Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TREBALL DE FI DE CARRERA

TÍTULO DEL TFC: Utilidades de Matlab para la gestión de los datos crudos obtenidos por un receptor GPS con tecnología SiRF

TITULACIÓN: Ingeniería Técnica de Telecomunicaciones, especialidad Sistemas de Telecomunicaciones

AUTOR: Alberto Fernández Sierra

DIRECTOR: José María González Arbesú

FECHA: 1 de Febrero de 2013

Título: Utilidades de Matlab para la gestión de los datos crudos obtenidos por un receptor GPS con tecnología SiRF

Autor: Alberto Fernández Sierra

Director: José María González Arbesú

Fecha: 1 de Febrero de 2013

Resumen

Mientras que el usuario corriente sólo se sirve de algunos datos básicos para su localización, los receptores de GPS reciben y determinan numerosos otros datos que si bien para el usuario no experimentado son superfluos no lo son para usuarios más avanzados.

A partir de un receptor comercial del fabricante Royaltek, un transmisor de Bluetooth y el software matemático Matlab, se han creado una serie de rutinas para la extracción de datos crudos del receptor y su correcta conversión e interpretación con el propósito de que los usuarios avanzados puedan disponer de ellos a partir de un receptor de bajo coste.

Para ello, en este proyecto encontrará un breve resumen sobre el sistema GPS, explicaciones sobre el funcionamiento o la interpretación de los protocolos SiRF, encontrará también una pequeña guía descriptiva de la transmisión de datos por parte del sistema GPS imprescindible para conseguir casar el protocolo SIRF con las tramas enviadas por el GPS. También se mostrara un experimento donde se prueba el uso de dichas rutinas en un entorno académico y como fundamento de partida sobre el cual trabajar en el futuro y desarrollar o perfeccionar las aplicaciones que lo empleen.

Title: Utilidades de Matlab para la gestión de los datos crudos obtenidos por un receptor GPS con tecnología SiRF

Author: Alberto Fernández Sierra

Director: José María González Arbesú

Date: February, 1st 2013

Overview

The normal users only uses some basic information for their location, the GPS receivers sends many other data that are useful for experts users.

With a commercial receiver from Royaltek, a Bluetooth transmitter and Matlab mathematical software, we've created a series of functions for extracting raw data from the receiver and for check the correct interpretation and conversion. The objective is the users could get this data as from low-cost receiver.

In this this project will find a brief summary of the GPS system, explanations on the operation or structure of the SiRF protocols, you will also find a small descriptive guide to the transmission of data from the GPS system, that's is important to get interoperability between SIRF protocol with the GPS frames. Also provides an experiment to show the use of such routines in an academic and as a foundation on which to base future work and develop or improve the applications that employ.

AGRADECIMIENTOS

Quisiera agradecer a todas las personas que con su apoyo y ayuda han hecho posible este proyecto: Jordi Clapés, Josep María de Batlle Surroca, David Fernández Sierra, Cristina Ojeda, Rocío Muñoz, Sílvia Míguez, Josep Roura, Jose Miguel Fernández Alonso, Dolores Sierra Rivera, Javier Portero, Alessandra Spee Parvis, José María González Arbesú, Marian Villalonga, José Alguacil Linde, Sílvia Cufí, Marc López, Silvia Llorens, Vanessa Mínguez-Olsson, Guillermo Rebollo, Laura Sitjá Xivillé, Beatriz Tejeiro, a las grandes personas que conocí en Milán, a toda mi familia en general. En especial, a Paco, toda la familia González Bártulos, y sobre todo a Marta González Bártulos, la persona más inspiradora en mi vida, cuya actitud ante la vida debería ser ejemplo para todos.

ÍNDEX

INTRODUCCIÓN

En este proyecto se describen los procesos de documentación, recopilación de información y creación de una serie de rutinas para la extracción de datos crudos de un receptor GPS con tecnología SiRF[1] de la compañía RoyalTek[2].

Con dichos datos se pretende ofrecer la oportunidad a los futuros estudiantes de asignaturas que traten los Sistemas de Posicionamiento de que puedan trabajar directamente con los datos que no son visibles al usuario corriente y que puedan sacar conclusiones a partir de experimentos. Además, la extracción de dichos datos proporciona una base sobre la cual poder desarrollar futuras aplicaciones.

El receptor de GPS usado es concretamente el modelo llamado a nivel comercial BT GPS X-Mini, aunque el modelo concreto es RBT-2001-1123[3] y que puede encontrarse en la página web del fabricante.
Como características principales cabe resaltar el chip SIRF STAR III GPS, que soporta 20 canales y protocolos como NMEA-0183[4]. Otras características interesantes:

- Gran sensibilidad de adquisición de datos.
- Batería recargable de más de 8 horas de duración en pleno funcionamiento.
- Soporta sistemas WAAS/EGNOS/MSAS[5]
- Posibilidad de usarlo tanto en dispositivos móviles (Smartphones, PDA) y ordenadores.

Puesto que el receptor se comunica mediante Bluetooth, hemos utilizado el receptor estándar F8T103[6] de la casa Belkin.

Debido a la cantidad de información existente en la red, la cantidad de foros de soporte con gran actividad y la cantidad de opciones que ofrece Matlab[7] para trabajar, hemos decidido crear las rutinas en esta plataforma de software matemático. En Matlab se puede programar en un lenguaje prácticamente igual al C.

En las pruebas previas a la creación de las rutinas se ha usado diferente software -siempre en versiones gratuitas- como: Docklight[8], Advanced Serial Port Manager[9] o SIRF Demo[10].
DockLight y Serial Port Manager son programas que interactúan con el puerto serie permitiendo entre otras cosas, enviar datos en diferentes formatos (Binario, Hexadecimal, decimal, ASCII) y escanear la transferencia de datos salientes y entrantes a través de él.
SIRF Demo es un software nativo de SIRF que se comunica con el receptor de GPS y ofrece datos como los que hemos extraído con las rutinas. Se ha usado

con la finalidad de contrastar los valores obtenidos con las rutinas diseñadas.

En el capítulo uno se describirá de forma breve el sistema GPS, sus características principales y las tareas de modernización. En el segundo, la estructura del mensaje del sistema GPS centrándonos en la propia estructura de las tramas y subtramas. El tercer capítulo trata sobre el protocolo SIRF donde se proporcionará información básica de dicho protocolo, cómo es el transporte del mensaje, el código de detección de errores (*Checksum*) y los seis mensajes de salida con los que se ha trabajado en este proyecto.
El cuarto capítulo trata de algunos puntos importantes que se repiten en todas las funciones, para facilitar así la comprensión del código que se encuentra en los anexos.
En el quinto capítulo es una presentación de resultados, donde se mostraran una serie de graficas sobre la evolución de las pseudodistancias y las distancias Doppler a lo largo de una hora y a lo largo de seis horas.

Finalmente se hayan las conclusiones, las referencias y el anexo.

1. GPS – Descripción

1.1. Breve historia

En el año 1957 la ex URSS envió al espacio el satélite Sputnik I. Éste era controlado observando el efecto Doppler de la señal transmitida.

A raíz de esto, e inmersos en plena guerra fría, se entendió que conocer la posición y sincronizar los relojes de diferente equipamiento militar alrededor del mundo, suponía una ventaja táctica en combate ya que permitía conocer en todo momento la localización de las propias tropas y recursos militares, coordinar ataques o trazar mejores rutas.

Rápidamente la Armada norteamericana utilizó este sistema para dar y conocer la exacta localización de sus flotas mientras navegaban. Así nacía TRANSIT[11] en 1960 y era probado entre 1964 y 1967. Sólo cinco satélites orbitando la Tierra permitían a los buques determinar su posición en el mar una vez cada 40 minutos además de que el observador debía permanecer prácticamente estático para poder obtener los datos útiles. Luego, en 1967 vino Timation[12] que, aprovechando el desarrollo de los relojes atómicos de alta precisión, demostró que podían funcionar en el espacio.

A partir de entonces, la Armada y la Fuerza Aérea combinaron sus programas en lo que se conoció como Programa de Tecnología de Navegación (Navigation Technology Program). Luego pasó a llamarse NAVSTAR GPS[13].

A partir de ese momento, el GPS se desarrolló rápidamente para fines militares con un total de 11 satélites “Block I” lanzados entre 1978 y 1985.

Sin embargo, fue el derribo de un avión de pasajeros coreano (vuelo 007)[14] por parte de la URSS en 1983 lo que llevó al Gobierno de Ronald Reagan en EE.UU. a establecer el GPS para aplicaciones civiles de modo que los aviones, las embarcaciones y medios de transporte de todo el mundo pudieran determinar su posición y evitar desviarse involuntariamente y entrar en límites territoriales extranjeros.

Más tarde, el desastre del transbordador Challenger en 1986 retrasó la actualización del sistema GPS hasta que en 1989 se lanzaron los primeros satélites Block II. En el verano de 1993, EEUU lanzó su 24º satélite Navstar a la órbita, lo que completó el sistema.

El Bloque II ha sufrido numerosas actualizaciones de satélites y es éste el que usamos actualmente.

El futuro del sistema GPS vendrá determinado por nuevas señales y la puesta en órbita del nuevo bloque –aun en desarrollo- Bloque III.

1.2. Características

El Sistema de Posicionamiento Global (SPG o comúnmente GPS), es un servicio propiedad de los EE.UU. que proporciona a los usuarios información sobre localización, velocidad y tiempo.

Consiste en 32 satélites que se encuentran a una altitud media de 20200 Km., tienen un periodo de 11 horas y 58 minutos (12 horas sidéreas) y una inclinación de 55º respecto al ecuador celeste. Las frecuencias portadoras de la señal civil y militar son respectivamente 1575’42 Mhz (señal L1) y 1227’60 Mhz (señal L2). Normalmente se necesitan cuatro satélites para conocer la posición y altitud exactas, pero si hay más satélites visibles los receptores pueden calcular mejor dicha posición.

La localización se realiza mediante una técnica llamada “trilateración” [15] y que usa como mínimo tres satélites. Cada uno de ellos representa un punto en la esfera terrestre con centro en sí mismos. Con otro satélite más ya se tienen dos esferas que intersecan en una circunferencia[Figura 1], lo cual produce una incertidumbre sobre cuál es, de toda esa circunferencia, el punto correcto.

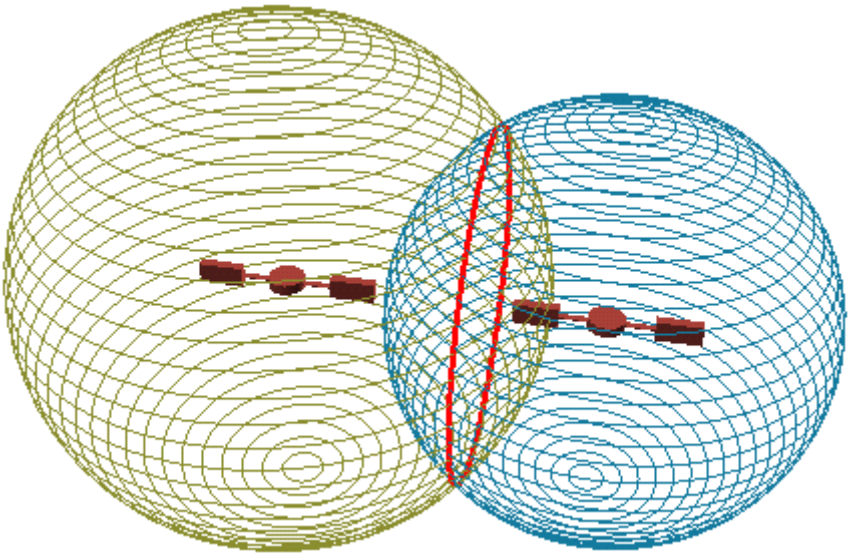

Figura 1. Dos esferas centradas en dos satélites.

El tercer satélite soluciona esta incertidumbre creando otra esfera centrada en si mismo que marca dos puntos. Uno de ellos es el correcto porqué está por encima del plano que forman los satélites y queda fuera del área de cobertura de los satélites[Figura 2].

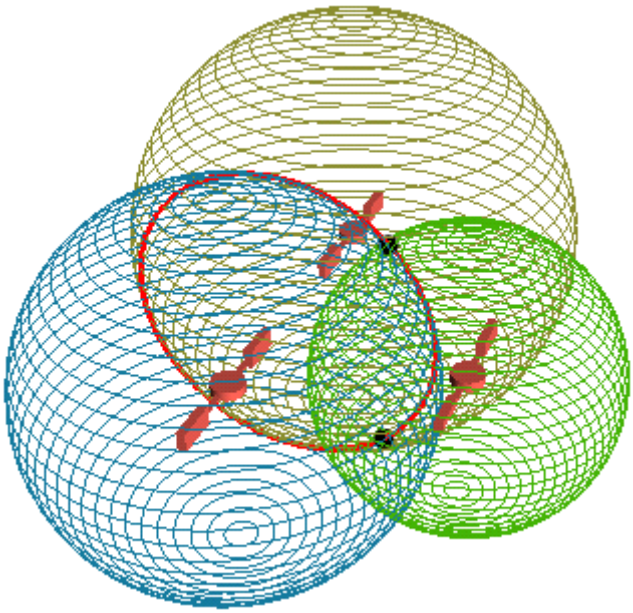

Figura 2. La intersección de tres esferas delimita la zona donde se encuentra el receptor (Punto A).

Este sistema está constituido por tres segmentos: el segmento espacial, el segmento de control y el segmento del usuario.

1.2.1. Segmento espacial

El segmento espacial lo forman 32 satélites, de los cuales como mínimo 24 de ellos están operativos el 95% del tiempo repartidos en 6 planos orbitales con 4 satélites por plano, una altura de la órbita de 20200 Km, 12 horas de periodo por satélite y 55º de inclinación. Aunque se han hecho pruebas con hasta 31 satélites, además de contar con entre 3 y 4 satélites que pueden ser reactivados en cualquier momento y si fuese necesario.

Los satélites no fueron todos lanzados a la vez, si no que en órbita coexisten diferentes generaciones de satélites más o menos avanzados, que trabajan al unísono. Han sido lanzados en diferentes generaciones que se llaman Bloques [16]. El Bloque I lo forman los satélites del 1 hasta el 11. Fueron puestos en órbita entre el 22 de Febrero del 1978 y el 9 de Octubre de 1985. Se les esperaba una vida media de 5 años.
El Bloque II, con una vida media de 7’3 años, fue puesto en órbita entre el 14 de Febrero de 1989 y el 4 de Octubre de 2012. El Bloque II tuvo varias actualizaciones cuyos nombres son: Bloque IIA, Bloque IIR, Bloque IIR-M y Bloque IIF. En total, en el Bloque II se pusieron en órbita 60 satélites: del 13 al 73
El pasado 4 de Octubre, la Fuerza Aérea Norteamericana lanzo el tercer satélite GPS IIF, que empezó a estar operativo el 13 de Noviembre del 2012.

El Bloque III –satélites del 74 en adelante- es la nueva generación de satélites. Ahora mismo se encuentran en desarrollo por la empresa Lockheed Martin. El Bloque III ofrecerá señales más potentes, una mayor fiabilidad, precisión e integridad. Además de una vida media de 15 años. Según los contratos actuales, la financiación alcanza para cuatro satélites, pero con opción de aumentar hasta ocho. Otra de las mejoras es que la intensidad de las señales permitiría recibir los datos sobre posicionamiento en zonas ligeramente cubiertas, tales como edificios bajos, aparcamientos o las calles rodeadas de grandes rascacielos.
Aparte de esto los nuevos GPS incluirán otras señales que servirían para nuevas aplicaciones, tanto militares como civiles. Las señales para usos militares, por ejemplo, serían más seguras, precisas y difíciles de interceptar.
Esta actualización del sistema GPS no será barata: el coste total se ha calculado en más de 4.000 millones de euros, que habrán de incluirse en los presupuestos a partir de 2014.
El primer satélite del bloque será lanzado en alguna fecha aún no determinada en el año 2014. Se espera que todo el bloque esté totalmente operativo hacia finales del 2030.

