estado que rastea el estado del protocolo y cualquier error que pueda ocurrir durante la transferencia de datos. El analizador que estamos discutiendo está basado en una simple máquina de estado. Usando una máquina de estado el computador puede mantener fácilmente el rastro de donde él esta adentro del protocolo como también de recuperar algunos errores como desbordamiento de tiempo y errores de checksum.

El ejemplo del análisis es diseñado como si un buffer puede ser enviado al analizador junto con la longitud del buffer para maximizar la eficiencia del procesador del computador. Esto permitirá al computador analizar los datos cuando son recibidos. Ya que los datos NMEA son típicamente enviados a 4800 baudios, los computadores a menudo esperan por los datos. Teniendo una máquina de estado en su lugar, los conjuntos parciales y completos de datos NMEA pueden ser analizados. Si solo un conjunto parcial de datos fuese recibido y enviado al analizador, entonces cuando el resto de los datos es recibido, el analizador completará cualquier sentencia NMEA que pueda haber quedado incompleta.

Debajo, en la **figura 1** ilustra el diagrama de flujo del analizador NMEA. Nuestro diseño asumirá dos capas abstractas de software, el analizador del protocolo NMEA y el analizador específico de las sentencias NMEA. No importa el número de bytes de datos recibidos o si hay solamente un mensaje parcial. El analizador manejará los datos y extractará individualmente las sentencias NMEA.

Cada vez que el computador recibe datos, el los enviará al analizador NMEA. Luego en este documento se encontrará una explicación de cómo ocurre cuando una sentencia es recibida.

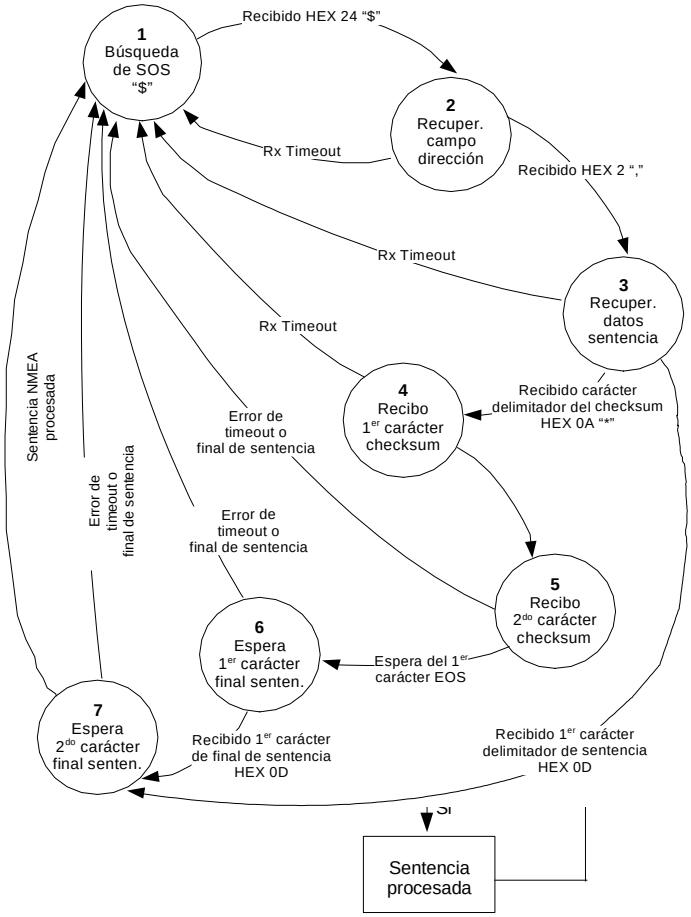


Figura 1

La **figura 2** ilustra la máquina de estado del analizador del protocolo NMEA. Esto ilustra cada estado y su transición. Note que hay una transición de interrupción en la mayoría de estados.

Figura 2

Puede ser necesario agregar una transición de interrupción para cada uno de los estados para sincronizar la máquina de estado cuando no se recibe datos por un período de tiempo. La máquina de estado del analizador ilustra esto haciendo que una transición para el estado 1 ocurra cuando los datos no han sido recibidos para un período de tiempo. Esta interrupción es una aplicación diferente. Un dispositivo NMEA de 4800 baudios enviará típicamente datos cada 1 ó 2 segundos. Por lo tanto, una interrupción de 3 a 4 segundos sería suficiente.