El estado de la constelación puede consultarse en el servidor del U.S. (Naval Observatory) [16]. La última vez que observamos el servidor fue el 26 de Enero de 2012 con los datos que se muestran a continuación[Figura 3]:

```
A.  BLOCK II/IIA/IIR/IIR-M INDIVIDUAL SATELLITE STATUS

SVN  PRN
23   32  Launched 26 NOV 1990; usable 26 FEB 2008; operating on Rb std
           Unusable 04 Dec 1746 UT until further notice
             (NANU 2012077)
26   26  Launched 07 JUL 1992; usable 23 JUL 1992; operating on Rb std
           Unusable 22 Jan 1621 UT to 1727 UT due to
             maintenance (NANUs 2013006, 2013007/22 JAN)
30   30  Launched 12 SEP 1996; usable 01 OCT 1996; operating on Cs std
33   03  Launched 28 MAR 1996; usable 09 APR 1996; operating on Cs std
34   04  Launched 26 OCT 1993; usable 22 NOV 1993; operating on Rb std
35   30  Launched 30 AUG 1993; usable 28 SEP 1993; operating on Rb std
           Unusable 17 Dec 2144 UT until further notice
             (NANU 2012083)
36   06  Launched 10 MAR 1994; usable 28 MAR 1994; operating on Rb std
38   08  Launched 06 NOV 1997; usable 18 DEC 1997; operating on Cs std
39   09  Launched 26 JUN 1993; usable 21 JUL 1993; operating on Rb std
           Scheduled unusable 15 Nov 1330 UT to 16 Nov 1330 UT for
             repositioning maintenance (NANU 2012068)
40   10  Launched 16 JUL 1996; usable 15 AUG 1996; operating on Cs std
41   14  Launched 10 NOV 2000; usable 10 DEC 2000; operating on Rb std
43   13  Launched 23 JUL 1997; usable 31 JAN 1998; operating on Rb std
44   28  Launched 16 JUL 2000; usable 17 AUG 2000; operating on Rb std
45   21  Launched 31 MAR 2003; usable 12 APR 2003; operating on Rb std
46   11  Launched 07 OCT 1999; usable 03 JAN 2000; operating on Rb std
47   22  Launched 21 DEC 2003; usable 12 JAN 2004; operating on Rb std
48   07  Launched 15 MAR 2008; usable 24 MAR 2008; operating on Rb std
49   27  Launched 24 MAR 2009; unusable until further notice
50   05  Launched 17 AUG 2009; usable 27 AUG 2009; operating on Rb std
51   20  Launched 11 MAY 2000; usable 01 JUN 2000; operating on Rb std
52   31  Launched 25 SEP 2006; usable 12 OCT 2006; operating on Rb std
           Unusable 23 Jan 1537 UT to 2242 UT due to
             maintenance (NANUs 2013005, 2013008/23 JAN)
```

Figura 3. Archivo gpstd.txt extraído del servidor U.S. Naval Observatory con el estado actual del sistema.

```
53   17  Launched 26 SEP 2005; usable 16 DEC 2005; operating on Rb std
           Scheduled unusable 30 Jan 2315 UT to 31 Jan 1115 UT for
             repositioning maintenance (NANU 2013009)
54   18  Launched 30 JAN 2001; usable 15 FEB 2001; operating on Rb std
55   15  Launched 17 OCT 2007; usable 31 OCT 2007; operating on Rb std
56   16  Launched 29 JAN 2003; usable 18 FEB 2003; operating on Rb std
57   29  Launched 20 DEC 2007; usable 02 JAN 2008; operating on Rb std
58   12  Launched 17 NOV 2006; usable 13 DEC 2006; operating on Rb std
59   19  Launched 20 MAR 2004; usable 05 APR 2004; operating on Rb std
60   23  Launched 23 JUN 2004; usable 09 JUL 2004; operating on Rb std
61   02  Launched 06 NOV 2004; usable 22 NOV 2004; operating on Rb std
62   25  Launched 28 MAY 2010; usable 27 AUG 2010; operating on Rb std
63   01  Launched 16 JUL 2011; usable 14 OCT 2011; operating on Rb std
65   24  Launched 04 OCT 2012;
           Set usable 14 Nov 0033 UT
             (NANU 2012070/14 NOV)
For information concerning:   BLOCK II       see file . . . . . GPSB2
                              GPS SYSTEM     see file . . . . . GPSSY
                              TIME TRANSFER  see file . . . . . GPSTT

--
File gpstd.txt
last updated
Sat Jan 26 11:00:05 UTC 2013
                               TIME TRANSFER  see file . . . . . GPSTT
```

Figura 3. Archivo gpstd.txt extraído del servidor U.S. Naval Observatory con el estado actual del sistema.

1.2.2. Segmento de Control

El segmento de control GPS consiste en una red global de instalaciones de tierra que realizan un seguimiento de los satélites, controlan sus transmisiones, llevan a cabo análisis y envían comandos y datos a la constelación para mantener actualizados parámetros como las efemérides, el almanaque, la deriva calculada para cada satélite, etc. En definitiva, el segmento de control utiliza mediciones recogidas por las estaciones de seguimiento para predecir el comportamiento de la órbita de cada satélite y se asegura que estas, estén dentro de unos límites aceptables.

El segmento actual de control operativo incluye una estación de control principal, una estación de control maestra suplente, 12 estaciones de mando y control de antenas y 16 estaciones de seguimiento[18]. La ubicación de estas instalaciones se muestra a continuación:

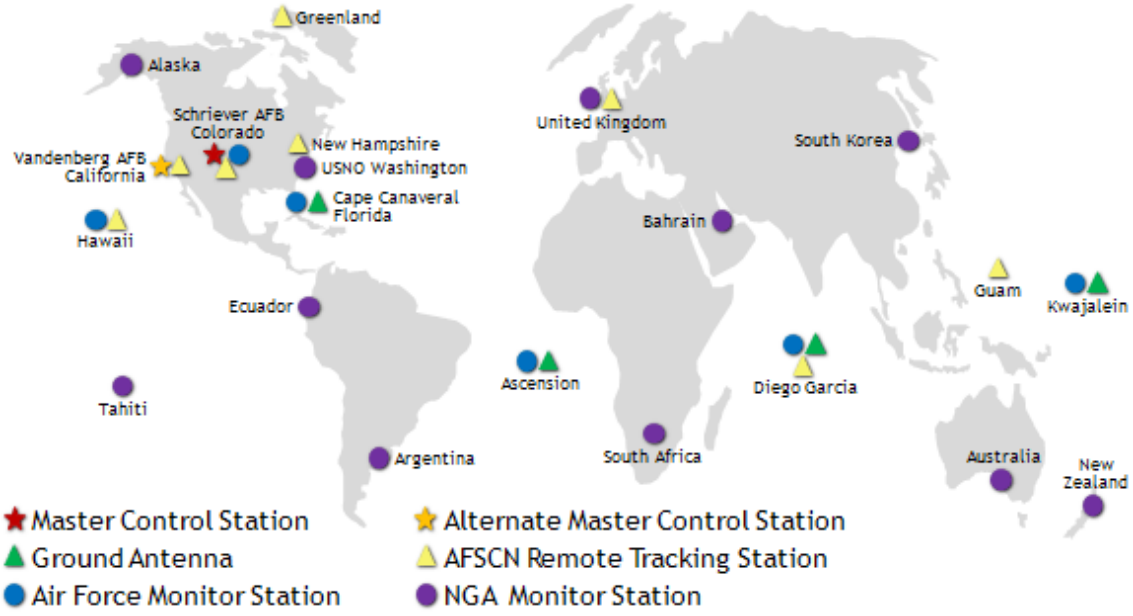

Figura 4. Posición de las estaciones de control, antenas y estaciones de la Fuerza Aerea.

De entre todas las estaciones, la más importante probablemente sea la estación de control principal en Colorado, donde se realizan funciones de control primarias del segmento, proporcionando el mando y control de la constelación GPS, se generan mensajes de navegación, se asegura la salud y la precisión de la constelación de satélites. Recibe información de navegación de los satélites provenientes de las estaciones de monitoreo, utiliza esta información para calcular las localizaciones precisas de los satélites en el espacio, y entonces genera sus órbitas. En caso de anomalía, puede modificar ligeramente la posición de los satélites para mantener óptima la constelación.

1.2.3. Segmento de Usuario

El segmento de usuario es la parte más conocida del GPS. El servicio que ofrece el sistema GPS no tiene costes directos. No obstante, el usuario debe hacerse con un receptor adecuado. El sistema es ininterrumpido y fiable, lo cual ha permitido a los usuarios de todo el mundo desarrollar cientos de aplicaciones que afectan a casi todas las facetas de la vida moderna. Cada día se inventan nuevos usos para el GPS, cuya única limitación es la creatividad de la imaginación humana.

Está formado por los receptores GPS. Sus funciones principales son las de recibir las señales emitidas por los satélites, decodificar los mensajes de navegación, medir el retardo de la señal (desde el transmisor hasta el receptor) a partir de los cuales calculan la localización y presentar la información de la posición

en la que se encuentra (ya sea en 2 dimensiones o en 3 dimensiones). Tienen una precisión entorno de los 15 metros aunque con los nuevos sistemas como EGNOS en Europa, esta cifra se ha reducido hasta a los 1’5 metros[19]. Permiten la comunicación con el usuario, pudiendo este añadir mapas de fondo, almacenar datos (rutas, posiciones), posiciones de inicio, etc.

Por poner algunos ejemplos, el sistema GPS tiene aplicaciones civiles en agricultura, aviación, carreteras y autopistas, topografía y cartografía, espacio, ocio, navegación marítima, cronometría, medio ambiente o seguridad y salvamento.

1.3. Modernización del sistema

Como casi cualquier tecnología, el GPS debe mantenerse en permanente revisión e innovación en aras de una continua modernización[20].
Un caso conocido y ya implementado es el Sistema GPS Diferencial (DGPS).
El DGPS es un sistema de potenciación del GPS de alta precisión, desarrollado por el Jet Propulsion Laboratory (JPL), para dar respuesta a la necesidad de determinar, en tiempo real, la localización, el tiempo más precisos y la órbita de las misiones científicas y espaciales de la Administración Nacional de Aeronáutica y del Espacio (NASA) de los Estados Unidos.
Entre los planes de futuro se incluyen la utilización del Sistema de Rastreo por Satélite y Retransmisión de Información (TDRSS) para diseminar, vía satélite y en tiempo real, los mensajes de corrección diferencial a fin de permitir determinar la posición de los satélites con mayor precisión.

Aunque en concreto, el programa de modernización del GPS implica una serie de adquisiciones consecutivas de satélites (antes se ha comentado el caso del Bloque GPS III),también incluye mejoras en el segmento de control o la implantación y uso de nuevas señales civiles. Sobre esto último se va a hablar en adelante y hasta el final del capítulo.

Uno de los principales objetivos del programa de modernización del sistema GPS es la implementación de nuevas señales de navegación a la constelación de satélites. Se les llama L1C, L2C y L5 [21]. Estas señales no serán compatibles con los receptores actuales. Sin embargo, estos receptores podrán seguir siendo utilizados. L1C se seguirá usando, al ser heredada del actual sistema. No obstante, el cambio es gradual ya que la Fuerza Aérea empieza a usar estas nuevas señales de forma progresiva y remplazando a las viejas. Muchas de estas señales están actualmente en experimentación y no se abrirán al público hasta que, según la web oficial del sistema GPS, se puedan emitir desde por lo menos 18 de los 24 satélites.

1.3.1. L1C: Primera señal civil

La señal L1C trabaja a 1575’42 MHz. Está diseñada conjuntamente entre los Estados Unidos y Europa para permitir la interoperabilidad entre el GPS y otros sistemas internacionales de navegación por satélite como el sistema europeo Galileo[22].
L1C cuenta con un *Multiplex Binary Offset Carrier* que permite la cooperación internacional a la vez que protege los intereses estadounidenses de seguridad nacional.
Otros proveedores de servicios de navegación por satélite están adoptando L1C como futuro estándar para interoperabilidad internacional. Por ejemplo, el sistema japonés Quasi-Zenith, el Sistema Regional de Navegación por Satélite de la India (IRNSS en inglés) o el sistema Compass chino.

1.3.2. L2C: Segunda señal civil

La señal L2C trabaja a 1227’60 Mhz. Cuando se combina con L1C/A en un receptor de frecuencia doble, L2C permite la corrección ionosférica, lo cual aumenta la precisión. De esta forma, los civiles que puedan usar este sistema dual, podrán disfrutar de la misma precisión que los militares o incluso mejor. Además también ofrece tiempos más rápidos de adquisición de la señal, mayor fiabilidad y margen dinámico de operación. Es una señal más potente que la actual, lo cual permite que se pueda recibir el señal desde debajo de árboles o desde interiores.
Según específica la web oficial del sistema GPS , web dependiente del Departamento de Defensa de los EEUU y el Gobierno de los Estados Unidos, el Departamento de Comercio calcula que el L2C podría generar 5800 millones de beneficios hasta el año 2030 aunque no especifica cómo.

1.3.3. L5: Tercera señal civil

La tercera señal, L5, trabaja a 1176’45 Mhz. Está diseñada para satisfacer los exigentes requisitos de seguridad en el transporte y aviación. Promete pequeñas o ninguna interferencia bajo cualquier circunstancia.
L5 se emite en una banda de radio reservada exclusivamente para los servicios de seguridad de la aviación. Cuenta con la mayor potencia y ancho de banda de todo el servicio proporcionando una ganancia de 10 veces menos tiempo de procesamiento de códigos de ensanchamiento..
Además de mejorar la seguridad, el uso de L5 incrementara la capacidad y eficiencia del combustible dentro del espacio aéreo de los EEUU, los ferrocarriles, las vías de navegación y las carreteras permitiendo trazar mejores rutas y que los vehículos se desvíen lo mínimo posible de estas.
Más allá del transporte, L5 proporcionará a los usuarios de todo el mundo la señal civil GPS más avanzada hasta la fecha. Cuando se utiliza en combinación con L1C/a, L2C, L5 proporcionará un servicio muy robusto.

Mediante una técnica llamada *trilaning*, el uso de estas tres frecuencias permitirá una precisión en el posicionamiento inferior al metro sin necesidad de sistemas aumentados.

2. ESTRUCTURA DEL MENSAJE GPS

2.1. Estructura del mensaje

Los satélites del sistema GPS envían regularmente mensajes que contienen información relativa a parámetros orbitales, cálculos relativistas, tiempos de sincronización e información del estado del sistema. El receptor se sirve de estos mensajes para calcular la posición, velocidad y tiempo. Esta información viene estructurada en tramas.
La estructura de las tramas enviadas por los satélites que conforman las constelaciones del sistema GPS viene descrita en el documento ICD-GPS-200[23].
Este documento recibe actualizaciones cada cierto tiempo por lo que es recomendable consultarlo periódicamente.
A pesar de que contiene una cantidad ingente de información respecto a tiempos de respuesta, descripción de los parámetros, vectores o circuitería digital, este apartado se centra exclusivamente en las cinco subtramas donde se encuentra confinada la información de interés.

Cinco subtramas conforman una trama (o frame) de mensajes. El mensaje se transmite con un bit rate de 50 bps. La longitud de cada subtrama es de 300 bits con lo que la longitud total de una página es de 1500 bits. La duración de cada página es de 6 segundos. La longitud total del mensaje es 37500 bits (25 páginas) con una duración de transmisión de 750 segundos[24].

En cada subtrama viene entrelazada la información de los almanaques y las efemérides[Figura 5], además de otros datos, tal y como se explica a continuación:

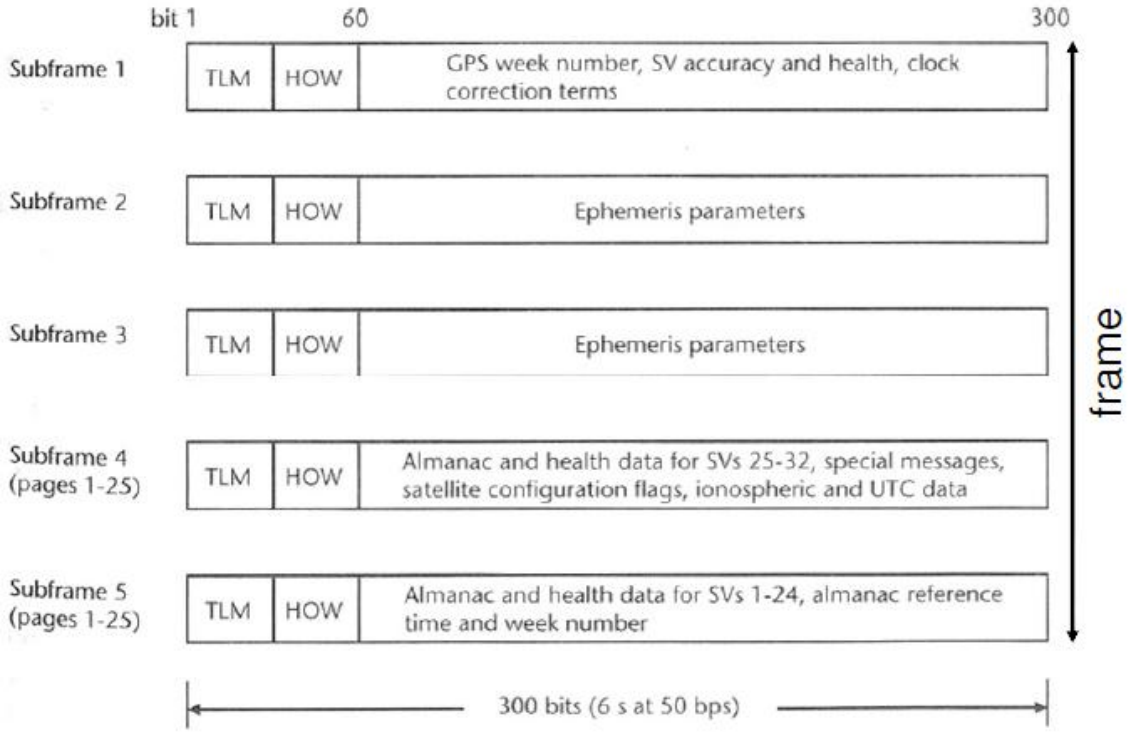

Figura 5. Resumen gráfico de la estructura del mensaje del sistema GPS.

Como se puede observar, en las subtramas 2-3 y en las 4-5 se encuentran los datos que conforman respectivamente las efemérides y el almanaque.

Las efemérides son la información de los parámetros orbitales keplerianos enviada por los satélites, dando la posición precisa de los mismos.
Los elementos keplerianos son datos que sirven para fijar la órbita y por consiguiente, determinar la posición del satélite en el espacio en un momento dado. Atendiendo puramente a la mecánica orbital y las leyes de Kepler, seis son los datos que se necesitan para definir la órbita de un satélite, los tres primeros (Inclinación, Ascensión Recta del Nodo Ascendente y Argumento del Perigeo) definen la orbita kepleriana en un espacio 3D y de los tres restantes (Excentricidad, Movimiento Medio, Anomalía Media) los dos primeros definen la forma y el tamaño de la órbita.
Esta información cambia frecuentemente, siendo actualizada por las estaciones de seguimiento de la Tierra. Los parámetros orbitales de los satélites se van actualizando a medida que su movimiento se ve alterado por la atracción del Sol y la Luna, la diferencia de gravedad entre distintas zonas de la corteza terrestre, viento solar, etc. Un periodo de cambio típico es de 4 horas.

El almanaque es información enviada también por los satélites de la constelación, informando sobre ellos mismos y el resto de satélites miembros del sistema, su nivel de salud, etc. Esta información suele variar teóricamente una vez a la semana, aunque como comentaremos en el siguiente capítulo esto no es exactamente así. El almanaque recibe diariamente una actualización a pequeña escala.

En las imágenes que siguen se pueden observar una por una las subtramas y las partes que las conforman.

2.2. Subtramas

Como se puede comprobar, los diferentes datos se envían entrelazados dentro de lo que se llaman *words* o palabras. Cada word es de 30 bits.

El inicio de cada subtrama contiene 60 bits que conforman la telemetría (TLM) y el *hand-over word* (HOW). La telemetría ocupa 30 bits y es la primera palabra de cada subtrama. La telemetría contiene información relacionada con los documentos ICD-GPS-203, ICD-GPS-224, ICD-GPS-225 y SS/CS interface, así como información relacionada con la edad de los datos de las efemérides y datos útiles para usuarios autorizados y el segmento de control.
El *hand-over word* ocupa también 30 bits y contiene, además de otros datos, información relacionada con el tiempo dentro de la semana GPS y bits de alerta que indican la calidad de la señal. Ayuda a sincronizar el código P(Y) a partir del código C/A.

Tanto en las subtramas 2 y 3 como en las 4 y 5 se dan casos como el de la variable A_0 –entre otras- que podemos encontrar en la subtrama 4, página/trama 18 donde se observa que esta variable ocupa 32 bits de los cuales los primeros 24 bits se encuentran en la word 7, luego vienen 8 bits de paridad y los 8 bits restantes de A_0 están en la siguiente word, la word 8.
Este dato es importante tenerlo en cuenta para posteriormente presentar los datos correctamente.

Es conveniente explicar que las efemérides y el almanaque comparten los valores de e, $(A)^{1/2}$, $(OMEGA)_0$, C_{is}, i_0 y ω. Es decir, acuden al mismo número de bits de las subtramas correspondientes para adquirir estos valores.

2.2.1. Subtrama 1

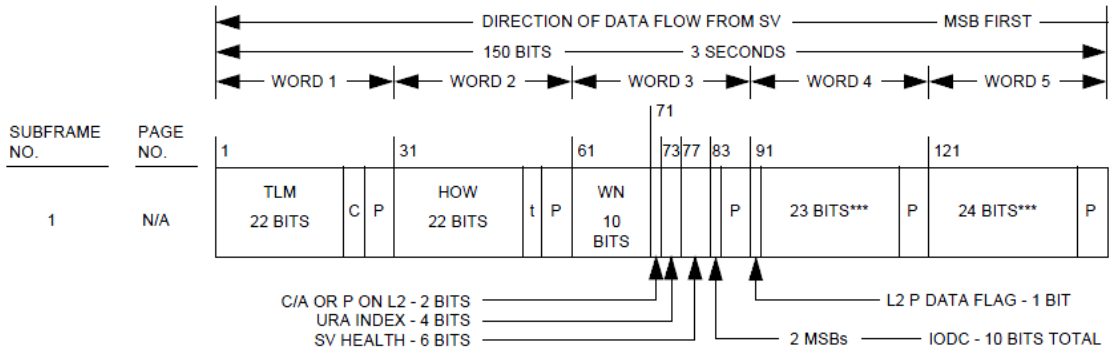

Figura 6. Subtrama 1

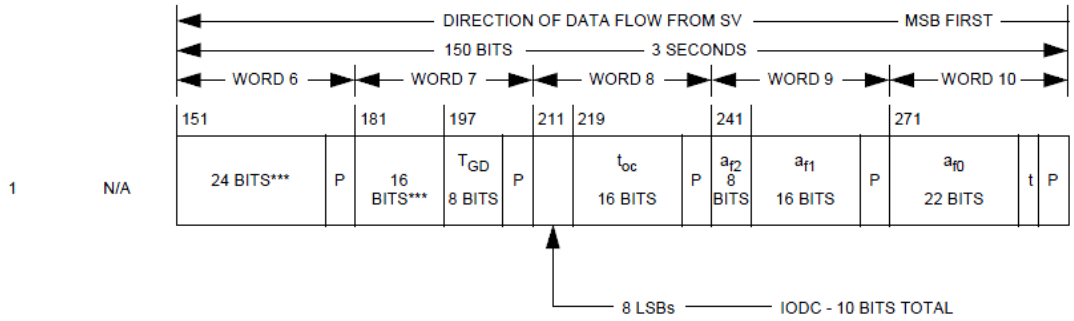

Figura 6. Subtrama 1

Esta trama contiene las variables siguientes:

- WN (Week Number)– Número de la semana desde el 5 de Enero del 1980
- IODC (Issue Of Data, Clock): identifican los datos actuales del mensaje de navegación, cambian con el cambio del mensaje de navegación cada 7 días.
- a_{f0}, a_{f1}, a_{f2}, t_{oc} – Correcciones del reloj del satélite y tiempo en el que son válidos.
- SV HEALTH – Indica la salud del satélite.
- URA (User Range Accurary) – Estima del error en la medida de la pseudodistancia.
- T_{GD} – Diferencial de Retardo de Grupo Estimado. Permite estimar a los receptores monofrecuencia el retardo introducido por la ionosfera.
- Data Flag – indica si el mensaje de navegación modula al código P en L2.

Los *** informan de que esos bits están reservados.
La P son 6 bits de paridad.
La C indica que los bits 23 y 24 de la telemetría (TLM) están reservados.
t son dos bits sin información.

2.2.2. Subtramas 2 y 3

Las subtramas 2 y 3 son importantes para nuestros objetivos ya que en ellas se encuentran los datos de las efemérides.

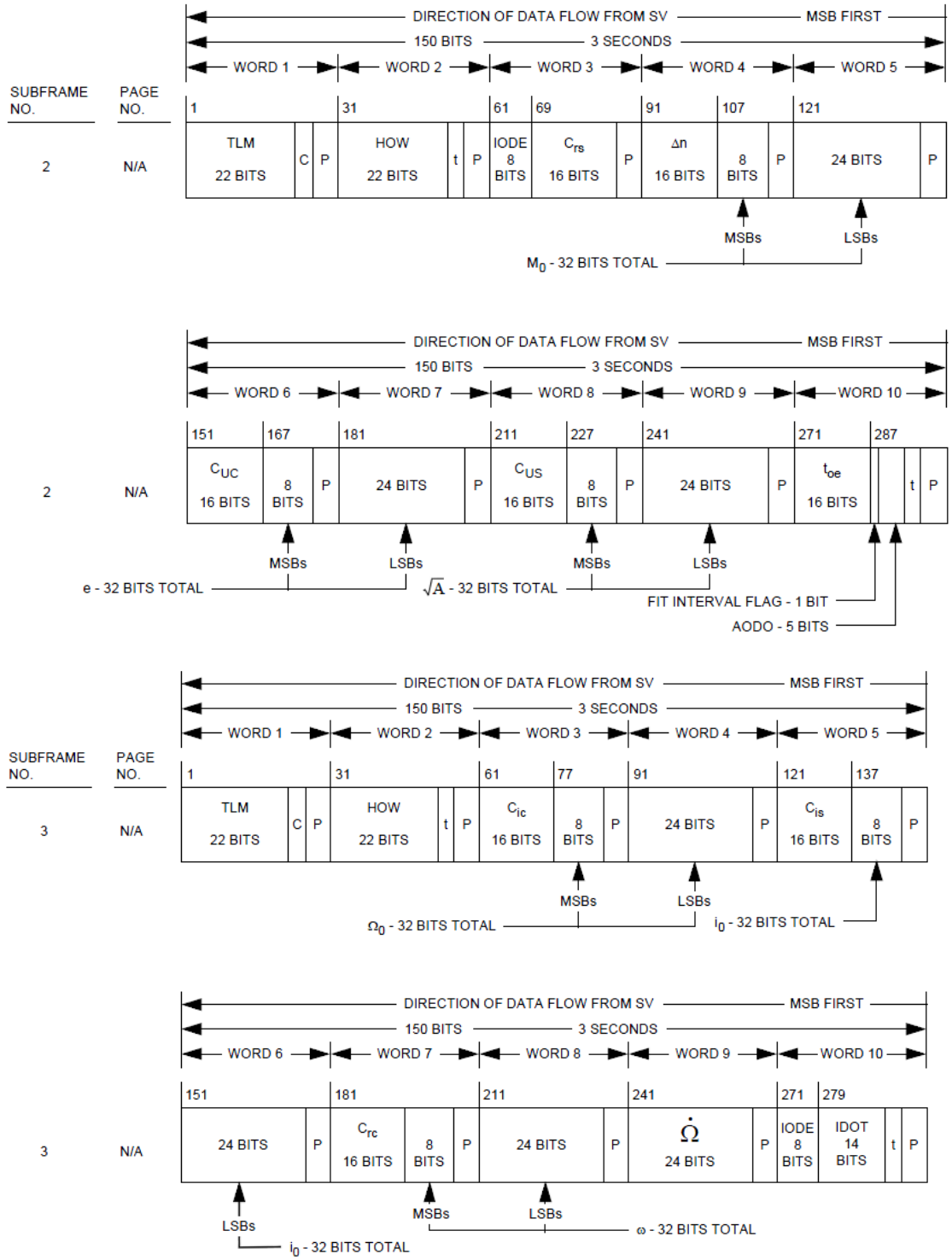

Figura 7. Subtramas 2 y 3

Esta trama contiene las variables siguientes:

- IODE (Issue Of Data, Ephemeris) - 8 bits que deben coincidir con los últimos 8 bits del IODC del mismo mensaje. Se repite en las subtramas 2 y 3. Permite detectar cambios en el mensaje de navegación.

- Fit Interval Flag - indica si el ajuste de los elementos orbitales corresponde a un intervalo de 4 horas o a un intervalo mayor.

- AODO (Age Of Data Offset) – 5 bits que indican la edad de los datos del mensaje de navegación para aplicar la tabla de corrección del mensaje de navegación (Navigation Message Correction Table (NMCT).

- M_0 – Anomalía media

- A_n – Diferencia media de movimiento a partir de un valor calculado

- e – excentricidad

- $(A)^{1/2}$.- Raíz cuadrada del semieje mayor

- $(OMEGA)_0$ – Longitud del nodo ascendente

- i_0 – Ángulo de inclinación

- ω – argumento del perigeo

- OMEGADOT – Cambio de ascensión recta

- IDOT – Cambio de ángulo de inclinación

- C_{uc},C_{us}, C_{rc},C_{rs},C_{ic},C_{is} – Diferentes amplitudes de términos corregidos referidos a los harmónicos del seno y el coseno del argumento del perigeo, el radio de la órbita y el ángulo de inclinación.

- T_{oe} - Tiempo de referencia de las efemérides

La P son 6 bits de paridad.
La C indica que los bits 23 y 24 de la telemetría (TLM) están reservados.
t son dos bits sin información.

2.2.3. Subtramas 4 y 5

Como en el apartado anterior, en este caso, las subtramas 4 y 5 contienen los datos del almanaque
.