Definiciones de una máquina de estado:

Búsqueda del comienzo de la sentencia (SOS, siglas en inglés). Este estado buscará el carácter '$' (HEX24) cuando SOS es encontrado, entonces solamente la transición conseguirá el campo de dirección NMEA.

Recuperación del campo de dirección. El analizador recolectará los caracteres hasta que encuentre el carácter ',' (coma HEX 2C). Nosotros dejamos el campo de dirección con una longitud flexible. Esto permitirá analizar cualquier sentencia extraña o propietaria.

Recuperación de los datos de la sentencia. Esto es cuando todos los datos asociados con la dirección NMEA son recolectados para un análisis posterior. Este estado continuará de recolectar los datos y ejecutará un cálculo del checksum antes que reciba un delimitador checksum "*" (HEX 2A) o el final de la sentencia <CR><LF> (HEX 0D 0A).

Recibo del primer carácter ASCII del checksum. Este estado simplemente espera por el primer carácter del checksum.

Recibo del segundo carácter ASCII del checksum. Cuando el segundo carácter del checksum es recibido, deberá haber suficiente información para verificar el checksum con el checksum calculado que fue realizado en el estado 3. Si los checksum coinciden, luego la única tarea por hacer es chequear el carácter de final de sentencia.

Espera del primer carácter de final de sentencia (ST, siglas en inglés). Esta tarea simplemente espera por el primer carácter del final de sentencia.

Espera del segundo carácter de final de sentencia. Cuando el segundo carácter del final de sentencia es recibido, entonces la máquina de estado del analizador llamará la función de proceso NMEA donde la dirección de la sentencia y los datos son usados como parámetros. Esta función buscará la dirección de la sentencia NMEA y llamará la función de la sentencia apropiada.

Función de procesamiento de la sentencia NMEA

La función de procesamiento es el paso final en el análisis de los datos actuales. Esta función simplemente usará la dirección de la sentencia NMEA para llamar la función apropiada de la sentencia para extractar los datos convenientemente.
Ejemplo del código

Hay varias opciones cuando se extracta datos de una sentencia específica. La opción más práctica es la de extraer los datos, se fija una bandera que indica que los datos han cambiado (normalmente un contador) y almacenarlos para usarlos luego. Otro método podría usar una técnica volver a llamar, cuando el usuario pueda especificar una función ellos la llamarían una vez la sentencia sea recibida. Nuestro ejemplo usará el primer método.

Este ejemplo usa el sistema operativo Windows™ para los datos NMEA usando el puerto serial RS232. El medio de desarrollo es Microsoft Visual

C/C++ versión 6.0. la clase parser no esta restringida al medio de desarrollo o al sistema operativo. Esta ha sido usada exitosamente en otros ambientes también.

Configurando un temporizador de Windows en 100 ms, el ejemplo sondeará el buffer de recibo del RS232 y enviará los datos al analizador. Abajo en la **figura 3** ilustra el diagrama de flujo básico que nuestro ejemplo usará.

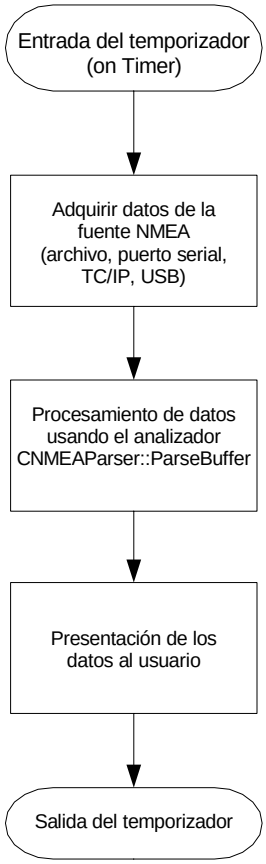


Figura 3

Para simplificar la interfase del puerto serial de Windows una clase envolvente fue creada, la Csio. Esta clase simplifica la inicialización del puerto serial, tiene métodos simples de lectura y envío del buffer.