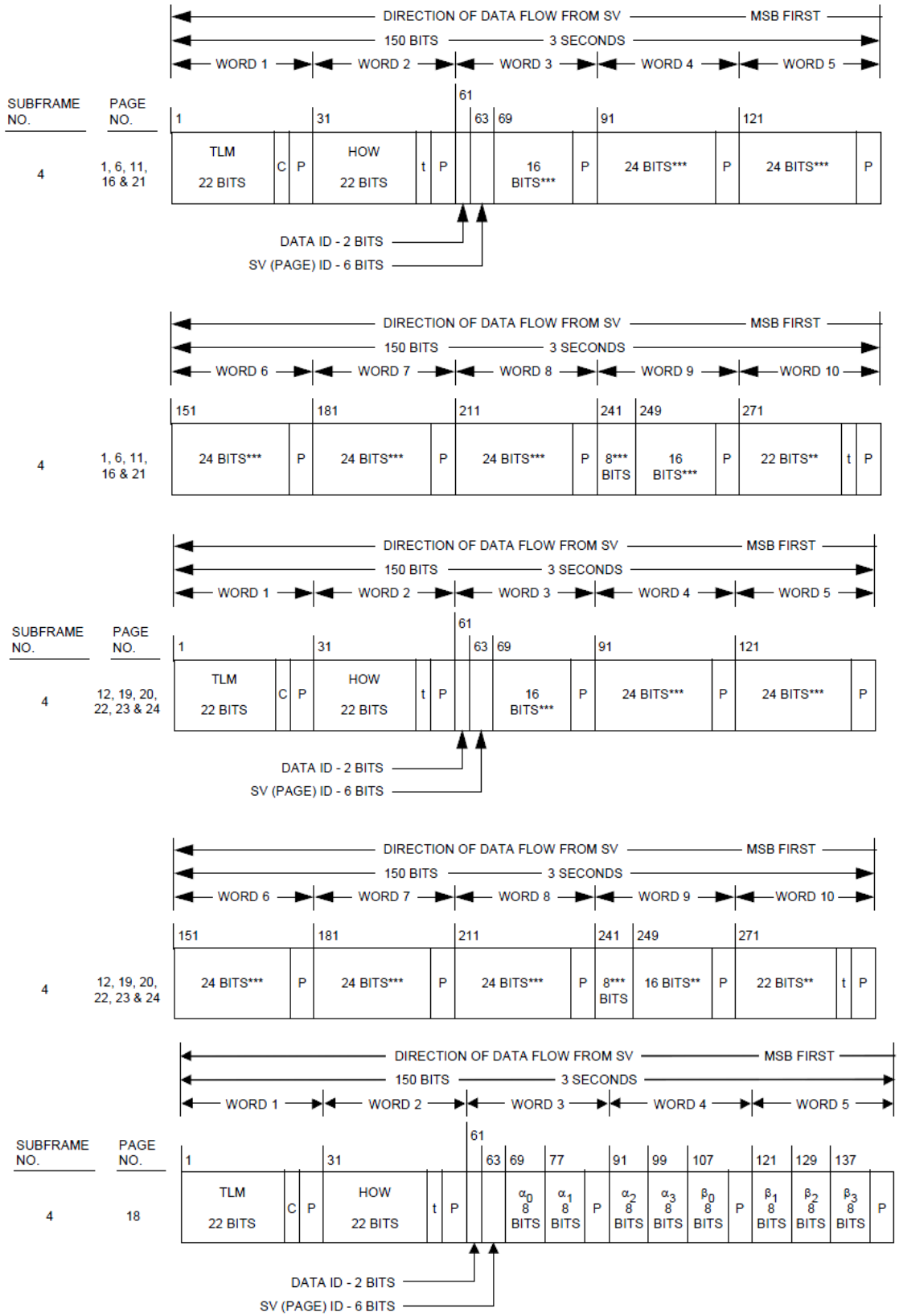

Figura 8. Subtramas 4 y 5.

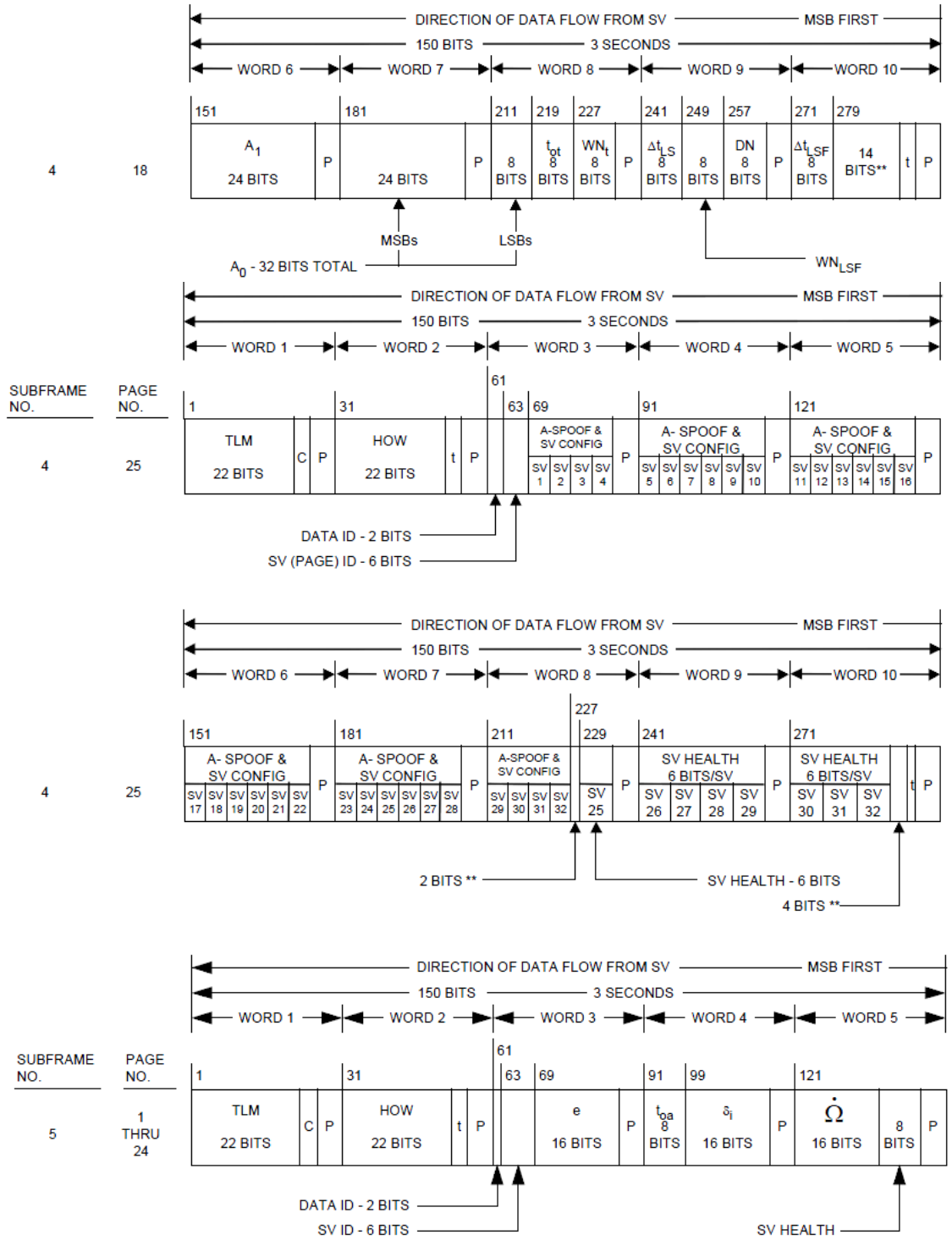

Figura 8. Subtramas 4 y 5.

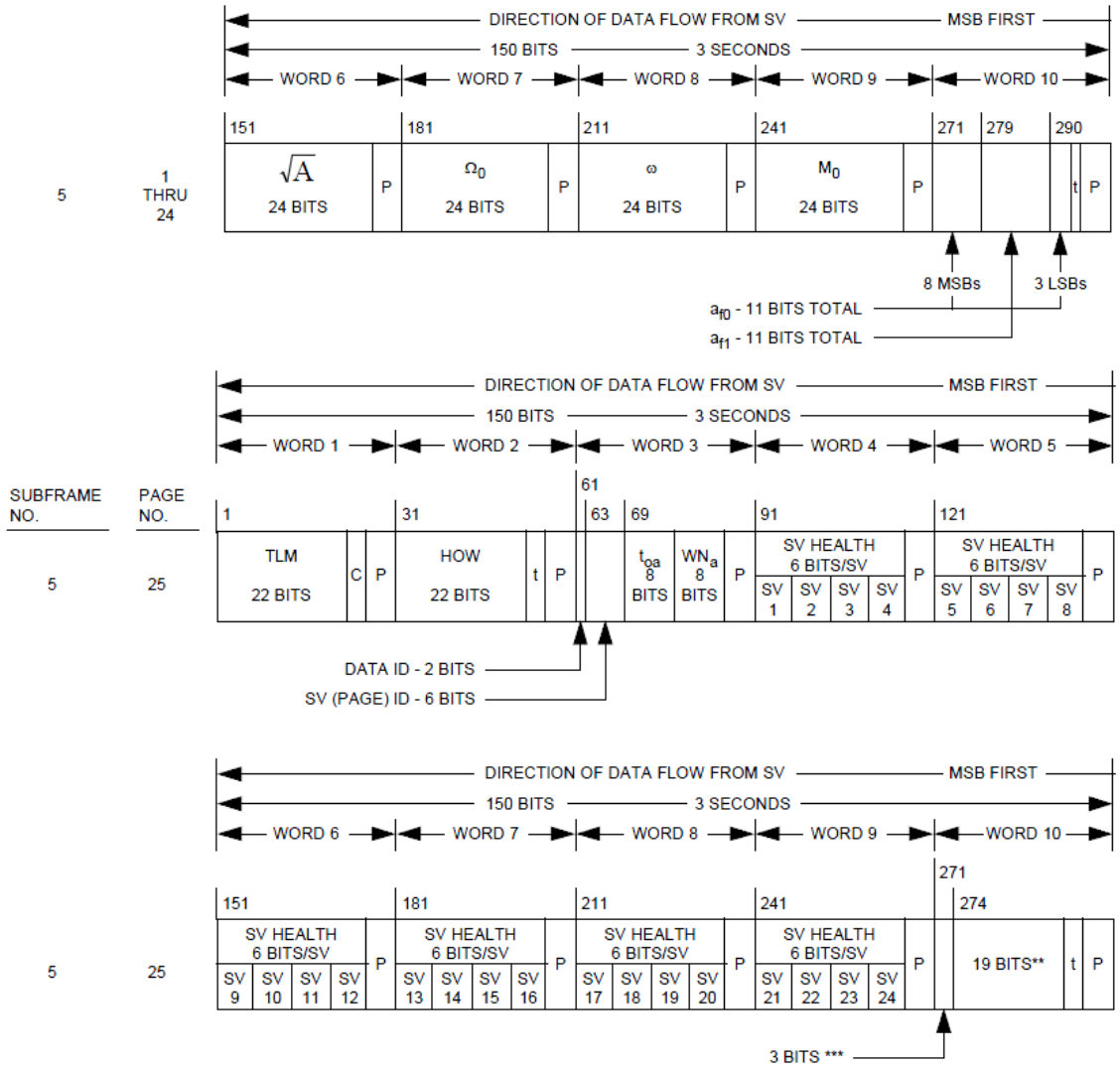

Figura 8. Subtramas 4 y 5.

Esta trama contiene, entre otras variables, las que conforman el almanaque:

- E – excentricidad
- Toa – Tiempo de referencia del almanaque
- δi – Inclinación orbital
- El resto de las variables que conforman el almanaque se toman de los los apartados anteriores. Estas variables son: OMEGADOT, $(A)^{1/2}$, $(OMEGA)_0$, M_0, af_0, af_1

Los ** indican que son bits reservados por el sistema.
Los *** informan de que esos bits están reservados.
La P son 6 bits de paridad.
La C indica que los bits 23 y 24 de la telemetría (TLM) están reservados.
t son dos bits sin información.

3. PROTOCOLO SiRF

3.1. Información sobre el protocolo SiRF

El protocolo SIRF es aquel que se usa en los GPS con chipset SiRF. Creado originalmente por SiRF Technology, Inc. en 1996, esta empresa fue posteriormente comprada por CSR PLC[25] en 2009.
Se trata de un protocolo que ofrece más información que el protocolo estándar llamado NMEA –National Marine Electronics Association-.

Entre las ventajas que ofrece SiRF frente al NMEA es que no se trata sólo de un protocolo de salida sino que admite parametrización para hacer peticiones al sistema GPS.
Este protocolo presenta una transferencia de datos entre el GPS y el ordenador –u otro dispositivo- más fluida, tiempos menores de adquisición de datos, mayor rapidez a la hora de calcular la posición y más precisión, además de permitir a los distintos programas un manejo más completo del receptor GPS; ya que admite el envío de comandos que permiten, por ejemplo, activar o desactivar el sistema WAAS/EGNOS, inicializar el receptor, ajustar una serie de parámetros del receptor, etc.

La velocidad típica a la que suele operar el protocolo SiRF es de 57600 baudios 8N1 (8 bits de datos, sin bits de paridad, 1 bit de parada), aunque se puede configurar a cualquier otra velocidad. La transmisión de datos se realiza en formato binario.

Otra característica de los receptores con este protocolo es que en cada segundo se actualiza la señal de los satélites individualmente. Esto puede ser útil para hacer un control preciso de la señal recibida de cada satélite en todo momento.

Para poder modificar los valores mediante el protocolo SiRF se pueden usar, por ejemplo:

- SiRF Demo – Un programa que distribuye la propia empresa que ha creado el protocolo y que permite manipular multitud de variables, activar o desactivar funciones y parámetros, visualizar localizaciones y capturar las transmisiones que recibe el receptor.

- SiRFTech[26] – Software que permite modificar prácticamente todas las características de un receptor con protocolo SiRF y evaluar dichos cambios. Se usa en agendas de bolsillo (PDAs).

- Mediante algunas de las rutinas creadas en este proyecto que proporcionan la capacidad de que cualquiera con MATLAB puede hacer una lectura automática de los parámetros.

Para información adicional es aconsejable consultar los siguientes documentos:

- NMEA Reference Manual – Para saber más de este protocolo de comunicación básico en el sistema GPS .
- ICD-GPS-200 – Entre otras cosas, para entender mejor las transmisiones del sistema GPS.

El documento SiRF Protocol Reference Manual se estructura en:

- Tablas
- Prefacio
- Reglas del protocolo
- Mensajes de Entrada
- Mensajes de Salida
- Información adicional

En este proyecto se centra en cinco mensajes concretos de salida –o respuestas- así como los mensajes para hacer las peticiones. Los mensajes elegidos nos permiten capturar datos que son vitales en el funcionamiento del sistema tales como el almanaque o las efemérides. También algunos datos de navegación. Aunque no se ha creado una función para ello, el mensaje del estado del reloj (Clock Status) nos proporcionaría entre otros datos la deriva del reloj (Clock Drift)[28]. Los relojes del sistema GPS se ven afectados por la gravedad lo que provoca que los relojes a bordo de los satélites parezcan ir más rápidos que los terrestres. Estos datos nos permitirían realizar correcciones relativistas en por ejemplo, la frecuencia de la portadora (Carrier Frequency)... Además, se capturarán las pseudodistancias, la hora del sistema, etc. Estos mensajes son los siguientes:

- Mensaje ID 2 – Measured Navigation Data
- Mensaje ID 14 – Alamanac Data
- Mensaje ID 15 – Ephemeris Data
- Mensaje ID 19 – Navigation Parameters
- Mensaje ID 29 – Navigation Library Measurement Data

Hay que tener en cuenta que el protocolo SIRF elimina todos los bits de paridad que aparecen en las subtramas comentadas en el apartado anterior.

A continuación pasaremos a describir algunos puntos concretos usados en las rutinas y que se sirven del protocolo SiRF. Para ello comentaremos tanto el código como tablas informativas del documento [SiRF Binary Protocol Reference Manual] clave para entender el protocolo y para la realización de este trabajo.

3.2. Transporte del mensaje

El mensaje viene encapsulado de la siguiente forma:

Start Sequence	Payload Length	Payload	Message Checksum	End Sequence
0xA0[1], 0xA2	Two-bytes (15-bits)	Up to 2^{10} -1 (<1023)	Two-bytes (15-bits)	0xB0, 0xB3

1. Characters preceded by "0x" denotes a hexadecimal value. 0xA0 equals 160.

Figura 9. Estructura del mensaje procesado por el protocolo SiRF

Como la totalidad de los mensajes empiezan –A0A2- y finalizan –B0B3- igual, es relativamente sencillo identificar los mensajes. La experiencia en el transcurso de este proyecto indica que es bastante improbable que estos dos caracteres que forman la secuencia de inicio se den durante el resto del mensaje. Asimismo sucede con los dos caracteres finales. Pero por si se diese el caso, el mensaje de Checksum es la suma de comprobación que nos dice si el mensaje es erróneo o no en su longitud

El Payload Lenght indica la longitud del mensaje útil y se transmite primero el byte de peso más alto para seguidamente enviar el byte de peso más bajo.

High Byte	Low Byte
< 0x7F	Any value

Figura 10. Payload Lenght.

3.3. Checksum

El Checksum se transmite igual que el Payload Lenght: primero el byte de peso más alto seguido del byte de peso más bajo.

High Byte	Low Byte
< 0x7F	Any value

Figura 11. Figura 10. Payload Lenght.

El siguiente código es el que comprueba que la longitud del mensaje es la correcta:

```
Index = first
checkSum = 0
while index < msgLen
        checkSum = checkSum + message[index]

checkSum = checkSum AND (2^15-1).
```

3.4. Mensajes de salida

A continuación se comentan algunos puntos importantes de las tablas que cabe tener en cuenta. Para más información y para observar las tablas de cada mensaje, se puede consultar el documento anexo del proyecto.

En cuanto a los valores capturados, su conversión puede ser directa a decimal, excepto algunos casos determinados que se pueden comprobar en los anexos, Estos deben convertir a decimal suponiendo que el valor de origen recibido por el receptor esta en Complemento a 2 o en IEEE754[27]. Algunos ejemplos aclaratorios:

- El valor del argumento del perigeo (ω) recibido en el almanaque se debe convertir usando el Complemento a 2

- El valor de las pseudodistancias (*Pseudorange* en inglés) recibido en el mensaje ID28 que se especifica más adelante, debe convertirse usando el estándar IEEE754

3.4.1. Measure Navigation Data Out – Message ID 2

El mensaje ID2 se envía periódicamente cada segundo y nos muestra información sobre coordenadas geográficas en el sistema cartesiano basadas en el Datum GSW84 que es el actual del sistema GPS, velocidad, el tiempo del sistema GPS o la semana actual desde que se inició el servicio.

La longitud total de este mensaje es de 49 bytes. Quitando los 4 bytes de la cabecera –A0A20029- así como los 4 bytes del cierre del mensaje (2 bytes de

checksum seguidos de B0B3 de final del mensaje) – 09BBB0B3-, nos quedan 41 bytes útiles de información relacionados con la navegación.

Cuando este mensaje llega al receptor, visto en hexadecimal, tiene el formato siguiente:

A0A20029—Start Sequence and Payload Length

02FFD6F78CFFBE536E003AC00400000030001040A00036B039780E3
0612190E160F04000000000000—Payload

09BBB0B3—Message Checksum and End Sequence

Figura 11. Ejemplo de mensaje ID2

Es decir, cada Measure Navigation Data Out se puede reconocer por que la cabecera del mensaje contiene A0A20029 y es importante notar el 02 inicial del Payload. Debido a que podría darse el caso de que surgiese un valor por azar que contuviese A0A20029, lo importante es fijarse en A0A2002902. De esta forma nos aseguramos que el mensaje siguiente contiene los datos.

		Binary (Hex)			ASCII (Decimal)	
Name	Bytes	Scale	Example	Unit	Scale	Example
Message ID	1 U		02			2
X-position	4 S		FFD6F78C	m		-2689140
Y-position	4 S		FFBE536E	m		-4304018
Z-position	4 S		003AC004	m		3850244
X-velocity	2 S	*8	0000	m/sec	Vx÷8	0
Y-velocity	2 S	*8	0003	m/sec	Vy÷8	0.375
Z-velocity	2 S	*8	0001	m/sec	Vz÷8	0.125
Mode 1	1 D		04	Bitmap[1]		4
HDOP[2]	1 U	*5	0A		³5	2.0
Mode 2	1 D		00	Bitmap[3]		0
GPS Week[4]	2 U		036B			875
GPS TOW	4 U	*100	039780E3	sec	÷100	602605.79
SVs in Fix	1 U		06			6
CH 1 PRN[5]	1 U		12			18
CH 2 PRN[5]	1 U		19			25
CH 3 PRN[5]	1 U		0E			14
CH 4 PRN[5]	1 U		16			22
CH 5 PRN[5]	1 U		0F			15
CH 6 PRN[5]	1 U		04			4
CH 7 PRN[5]	1 U		00			0
CH 8 PRN[5]	1 U		00			0
CH 9 PRN[5]	1 U		00			0
CH 10 PRN[5]	1 U		00			0
CH 11 PRN[5]	1 U		00			0
CH 12 PRN[5]	1 U		00			0

Figura 12. Tabla de variables del mensaje ID2

Este mensaje se recibe periódicamente sin hacer petición alguna al GPS.

3.4.2. Almanac Data – Message ID 14

Este mensaje responde a la petición Poll Almanac - Message ID 146, que es una solicitud de envió del almanaque del sistema:

A0A20002—Start Sequence and Payload Length

9200—Payload

0092B0B3—Message Checksum and End Sequence

Figura 14. Ejemplo de mensaje ID14

La longitud total del mensaje que devuelve el receptor es de 30 bytes más bytes de cabecera y 4 bytes de cierre del encapsulado. Es decir, la información útil con los datos del almanaque se encuentran en los 30 bytes.

La tabla de la estructura del mensaje SiRF es la siguiente:

Name	Bytes	Description
Message ID	1	Hex 0x0E (decimal 14)
SV ID	1	SV PRN code, hex 0x01..0x02, decimal 1..32
Almanac Week & Status	2	10-bit week number in 10 MSBs, status in 6 LSBs (1 = good; 0 = bad)
Data[1]	24	UINT16[12] array with sub-frame data.
Checksum	2	

Payload length: 30 bytes

Figura 15. Tabla de variables del mensaje ID14

El mensaje consiste en un array de 12 palabras de 16 bytes cada una. Como comentábamos al inicio de este capítulo, el protocolo SIRF elimina los bits de paridad con lo que la información que ocupa los 30 bytes contiene todas las variables referentes a la información relativa a un satélite del almanaque sin huecos entre ellas. Es decir, todos los datos vienen seguidos, sin espacios entre ellos.

Las primeras 2 palabras de 16 bits cada una, contienen por este orden la telemetría (TLM) y el *Handover Word* (HOW). Las siguientes palabras están ordenadas según la tabla adjunta.

Navigation Message		Data Array		Navigation Message		Data Array	
Word	Byte	Word	Byte	Word	Byte	Word	Byte
3	MSB	[0]	LSB	7	MSB	[6]	MSB
3	Middle	[0]	MSB	7	Middle	[6]	LSB
3	LSB	[1]	LSB	7	LSB	[7]	MSB
4	MSB	[1]	MSB	8	MSB	[7]	LSB
4	Middle	[2]	LSB	8	Middle	[8]	MSB
4	LSB	[2]	MSB	8	LSB	[8]	LSB
5	MSB	[3]	LSB	9	MSB	[9]	MSB
5	Middle	[3]	MSB	9	Middle	[9]	LSB
5	LSB	[4]	LSB	9	LSB	[10]	MSB
6	MSB	[4]	MSB	10	MSB	[10]	LSB
6	Middle	[5]	LSB	10	Middle	[11]	MSB
6	LSB	[5]	MSB	10	LSB	[11]	LSB

Figura 16. Tabla de ordenación del mensaje ID14

Teóricamente, el almanaque se actualiza una vez a la semana. Sin embargo hemos observado que cada día se actualiza con pequeñas correcciones en prácticamente todos los parámetros. Estas actualizaciones diarias se hacen en base al tiempo de aplicación (*Time of Applicability*).
Lo observamos cuando comparábamos los resultados obtenidos a través de las rutinas con los almanaques descargados de la web NAVSTAR. Veíamos que los datos diferían a partir del octavo decimal. En ocasiones a partir del noveno o el decimo.
Se ha observado que suben al sistema un almanaque diario y si usando la rutina se capturan los datos del almanaque en el instante en que el nuevo ha sido subido al sistema, los valores son iguales o prácticamente iguales (con diferencias a partir del noveno o decimo decimal), pero conforme avanzan las horas y se hacen nuevas capturas, las diferencias son en el octavo o noveno decimal. Unas horas más tarde, en el séptimo u octavo. Luego, vuelven a subir otro almanaque y sucede lo mismo.
Parece que actualizan el almanaque diariamente con correcciones del día anterior.

3.4.3. Epehemeris Data – Message ID 15

Este mensaje responde a la petición Poll Ephemeris – Message ID 147, que es una solicitud de envío de las efemérides de los satélites disponibles:

A0A20003—Start Sequence and Payload Length

930000—Payload

0092B0B3—Message Checksum and End Sequence

Figura 17. Ejemplo de mensaje ID15

La longitud total del mensaje que devuelve el receptor es de 92 bytes más 4 bytes de cabecera y 4 bytes de cierre del encapsulado. Es decir, la información útil con los datos de las efemérides se encuentran en dichos 92 bytes.

La tabla de la estructura del mensaje SiRF es la siguiente:

Name	Bytes	Description
Message ID	1	Hex 0x0F (decimal 14)
SV ID	1	SV PRN code, hex 0x01..0x02, decimal 1..32
Data[1]	90	**UINT16** [3][15] array with sub-frames 1..3 data.

Payload length: 92 bytes

Figura 18. Tabla de variables del ID15

Aquí hemos detectado un nuevo error. El Message ID no es 0E como aparece en el manual si no que es 0F tal y como queda reflejado en la tabla de arriba.

También, como puede observarse en el apartado Data, el contenido viene estructurado en un array de 3 filas y 15 columnas por cada satélite visible. La primera palabra de cada fila de cada array ([0,0] ; [1,0] y [2,0]) contienen la identificación del satélite correspondiente. Como en los casos anteriores, el protocolo SIRF elimina los bits de paridad con lo que los datos vienen todos seguidos y siguiendo el orden de la tabla siguiente:

Navigation Message		Data Array		Navigation Message		Data Array	
Word	Byte	Word	Byte	Word	Byte	Word	Byte
2 (HOW)	MSB	[][1]	LSB	7	MSB	[][9]	MSB
2	Middle	[][2]	MSB	7	Middle	[][9]	LSB
2	LSB	[][2]	LSB	7	LSB	[][10]	MSB
3	MSB	[][3]	MSB	8	MSB	[][10]	LSB
3	Middle	[][3]	LSB	8	Middle	[][11]	MSB
3	LSB	[][4]	MSB	8	LSB	[][11]	LSB
4	MSB	[][4]	LSB	9	MSB	[][12]	MSB
4	Middle	[][5]	MSB	9	Middle	[][12]	LSB
4	LSB	[][5]	LSB	9	LSB	[][13]	MSB
5	MSB	[][6]	MSB	10	MSB	[][13]	LSB
5	Middle	[][6]	LSB	10	Middle	[][14]	MSB
5	LSB	[][7]	MSB	10	LSB	[][14]	LSB
6	MSB	[][7]	LSB				
6	Middle	[][8]	MSB				
6	LSB	[][8]	LSB				

Figura 19. Ejemplo de mensaje ID15

La tabla con las variables de este mensaje se encuentran en los anexos.

3.4.4. Navigation Parameters – Message ID 19

Este mensaje responde a la petición Poll Navigation Parameters – Message ID 152, que es una solicitud de envió de parámetros de navegación.

A0A20002—Start Sequence and Payload Length

9800—Payload

0098B0B3—Message Checksum and End Sequence

Figura 20. Ejemplo de mensaje ID19

La longitud total del mensaje que devuelve el receptor es de 65 bytes más 4 bytes de cabecera y 4 bytes de cierre del encapsulado. Es decir, la información útil con los datos de las efemérides se encuentran en dichos 65 bytes.

Este mensaje contiene algunos parámetros relacionados con la navegación que son establecidos mediante otras instrucciones del manual de SiRF.

		Binary (Hex)			ASCII (Decimal)	
Name	Bytes	Scale	Example	Unit	Scale	Example
Message ID	1 U		13			19
Message Sub ID[1]	1 U		00			
Reserved	3 U		00			
Altitude Hold Mode[2]	1 U		00			
Altitude Hold Source[2]	1 U		00			
Altitude Source Input[2]	2 S		0000	m		
Degraded Mode[2]	1 U		00			
Degraded Timeout[2]	1 U		00	sec		
DR Timeout[2]	1 U		01	sec		
Track Smooth Mode[2]	1 U		1E			
Static Navigation[3]	1 U		0F			
3SV Least Squares[4]	1 U		01			
Reserved	4 U		00000000			
DOP Mask Mode[5]	1 U		04			
Navigation Elevation Mask[6]	2 S		004B			
Navigation Power Mask[7]	1 U		1C			
Reserved	4 U		00000000			
DGPS Source[8]	1 U		02			
DGPS Mode[9]	1 U		00			
DGPS Timeout[9]	1 U		1E	sec		
Reserved	4 U		00000000			
LP Push-to-Fix[10]	1 U		00			
LP On-time[10]	4 S		000003E8			

Figura 21. Tabla de variables del mensaje ID19

LP Interval[10]	4 S		000003E8			
User Tasks Enabled[4]	1 U		00			
User Task Interval[4]	4 S		00000000			
LP Power Cycling Enabled[11]	1 U		00			
LP Max. Acq. Search Time[12]	4 U		00000000	sec		
LP Max. Off Time[12]	4 U		00000000	sec		
APM Enabled/Power Duty Cycle[13,14]	1 U		00			
Number of Fixes[14]	2 U		0000			
Time Between Fixes[14]	2 U		0000	sec		
Horizontal/Vertical Error Max[15]	1 U		00	m		
Response Time Max[14]	1 U		00	sec		
Time/Accu & Time/Duty Cycle Priority[16]	1 U		00			

Payload length: 65 bytes

Figura 21. Tabla de variables del mensaje ID19

Para más información respecto a cada parámetro de navegación de la tabla anterior, invitamos al lector a consultar los anexos de este proyecto donde encontrará los mensajes que corresponden a las referencias que se observan al lado de varios parámetros así como otra tabla informativa.

3.4.5. Navigation Library Measurement Data - Message ID 28

Este parámetro no responde a ninguna petición del usuario si no que es recibido periódicamente por el receptor a base de peticiones que hace este de forma oculta.

La longitud total del mensaje que devuelve el receptor es de 56 bytes más 4 bytes de cabecera y 4 bytes de cierre del encapsulado. Es decir, la información útil con los datos de las efemérides se encuentran en dichos 56 bytes.

Name	Bytes	Binary (Hex)		Units	ASCII (Decimal)	
		Scale	Example		Scale	Example
Message ID	1		1C			28
Channel	1		00			0
Time Tag	4		000660D0	ms		135000
Satellite ID	1		15			20
GPS Software Time	8		41740B0B48353F7D	sec		2.4921113696e+005
Pseudorange	8		7D3F354A0B0B7441	m		2.1016756638e+007
Carrier Frequency	4		89E98246	m/s		1.6756767578e+004
Carrier Phase[1]	8		A4703D4A0B0B7441	m		2.1016756640e+007
Time in Track	2		7530	ms		10600
Sync Flags	1		17			23
C/No 1	1		34	dB-Hz		43
C/No 2	1			dB-Hz		43
C/No 3	1			dB-Hz		43

Figura 22. Tabla de variables del mensaje ID28

C/No 4	1			dB-Hz		43
C/No 5	1			dB-Hz		43
C/No 6	1			dB-Hz		43
C/No 7	1			dB-Hz		43
C/No 8	1			dB-Hz		43
C/No 9	1			dB-Hz		43
C/No 10	1			dB-Hz		43
Delta Range Interval	2		03E801F4	ms		1000
Mean Delta Range Time	2		01F4	ms		500
Extrapolation Time	2		0000	ms		
Phase Error Count	1		00			0
Low Power Count	1		00			0

Payload length: 56 bytes

Figura 22. Tabla de variables del mensaje ID28

En el manual aparecen otras tablas con información de este mensaje. Invitamos al lector a consultar los anexos para saber más.

No obstante, para mayor comprensión, es importante matizar algunos aspectos.
En el manual del protocolo se especifica que los bits recibidos deben reordenarse según varias condiciones:

- La versión del software de nuestro receptor. Nuestra versión es la GSW 3.1.1.El software GSW 2.2.0 y las versiones anteriores, y el software GSW 3.x usan las mismas reglas de ordenación. El resto de software utiliza otro (especificado en la tabla correspondiente en el anexo).