El demo del analizador NMEA es una aplicación basada en ventanas de dialogo donde el objeto serial (m_Sio) y los objetos del analizador NMEA (m_NMEAParser) son definidos. En el perído de inicialización, un temporizador de 100 ms y uno de 1000 ms son inicializados. El temporizador de 100 ms es usado para guardar el buffer de recibo del puerto serial mientras el temporizador de 1000 ms es usado para actualizar el texto en la ventana de dialogo. En la parte de abajo se presenta un segmento del código fuente para el temporizador.

```
void CNMEAParserDemoDlg::OnTimer (UINT nIDEvent)
{
    switch (nIDEvent)
    {
        // Read data from serial port and send it to the parser
        case TIMER_ID_POLL_COMM :
            BYTE pBuff[256];
            DWORD dwBytesRead;
            //
            //Read serial port, bytes read returned
            //
            dwBytesRead = m_Sio.Read (pBuff, 255);
            //
            // Parse data returned in pBuff
            //
            m_NMEAParser.ParseBuffer (pBuff, dwBytesRead);
        break;
        // Update text in dialog
        case TIMER ID UPDATE TEXT :
            UpdateText ();
        break;
    }
    Cdialog::OnTimer (nIDEvent);
}
```

Cada 100 ms (TIMER_ID_POLLCOMM) el m_Sio.Read(pBuff, 255) leerá más de 255 bytes de datos del buffer del puerto serial y colocará el contenido dentro del arreglo pBuff. El númeor de bytes leidos del buffer del puerto serial es retornado por m_Sio.Read(). Luego el método m_NMEAParser.ParserBuffer() es llamado para analizar el buffer y extraer los datos útiles. Cada 1000 ms el método Update Text () es llamado para actualizar el texto en la ventana de dialogo mostrando los datos específicos de la clase analizador del NMEA (NMEA parser class).

El código fuente completo del analizador NMEA esta ubicado en NMEAParser.cpp y NMEAParser.h. Esta definición de la clase refleja la máquina de estado de un analizador en general, que fue discutido anteriomente en este documento. Los datos específicos de las sentencias

NMEA están almacenados en miembros de clase individual definidos públicamente. Vea el archivo NMEAParser.h para las definiciones de los datos. Además, cada sentencia que es analizada éxitosamente, su correspondiente contador es incrementado. Por ejemplo, si el mensaje GGA es recibido y analizado, entonces el contador GGA m_dwGGACount será incrementado. Esto es de gran utilidad para determinar cuando un comando específico fue recibido.

El código fuente puede ser encontrado en:
http://www.visualgps.net/papers/NMEAParser

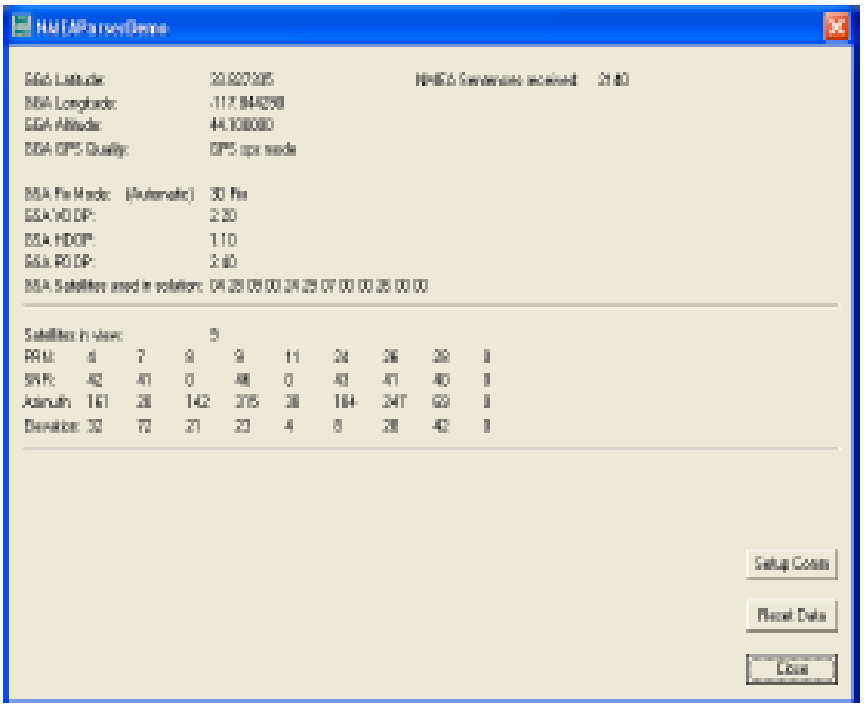


Figura 4

La **figura 4** muestra un pantallazo de la aplicación del Demo del Analizador NMEA. Solamente algunos datos de sentencias GGS, GSV y GSA son mostrados. En la esquina inferior derecha están los controles para resetear todos los datos de los miembros a cero y permitir el control sobre el puerto serial.

Referencias

[1] Asociación Nacional de Electrónicos Maritimos, “Norma NMEA 0183 para interface con dispositivos electrónicos maritimos.2”, versión 3.01, Enero 1, 2002.