- El GSW 3.x no devuelven valores para el *Carrier Phase*.

- Los valores *GPS Software Time*, *Pseudorange*, *Carrier Frequency* y *Carrier Phase* utilizan el formato IEEE-754.

- Para convertir los datos de forma correcta, hay que hacer lo siguiente:

    - Para doble precisión (8 bytes), asumir que los bits son transmitidos en orden B0,B1,…B7. Para nuestra versión del software hay que reordenarlos de la siguiente manera: B3,B2,B1,B0,B7,B6,B5,B4
    - Para 4 bytes hay que asumir que los bits se han transmitido tal como B0,B1,B2,B3 y se reordenan como B3,B2,B1,B0.

- Para ajustar el *Carrier Frequency* (CFr), *GPS Software Time* y *Pseudorange* hay que usar el valor de *Clock Drift* que se obtiene con en el mensaje ID7, de la siguiente forma:

  CFr (m/s) = Reported CFr (m/s) – Clock Drift (Hz) / 1575420000 Hz

4. RUTINAS DE MATLAB

Matlab es un software matemático desarrollado por The Mathworks y nace como resultado de tratar de satisfacer las necesidades computacionales de científicos, ingenieros y matemáticos.
Integra computación, visualización y programación en un entorno fácil de usar.
Tiene un lenguaje de programación propio (lenguaje M) muy similar al C. Entre sus prestaciones básicas se hallan: la manipulación de matrices (unidad básica de Matlab), representación de datos y funciones, implementación de algoritmos, comunicación con otros dispositivos y lenguajes y creación de interfaces gráficas.

Desde hace varios años constituye una herramienta básica en universidades y centros de investigación y desarrollo.

4.1. Generalidades sobre las rutinas

Hemos creado las siguientes rutinas:

- ephemerides.m – Petición al receptor los datos que conforman las efemérides
- PollAlmanac.m – Petición al receptor de los datos del almanaque.
- MeasureNavDatOut.m – Devuelve posición, velocidad y tiempo
- NLMeasurementData.m – Datos de distinta índole como pseudodistancias, distancias Doppler, tiempo del GPS, etc.
- PollNavigation – Petición al receptor de datos actuales de navegación.
- bin2ieee754.m – Rutina que permite la conversión de binario al estándar ieee754. Para ello utiliza la rutina “h2b.m para un cálculo intermedio:
    - h2b.m – Rutina encontrada en el foro de Matlab que pasa de Hexadecimal a Binario.

La idea principal consiste en realizar una petición del dato determinado, acto seguido captar una gran cantidad de datos, cerrar la comunicación y procesar los datos recibidos en busca de los que hemos pedido.
Existen algunos mensajes que se envían periódicamente como Measure Navigation Data Out y Navigation Library Measurement Data, así que en estos casos simplemente hemos capturado una cantidad de datos suficientemente grande como para que en ellos se encuentren los mensajes anteriores.

4.2. Inicialización del puerto serie y inputs / ouputs.

Todas las rutinas inician el puerto serie mediante los parámetros de entrada de la función correspondiente. Comúnmente es así: [variables de salida = nombre-funcion (elport,velocidad,longitud_buffer).

```
s = serial(elport);      %crear objeto serial
set(s, 'BaudRate', velocidad)
set(s, 'DataBits', 8)
set(s, 'InputBufferSize', longitud_buffer);
set(s, 'Parity','none')
set(s, 'StopBits', 1)
set(s, 'Terminator', 'LF/CR')
set(s, 'FlowControl', 'none')
```

Esta es una configuración típica del puerto serie. El valor del InputBufferSize depende de la rutina. La velocidad máxima que soporta el receptor de GPS es de 57600 bauds por segundo.

El input de cada función es, como se ha comentado en la página anterior, el puerto serie que quiere usarse entrado como string, la velocidad de transmisión y la longitud del buffer. La salida de cada función son las variables que conforman ese mensaje del receptor. A continuación se pueden ver dos ejemplos. El primero es la estructura básica de las funciones. El segundo es una de las funciones desarrolladas:

[variableSalida1, variableSalida2,…variableSalidaN] = nombrefuncion (‘COM5’,57600,7000)

[Altitude, Degraded, Masks, DGPS, LP, UserTasks, Other] =pollNavigation(‘COM5’,57600,7000)

Esto es así para, por ejemplo, tener 2 receptores de GPS y elegir con cual queremos comunicarnos.
El output de cada función es una tabla con las variables capturadas.

4.3. Envío de instrucciones / peticiones

Salvo que el mensaje que captemos se envíe con regularidad, como es el caso de Measure Navigation Data Out y Navigation Library Measurement Data, en el resto debemos hacer la petición.

La estructura de las instrucciones es siempre igual:

- Primero abrimos el objeto puerto serie

```
fopen(s);
```

- Creamos una variable con la cadena de valores en hexadecimal ya que es como nos muestra que hay que realizar las peticiones el manual de SiRF. Posteriormente convertimos la variable hexadecimal a valores decimales, puesto que el receptor se comunica así.

```
DataToSend = ['A0';'A2';'00';'02';'98';'00';'00';'98';'B0';'B3'];
DataDec = hex2dec(DataToSend);
```

- Escribimos en “s”, que es el objeto puerto serie la variable “DataDec”, que corresponde a la petición en decimal. Añadimos el formato de los valores.

```
fwrite(s,DataDec,'uint8');
```

- Inmediatamente después de la instrucción anterior empezamos a leer del Puerto serie y a almacenar los datos en una nueva variable llamada “DataReceive” hasta llenar el buffer establecido en la creación del objeto “s” puerto serie. Convertimos los valores de nuevo a hexadecimal para poder contrastar mejor y hallar los valores que nos interesan.

```
DataReceive = fread(s);
DataHex = dec2hex(DataReceive);
```

- Cerramos el puerto serie y lo eliminamos de la memoria.

```
fclose(s);
delete(s);
clear s
```

4.4. Procesamiento de los datos recibidos

Como hemos comentado, el procesamiento de datos se hace a posteriori, una vez ya cerrada la comunicación con el puerto serie.

El manual de SiRF nos indica el encabezado de los mensajes, así como la longitud en bytes de los datos útiles. Luego, dentro de “DataReceive” sólo tenemos que encontrar ese encabezado para localizar el mensaje y mirar si lo que sigue cumple con las condiciones de longitud.

Tomemos como ejemplo las Navigation Library Measurement Data. Según el manual de SiRF, el mensaje de respuesta empieza con A0 A2 00 38 y luego viene el identificador del mensaje que en este caso es 1C. A continuación siguen 56 bytes de datos.
Hemos comprobado después de sucesivas simulaciones y contrastando con SiRF Demo que nunca se dan por azar esta combinación de 5 bytes que son 4 bytes de encabezado y 1 más que identifica el mensaje.

Lo que se ha hecho es entrar en la variable “DataReceive” y buscar el encabezado correspondiente, el identificador de mensaje y la longitud de datos que siguen.
Si se cumple que encontramos los 5 bytes, guardamos los datos que vengan a continuación hasta cumplir la longitud estipulada por el tipo de mensaje. En el caso presentado, después del byte c38 miramos si el que sigue es el 1C. Si es así, guardamos después del byte cuyo valor es 38, los 56 bytes que vengan (ya

que el identificador del mensaje corresponde al primero de los bytes de la longitud de datos útiles).

```
for i=m:5000
        %var = DataReceive(i);
        if DataReceive(i) == hex2dec(('A0'))

            if DataReceive(i+1) == hex2dec(('A2'))

               if DataReceive(i+2) == hex2dec(('00'))

                    if DataReceive(i+3) == hex2dec(('38'))

                        if DataReceive(i+4) == hex2dec(('1C'))

                            valorsTrama(:,j) = DataReceive(i + 4: i + 4
+ 55);
                            checksumTrama(:,j) = DataReceive(i + 4 + 56
: i + 4 + 56+ 1);
                             i=i+m;
                             j=j+1;
                        end
                    end
                end
            end
        end
 end
```

Seguidamente convertimos los valores almacenados en “ValorsTrama” a hexadecimal ya que así se hace más cómodo el estudio de los datos obtenidos.

```
for i=1:56
    for j=1:18
     valorsTramaHex{i,j} = dec2hex(valorsTrama(i,j),2);
    end
end
```

A continuación, el manual de SiRF nos indica en que posiciones se encuentran los datos. Así pues, consistiría únicamente en buscar dentro de las diferentes filas y columnas de la variable “valorsTramaHex” la que corresponda con lo que queramos obtener. Aquí un ejemplo:

```
    Message_ID = dec2hex(valorsTrama(1,:));
    Channel = valorsTrama(2,:);
    Satellite_ID = valorsTrama(7,:);
```

Según el manual y como se puede observar que hace el código, el identificador del mensaje es el primer byte de cada mensaje. El canal, el segundo y la identificación del satélite, el séptimo byte.

Existen casos en que en la obtención del valor real hay que realizar una conversión. En el siguiente caso se debe convertir el valor obtenido a decimal mediante el estándar IEEE 754.

```
for t=1:19
    a = valorsTramaHex(8,:);
    b = valorsTramaHex(9,:);
    c = valorsTramaHex(10,:);
```

```
    d = valorsTramaHex(11,:);
    e = valorsTramaHex(12,:);
    f = valorsTramaHex(13,:);
    g = valorsTramaHex(14,:);
    h = valorsTramaHex(15,:);
    concate = strcat(e,f,g,h,a,b,c,d);
    numero = char(concate(1,t));
    GPS_Software(t) = bin2ieee754 ([numero]);
    clear numero concate a b c d e f g h
end
```

La función “bin2ieee754” convierte valores de hexadecimal al estándar conocido como IEEE754.

```
function salida=bin2ieee754(numhex)
N=length(numhex)*4;
numbin = fliplr(h2b(numhex));
for i=1:N,
    numdec(i)=str2num(numbin(i));
end
switch N
    case 32
        signo=numdec(32);
        exponente=sum(numdec(24:31).*[2.^(0:7)]);
        mantisa=sum(fliplr(numdec(1:23)).*[2.^(-[1:23])]);
        salida=(-1)^signo*2^(exponente-127).*(1+mantisa);
    case 64
        signo=numdec(64);
        exponente=sum(numdec(53:63).*[2.^(0:10)]);
        mantisa=sum(fliplr(numdec(1:52)).*[2.^(-[1:52])]);
        salida=(-1)^signo*2^(exponente-1023).*(1+mantisa);
    otherwise
        salida=0;
end
```

Esta función recibe como parámetro de entrada una variable en hexadecimal, calcula la longitud de dicha variable, convierte dicha variable a binario para posteriormente aplicar la fórmula del estándar IEEE754 según si la variable es de 32 o 64 bits.

En otros casos, como en la rutina “PollAlmanac.m” la conversión de algunos valores se realiza mediante Complemento a 2. El siguiente es un ejemplo que corresponde a la inclinación de la órbita:

```
Orbital = (strcat(valorsTramaBin_9,valorsTramaBin_10));
val = [];
value = [];
Orbital_Inclination = [];

for i=1:k

val = bin2dec(Orbital(i,:));
y = sign(2^(16-1)-val)*(2^(16-1)-abs(2^(16-1)-val));
    if ((y == 0) && (val ~= 0))
        value{i} = -val;
    else
        value{i} = y;
    end
end
dates = (cell2mat(value));
Orbital_Inclination = (dates*2^(-19)+0.30)*pi;
```

Algunas variables deben multiplicarse por un factor de escala. Para el valor de dicho factor, es necesario consultar el anexo.

Como se puede comprobar en el fragmento de código de las líneas anteriores, hay variables que ocupan más de 1 byte. Puede observarse aquí:

```
Orbital = (strcat(valorsTramaBin_9,valorsTramaBin_10));
```

Con lo cual, debemos unir esos dos bytes.

5. VERIFICACIÓN DE RESULTADOS

Adjuntamos capturas de pantalla de los resultados obtenidos con cada función creada con Matlab. Para comprobar que los valores son correctos, en los casos que sea posible se adjuntan capturas de SiRF Demo o en el caso del Almanaque, el que oficial a día 8 de Febrero del 2013 que es cuando se escribió el capítulo.
También hemos realizado otra prueba donde hemos capturado las pseudodistancias (*Pseudorange*) durante 60 minutos, a razón de una captura por minuto.
A continuación hemos realizado otro experimento que consistía en lo mismo que el primero pero en lugar de las pseudodistancias hemos elegido las distancias Doppler (Carrier Frequency).
Finalmente, para comprobar su evolución a lo largo de un periodo suficientemente largo, se ha realizado otra captura de pseudodistancias y Carrier Frequency durante 6 horas a razón de una captura cada 5 minutos.

5.1. Valores de Measure Navigation Data Out

En la siguiente figura se muestran, en una tabla copiada de Matlab, los valores de las variables correspondientes al mensaje del título y que corresponden a lo que se obtiene con la función “MeasureNavDataOut.m”. Con esta función se desea capturar parámetros de posición, velocidad y tiempo. Esta tabla esta superpuesta a los valores obtenidos por SiRF Demo para la misma petición y unos instantes después de la captura de Matlab. De ahí la pequeña diferencia en los parámetros. La correspondencia de cada valor se encuentra en las tablas del protocolo SíRF. Concretamente en la página 24 del apartado 3.4.1 Measure Navigation Data Out.

```
SiRFDemo Version 3.87
log file opened 02/09/13 01:55:35
SiRFDemo Build Date: June 20, 2007
Tx PC Time=1360371335.000
Tx: 0xA0A2000284000084B0B3
2,
```

		Valores Matlab
Coordenadas X	4743263,	4743270
Coordenadas Y	232531,	232535
Coordenadas Z	4243639,	4243636
Velocidad X	0.000,	-0.1250
Velocidad Y	0.000,	-0.1250
Velocidad Z	0.000,	-0.2500
HDOP	2.2,	2.2000
GPS Week	702,	702
GPS TOW	52175600,	521727
SV in Fix	5,	5
Códigos PRN	12,	12
Códigos PRN	25,	25
Códigos PRN	2,	2
Códigos PRN	31,	31
Códigos PRN	29,	29
Códigos PRN	0,	0
Códigos PRN	0,	0
Códigos PRN	0,	0
Códigos PRN	0,	0
Códigos PRN	0,	0
Códigos PRN	0,	0
Códigos PRN	0	0

Figura 23.Valores extraídos de SiRF Demo y en la cuadrícula, valores obtenidos por Matlab un instante antes.

5.2. Valores de Almanaque

Los valores de las variables correspondientes al mensaje del título y que corresponden a lo que se obtiene con la función “pollAlmanac.m”. Con esta función se desea capturar parámetros correspondientes al almanaque del sistema GPS vigente en el momento de la ejecución de la función.

En este caso, la comparación de los valores obtenidos por Matlab (día 8 de Febrero del 2013 a las 9:52 a.m. que es cuando se ha realizado la prueba) la hacemos con el almanaque oficial a fecha 9 de Febrero del 2013 a las 10:00 a.m. cuyo nombre es “almanac.yuma.week0702.589824.txt" y que se puede encontrar en la web oficial.

******** Week 702 almanac for PRN-01 ********		
ID:	01	1
Health:	000	
Eccentricity:	0.1578330994E-002	0.0016
Time of Applicability(s):	589824.0000	589824
Orbital Inclination(rad):	0.9605080624	0.9605
Rate of Right Ascen(r/s):	-0.7714607059E-008	-7.7146e-09
SQRT(A) (m 1/2):	5153.660156	5.1537e+03
Right Ascen at Week(rad):	0.5369644321E+000	0.5370
Argument of Perigee(rad):	0.430371118	0.4304
Mean Anom(rad):	0.8047422004E+000	0.8048
Af0(s):	-0.9536743164E-006	9.5367e-07
Af1(s/s):	0.3637978807E-011	3.6380e-12
week:	702	702

******** Week 702 almanac for PRN-02 ********		
ID:	02	2
Health:	000	
Eccentricity:	0.1200866699E-001	0.0120
Time of Applicability(s):	589824.0000	589824
Orbital Inclination(rad):	0.9390383235	0.9390
Rate of Right Ascen(r/s):	-0.7943188009E-008	-7.9432e-09
SQRT(A) (m 1/2):	5153.555664	5.1536e+03
Right Ascen at Week(rad):	0.5176424910E+000	0.5176
Argument of Perigee(rad):	-2.672961148	-2.6730
Mean Anom(rad):	0.1734007988E+001	1.7340
Af0(s):	0.4177093506E-003	0
Af1(s/s):	0.0000000000E+000	0
week:	702	702

Figura 24.Para los satélites 1 y 2, a la izquierda los valores oficiales del almanaque y a la derecha los correspondientes de Matlab.

5.3. Valores de Efemérides.

Los valores de las variables correspondientes al mensaje del título y que corresponden a lo que se obtiene con la función “pollEpheme.m”. Con esta función se desea capturar parámetros correspondientes a las efemérides del sistema GPS vigentes en el momento de la ejecución de la función.

Los valores de las efemérides no podemos compararlos con SiRF Demo. Sin embargo, los valores concuerdan con lo que se espera, como por ejemplo la variable “sqrt_A” es similar a la del almanaque, que es cómo debe ser. O la excentricidad de la órbita, que en el momento de escribir esto, es para el satélite 25, 0.0021. Si consultamos el almanaque del día 11/2/2013 la excentricidad tiene un valor de $0.2159*10^{-2}$

A_n	1.3945e-09	1.4165e-09	1.6827e-09	1.8716e-09	1.8550e-09	1.4865e-09
C_ic	1.1176e-08	9.4995e-08	-1.6019e-07	-1.3039e-08	6.7055e-08	4.6566e-08
C_is	9.8720e-08	7.8231e-08	-1.0617e-07	2.0862e-07	1.4715e-07	5.9605e-08
C_rc	322.6875	232.3750	244.3750	277.6875	275.8125	318.8438
C_rs	70.1563	-64.1563	-65.4375	60	60.8125	70.7813
C_uc	3.6508e-06	-3.2354e-06	-3.4366e-06	3.3472e-06	3.3211e-06	3.7048e-06
C_us	3.4533e-06	7.9479e-06	6.6161e-06	4.5393e-06	4.3884e-06	3.2857e-06
IODE	5.8208e-11	-1.6735e-10	-1.6735e-10	1.4552e-10	1.4552e-10	8.0036e-11
Idot	0.4049	-0.6855	-0.8374	-0.1539	-0.3538	0.1159
M_0	12	14	15	18	22	25
SV_ID	165600	165600	165600	165600	165600	165600
T_oe	0.0043	0.0067	0.0053	0.0137	0.0062	0.0021
eccentricity	0.3128	0.3105	0.2999	0.2952	0.2945	0.3092
i_0	0.0479	-0.6426	0.0428	-0.6775	-0.6485	0.1773
omega	-2.6282e-09	-2.5312e-09	-2.6914e-09	-2.8904e-09	-2.8300e-09	-2.6777e-09
omegaDOT	5.1536e+03	5.1537e+03	5.1537e+03	5.1538e+03	5.1537e+03	5.1536e+03
sqrt_A	-0.5251	0.8211	0.7905	0.4664	0.4673	-0.5369
w0						

Figura 25. Valores de las Efemérides capturados con Matlab

5.4. Valores de Poll Navigation Parameters

Los valores de las variables correspondientes al mensaje del título y que corresponden a lo que se obtiene con la función “pollNavigation.m”. Con esta función se desea capturar los parámetros de navegación del sistema GPS vigentes en el momento de la ejecución de la función.

Comparación entre los valores obtenidos en Matlab con los capturados con SiRF Demo.

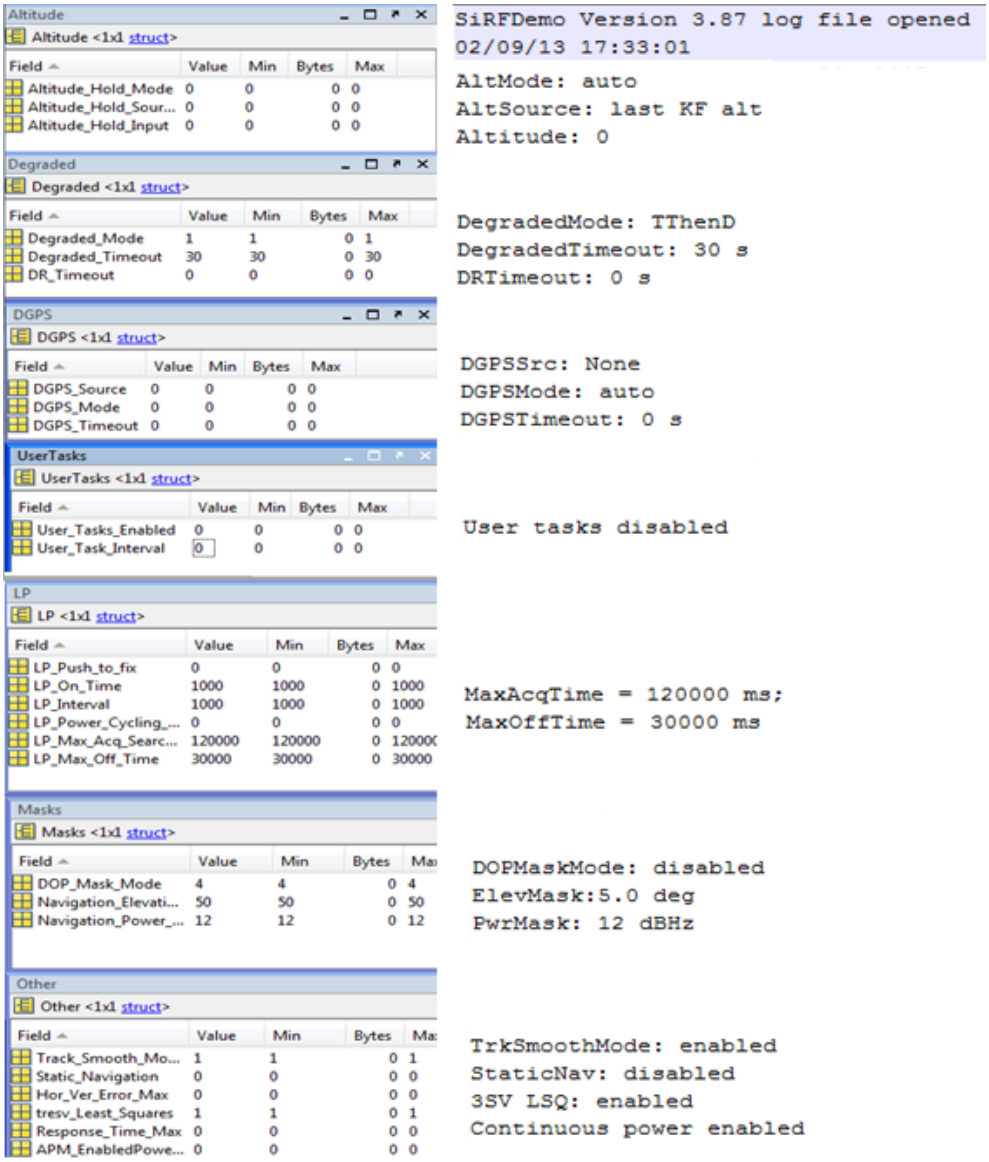

Figura 26. Parámetros de navegación capturados con Matlab y SiRF Demo

5.5. Valores de Navigation Library Measurement Data

Los valores de las variables correspondientes al mensaje del título y que corresponden a lo que se obtiene con la función “NLMeasurementData.m”. Con esta función se desea capturar parámetros correspondientes a las pseudodistancias, distancias Doppler, hora del software del GPS y más datos especificados en las tablas que se pueden consultar en el apartado 3.4.5. Navigation Library Measurement Data.

Como los valores están tomados con unos segundos de diferencia, vemos que varias medidas difieren aunque muy poco. Esto es bueno porque nos indica que se están capturando los valores correctos.

tablaMeasurement <23x23 double>

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	1	11	7	5.7941e+05	5.0048e+07	1.8475e+04	1.1125e-308	30000	35	35	34	34	34	34	34	34	34	34	1000	500	65279	0	0
2	2	11	5	5.7941e+05	1.1125e-308	5.8775e-39	1.1125e-308	30000	11	11	11	11	10	10	10	10	10	10	0	0	0	50	45
3	3	11	26	5.7941e+05	4.6066e+07	1.7705e+04	1.1125e-308	30000	36	36	36	36	36	36	36	35	35	35	1000	500	65023	0	0
4	4	11	15	5.7941e+05	4.8656e+07	1.7371e+04	1.1125e-308	30000	21	21	21	21	21	21	21	21	21	21	0	0	65279	50	0
5	5	11	10	5.7941e+05	4.9776e+07	1.8575e+04	1.1125e-308	30000	28	28	28	28	28	27	27	27	27	27	1000	500	65023	0	0
6	6	11	9	5.7941e+05	4.7680e+07	1.7362e+04	1.1125e-308	30000	17	17	17	17	17	17	17	17	17	17	0	0	65023	50	0
7	7	11	21	5.7941e+05	1.1125e-308	5.8775e-39	1.1125e-308	2560	7	7	7	7	7	7	7	7	7	7	0	0	0	50	63
8	8	11	28	5.7941e+05	4.7438e+07	1.7739e+04	1.1125e-308	30000	37	37	37	37	36	36	36	36	36	36	1000	500	65023	0	0
9	0	11	8	5.7941e+05	4.7621e+07	1.8284e+04	1.1125e-308	30000	42	42	42	42	42	42	42	42	42	41	1000	500	65279	0	0
10	1	11	7	5.7941e+05	5.0067e+07	1.8474e+04	1.1125e-308	30000	35	35	34	34	34	34	34	34	34	34	1000	500	65279	0	0
11	2	11	5	5.7941e+05	1.1125e-308	5.8775e-39	1.1125e-308	30000	12	12	12	12	12	11	11	11	11	11	0	0	0	50	35
12	3	11	26	5.7941e+05	4.6084e+07	1.7705e+04	1.1125e-308	30000	36	36	36	36	36	36	35	35	35	35	1000	500	65023	0	0
13	4	11	15	5.7941e+05	4.8673e+07	1.7370e+04	1.1125e-308	30000	22	21	21	21	21	21	21	21	21	21	0	0	65279	50	0
14	5	11	10	5.7941e+05	4.9795e+07	1.8575e+04	1.1125e-308	30000	28	28	28	28	27	27	27	27	27	27	1000	500	65023	0	0
15	6	11	9	5.7941e+05	4.7698e+07	1.7361e+04	1.1125e-308	30000	17	17	17	17	17	17	17	17	17	17	0	0	65023	50	0
16	7	11	28	5.7941e+05	4.7456e+07	1.7739e+04	1.1125e-308	30000	37	37	37	37	37	37	36	36	36	36	1000	500	65023	0	0
17	0	11	8	5.7941e+05	4.7639e+07	1.8284e+04	1.1125e-308	30000	42	42	42	42	42	42	42	42	42	42	1000	500	65279	0	0
18	1	11	7	5.7941e+05	5.0085e+07	1.8474e+04	1.1125e-308	30000	35	35	34	34	34	34	34	34	34	34	1000	500	65279	0	0
19	2	11	5	5.7941e+05	4.6833e+07	1.8115e+04	1.1125e-308	30000	12	12	12	12	12	12	11	11	11	11	0	0	65279	50	18
20	3	11	26	5.7941e+05	4.6102e+07	1.7705e+04	1.1125e-308	30000	36	36	36	36	36	35	35	35	35	35	1000	500	65023	0	0
21	4	11	15	5.7941e+05	4.8690e+07	1.7370e+04	1.1125e-308	30000	22	22	22	22	21	21	21	21	21	21	0	0	65279	50	0
22	5	11	10	5.7941e+05	4.9814e+07	1.8575e+04	1.1125e-308	30000	28	28	28	28	27	27	27	27	27	27	1000	500	65023	0	0
23	6	11	28	5.7941e+05	4.7474e+07	1.7739e+04	1.1125e-308	30000	37	37	37	37	37	37	37	36	36	36	1000	500	65023	0	0

Figura 27. Valores capturados con Matlab

```
SiRFDemo Version 3.87 log file opened 02/09/13 17:57:04
SiRFDemo Build Date: June 20, 2007

Tx PC Time=1360429024.000
Tx: 0xA0A2000284000084B0B3
28,8,12,28,5.79446088980336e+005,4.8147794263e+007,1.7744197266e+004,0.0000000000e+000,30000,191,36,36,36,36,36,36,36,36,35,1000,500,-3,0,0
28,0,12,8,5.79447089039965e+005,4.8352653776e+007,1.8286251953e+004,0.0000000000e+000,30000,191,42,42,42,41,41,41,41,41,41,1000,500,-2,0,0
28,1,12,7,5.79447089039965e+005,5.0805616470e+007,1.8475230469e+004,0.0000000000e+000,30000,191,36,36,36,35,35,35,35,35,35,1000,500,-2,0,0
28,2,12,21,5.79447089039965e+005,0.0000000000e+000,0.0000000000e+000,0.0000000000e+000,1760,76,16,16,16,16,16,16,15,15,15,15,0,0,0,50,0
28,3,12,26,5.79447089039965e+005,4.6792365545e+007,1.7708654297e+004,0.0000000000e+000,30000,191,33,33,33,33,33,33,33,33,32,1000,500,-3,0,0
28,4,12,15,5.79447089039965e+005,4.9367765402e+007,1.7371855469e+004,0.0000000000e+000,30000,45,25,25,25,25,25,24,24,24,24,0,0,-2,50,0
28,5,12,10,5.79447089039965e+005,5.0538063981e+007,1.8576800781e+004,0.0000000000e+000,30000,191,28,28,28,28,28,27,27,27,27,1000,500,-3,0,0
28,6,12,5,5.79447089039965e+005,4.7540007703e+007,1.8121314453e+004,0.0000000000e+000,12760,173,18,17,17,17,17,17,17,17,17,17,0,0,-3,50,0
28,7,12,9,5.79447089039965e+005,4.8392217045e+007,1.7365130859e+004,0.0000000000e+000,24560,45,22,22,21,21,21,21,21,21,21,0,0,-3,50,0
28,8,12,28,5.79447089039965e+005,4.8165539617e+007,1.7744339844e+004,0.0000000000e+000,30000,191,36,36,36,36,36,36,36,35,35,35,1000,500,-3,0,0
28,0,12,8,5.79448089099593e+005,4.8370941128e+007,1.8286294922e+004,0.0000000000e+000,30000,191,42,42,42,41,41,41,41,41,41,1000,500,-2,0,0
28,1,12,7,5.79448089099593e+005,5.0824092795e+007,1.8475218750e+004,0.0000000000e+000,30000,191,36,36,36,36,35,35,35,35,35,1000,500,-2,0,0
28,2,12,21,5.79448089099593e+005,5.2178649249e+007,1.7717332031e+004,0.0000000000e+000,2760,44,17,17,16,16,16,16,16,16,16,0,0,-2,50,0
```

Figura 28. Valores capturados en SiRF Demo

5.6. Pseudodistancias durante 60 minutos

En las siguientes imágenes vemos 2 gráficas de las pseudodistancias de los satélites 1 y 2. El eje vertical está representado en metros. El horizontal en minutos.

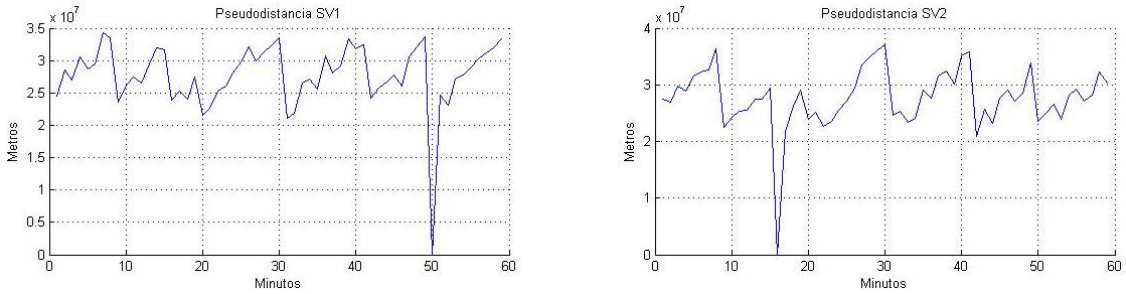

Figura 29. Gráficas de las pseudodistancias

Los valores obtenidos son del orden de lo que espera para las pseudodistancias ya que como hemos visto en el apartado anterior, captamos los mismos valores que SiRF Demo.

5.7. Pseudodistancias durante 6 horas

En las siguientes imágenes vemos, de nuevo, 2 gráficas de las pseudodistancias de los satélites 1 y 2. El eje vertical está representado en metros. El eje horizontal representa el número de capturas realizadas (74) a razón de una cada 5 minutos (370 minutos. 6 horas y 10 minutos). Se observan multitud de errores o valores no esperados.
Se esperaban obtener valores negativos en algún momento y que la gráfica tuviese forma de letra S puesta en horizontal, pero al no ser así, nos hemos dado cuenta que se deben aplicar correcciones posteriormente a la captura de este valor.

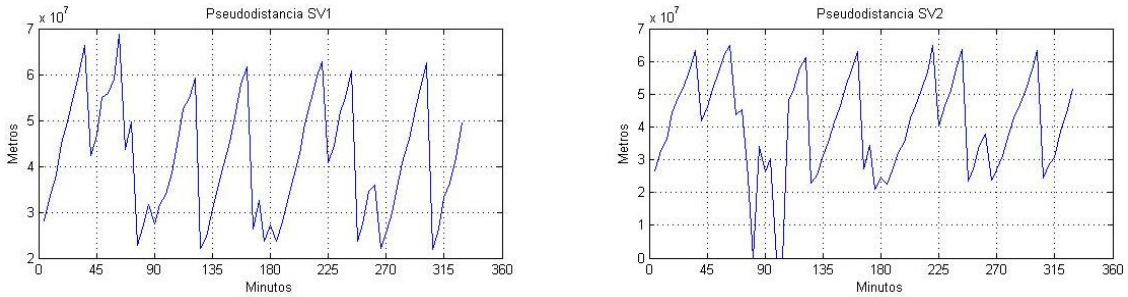

Figura 30. Gráficas de las pseudodistancias durante 6 horas y 10 minutos

5.8. Distancias Doppler durante 60 minutos

Las distancias Doppler se han tomado de los satélites 1 y 2. El eje vertical está en metros/segundo. El horizontal en minutos. Los valores obtenidos son del orden de lo que espera para el Carrier Frequency.

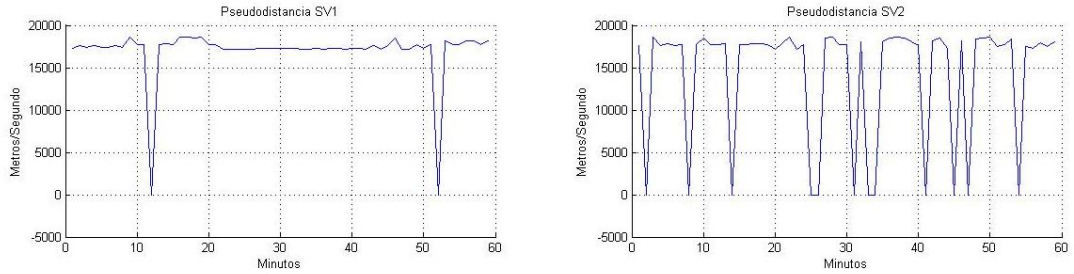

Figura 31. Gráficas de las distancias Doppler durante 60 minutos.

5.9. Distancias Doppler durante 6 horas.

En las siguientes imágenes vemos, de nuevo, 2 gráficas de las distancias Doppler de los satélites 1 y 2. El eje vertical está en metros/segundo. El eje horizontal representa el número de capturas realizadas (74) a razón de una cada 5 minutos (370 minutos. 6 horas y 10 minutos). Se observan multitud de errores o valores no esperados.
Aunque los valores parecen ser del orden de lo que se espera (en comparación con SiRF Demo, esperábamos unas gráficas con forma de letra U, que representa la distancia del satélite desde que aparece por el horizonte, en el valle de la U representa cuando está en nuestra vertical y cuando se pone por el otro extremo del horizonte.
Pero al no ser así, nos hemos dado cuenta que se deben aplicar correcciones posteriormente a la captura de este valor.

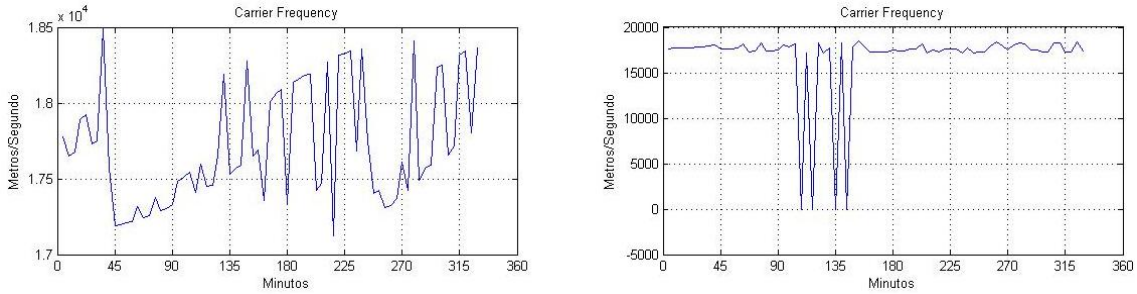

Figura 32. Gráficas de las distancias Doppler
durante 6 horas y 10 minutos

CONCLUSIONES

El objetivo de este proyecto ha sido obtener una visión completa de las técnicas y herramientas disponibles para poder capturar una serie de datos que, para el usuario normal, no son necesarios pero que en un entorno académico o de investigación pueden resultar útiles. Esto ha implicado realizar tareas de documentación sobre la estructura del mensaje que envía el sistema GPS, la estructura de los datos procesados mediante el protocolo SiRF, investigación de aplicaciones o comandos de MATLAB para crear y controlar las funciones que extraen los datos crudos de un receptor de GPS.

Se ha puesto especial énfasis en los elementos más importantes de la comunicación con el receptor. Así pues, se ha analizado el tipo de mensaje que envía el sistema GPS así como su interpretación mediante el protocolo SiRF, protocolo sobre el que gira todo el proyecto.

Tiene especial importancia el apartado sobre las rutinas ya que permitirá al lector entender mejor y de forma más rápida las diferentes funciones creadas y que por su extensión se ha decidido añadir en los anexos.

Se propone para el futuro desarrollar un código con más y mejor control de errores, una captura constante y un procesamiento al momento en lugar del actual, que se realiza una vez se ha cortado la comunicación. Sería interesante también crear una interfaz gráfica sobre la que controlar todas las funciones y que pudiese presentar los datos de forma más ordenada, además de una opción para abrir y cerrar la comunicación según la elección del usuario.

Añadir también que se han detectado numerosos errores en el manual del protocolo SiRF proporcionado por el propio fabricante, y que se repiten sistemáticamente en cada nueva actualización.

Consideramos que este proyecto fomenta una base sobre la que futuros proyectistas puedan realizar mejoras, así contribuir a la mejora constante de la idea inicial y ofrece una plataforma para uso académico que ayude a los futuros alumnos del ramo a conocer mejor un campo tan extendido como es el de las aplicaciones que usan el posicionamiento por satélite. Aplicaciones a las que ya hoy, nadie es ajeno.

REFERENCIAS

[1] SiRF
Web / URLs : http://www.csr.com/

[2] RoyalTek
Web / URLs : http://www.royaltek.com/

[3] GPS X-Mini RBT-2001 Royaltek
Web / URLs :
http://www.royaltek.com/download/RBT-2001%20User%20Manual%20V1.0.pdf

[4] National Marine Electronics Association (NMEA).
Web / URLs : http://www.nmea.org/

[5]WAAS / EGNOS System. Garmin.
Web / URLs : http://www.fisacaviation.com/garmin/info/waas/waas.htm

[6] Belkin Bluetooth USB Adapter F8T013
Web / ULRs :
http://www.belkin.com/IWCatProductPage.process?Product_Id=425750

[7] Matlab
Web / URLs : http://www.mathworks.es/

[8] Docklight RS232 Terminal Monitor
Web / URLs : http://www.docklight.de/

[9] Advanced Serial Port Monitor
Web / URLs : http://www.aggsoft.com/serial-port-monitor.htm

[10] SiRF Demo
Web / URLs http://www.falcom.de/support/software-tools/sirf/

[11] Transit (satellite). Wikipedia.
Web / URLs : http://en.wikipedia.org/wiki/Transit_(satellite)

[12] Timation. Wikipedia
Web / URLs : http://ncst-www.nrl.navy.mil/NCSTOrigin/Timation.html#Timation

[13] GPS.gov. Official U.S. Government information about the Global Positioning System (GPS) and related topics.
Web / URLs : http://www.gps.gov/

[14] Korean Air Lines Flight 007. Wikipedia.
Web / URLs : http://en.wikipedia.org/wiki/Korean_Air_Lines_Flight_007

[15] Tema 9: Triangulación y Trilateración. Universidad Politécnica de Madrid. Archivo PDF.
Web / URLs : http://ocw.upm.es/ingenieria-cartografica-geodesica-y-fotogrametria/topografia-ii/contenidos/Mis_documentos/Tema-9-Triangulacion-y-Trilateracion/Teoria_Triang_Tema_9.pdf

[16] Current and Features Satellite Generation. GPS.gov
Web / URLs : http://www.gps.gov/systems/gps/space/#generations

[17] Current GPS Constellation. U.S. Naval Observatory.
Web / URLs : http://www.usno.navy.mil/USNO/time/gps/current-gps-constellation

[18] Control Segment. GPS.gov
Web / URLs : http://www.gps.gov/systems/gps/control/

[19] EGNOS Precision
Web / URLs : http://www.esa.int/Our_Activities/Navigation/The_present_-_EGNOS/What_is_EGNOS

[20] GPS Modernization.
Web/URLs : http://www.gps.gov/systems/gps/modernization/

[21] New Civil Signals. GPS.gov
Web / URLs : http://www.gps.gov/systems/gps/modernization/civilsignals/

[22] Sistema Galileo
Web / URLs : http://www.esa.int/Our_Activities/Navigation/The_future_-_Galileo/What_is_Galileo

[23] ICD-GPS-200 - US Coast Guard Navigation Center. Archivo PDF.
Web / URLs : http://www.navcen.uscg.gov/pubs/gps/icd200/icd200cw1234.pdf

[24] Sistemas de Posicionamiento por Satélite. NAVSTAR GPS. Tema 5. Universitat Politécnica de Catalunya (UPC). José María González Arbesú.

[25] CSR PLC. Empresa propietaria del protocolo SiRF.
Web / URLs : http://www.csr.com/

[26] SiRFTech.
Web / URLs : http://w5.nuinternet.com/s660100031/SirfTech.htm

[27] IEEE coma flotante. Wikipedia.
Web / URLs : http://es.wikipedia.org/wiki/IEEE_coma_flotante

[28]Deriva del reloj.
Web / URLs :
http://books.google.es/books/about/Physics_for_Scientists_and_Engineers.html?id=6upvonUt0O8C&redir_esc=y

Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

ANNEXOS

TÍTULO DEL TFC: Utilidades de Matlab para la gestión de los datos crudos obtenidos por un receptor GPS con tecnología SiRF

TITULACIÓN: Ingeniería Técnica de Telecomunicaciones, especialidad Sistemas de Telecomunicaciones

AUTOR: Alberto Fernández Sierra

DIRECTOR: José María González Arbesú

FECHA: 1 de Febrero de 2013

1. FUNCIONES

1.1. MeasureNavDataOut.m

```
function [vPosition, vVelocity, HDOP, GPS_Week, GPS_TOW, SV_in_Fix, ...
    vCH]= MeasureNavDataOut(elport, velocidad, longitud_buffer)
%          MEASURE NAVIGATION DATA OUT - MESSAGE ID 2
%
% En las variables de entrada de la función hay que poner:
%
%   - elport: puerto de comunicación con el receptor GPS. Se introduce como
%             un string entre comillas simples. Por ejemplo: 'COM5'
%
%   - velocidad: velocidad de transferencia hasta 9600, 19200, 38400 y
%                57600 bauds / s.
%
%   - longitud_buffer: tamaño del buffer. Se aconseja un mínimo de
%                      6000 bits. Por ejemplo: 6000
%
% Ejemplo variables de entrada: [...] = MeasureNavDataOut('COM5',57600,6000)
%
% Esta función presenta parámetros de localización, velocidad y tiempo.
%
%       X_Position, Y_Position, Z_Position: Vector. Indican la posición en
%       coordenadas cartesianas. las unidades son "metros".
%       X_Velocity, Y_Velocity, Z_Velocity: Indican la velocidad a través
%       de coordenadas cartesianas. Las unidades son "metros / segundo"
%       HDOP: Incertidumbre en 2D. Adimensional.
%       GPS_Week: indica el valor de la semana actual.
%       SV_in_Fix: la cantidad de satélites usados para calcular los
parámetros.
%       GPS_TOW: segundos
%       vCH: valores codigos PRN. Vector. Adimensional.

s = serial(elport);
set(s, 'BaudRate', velocidad)
set(s, 'DataBits', 8)
set(s, 'InputBufferSize', longitud_buffer);
set(s, 'Parity','none')
set(s, 'StopBits', 1)
set(s, 'Terminator', 'LF/CR')
set(s, 'FlowControl', 'none')


fopen(s);

DataReceive = fread(s);
DataReceiveHex = dec2hex (DataReceive);


fclose(s);
delete(s);
clear s

nel = numel( DataReceive);
valorsTrama = [];
x = 0;

checksumTrama = [];
for i=1:nel
    var = DataReceive(i);
    if DataReceive(i) == hex2dec(('A0'))
```

```
        if DataReceive(i+1) == hex2dec(('A2'))

            if DataReceive(i+2) == hex2dec(('00'))

                if DataReceive(i+3) == hex2dec(('29'))

                    if DataReceive(i+4) == hex2dec(('02'))

                        valorsTrama = DataReceive(i + 4: i + 4 + 40);
                        checksumTrama = DataReceive(i + 4 + 41 : i + 4 + 41+
1);
                    end
                end
            end
        end
    end
end

valorsTramaHex = dec2hex(valorsTrama);
Message_ID = valorsTrama(1);

a = strcat(valorsTramaHex(2,1), valorsTramaHex(2,2));
b = strcat(valorsTramaHex(3,1), valorsTramaHex(3,2));
c = strcat(valorsTramaHex(4,1), valorsTramaHex(4,2));
d = strcat(valorsTramaHex(5,1), valorsTramaHex(5,2));
concate = strcat(a,b,c,d);
X_Position = hex2dec(concate);
clear concate a b c d

a = strcat(valorsTramaHex(6,1), valorsTramaHex(6,2));
b = strcat(valorsTramaHex(7,1), valorsTramaHex(7,2));
c = strcat(valorsTramaHex(8,1), valorsTramaHex(8,2));
d = strcat(valorsTramaHex(9,1), valorsTramaHex(9,2));
concate = strcat(a,b,c,d);
Y_Position = hex2dec(concate);
clear concate a b c d

a = strcat(valorsTramaHex(10,1), valorsTramaHex(10,2));
b = strcat(valorsTramaHex(11,1), valorsTramaHex(11,2));
c = strcat(valorsTramaHex(12,1), valorsTramaHex(12,2));
d = strcat(valorsTramaHex(13,1), valorsTramaHex(13,2));
concate = strcat(a,b,c,d);
Z_Position = hex2dec(concate);
clear concate a b c d

a = strcat(valorsTramaHex(14,1), valorsTramaHex(14,2));
b = strcat(valorsTramaHex(15,1), valorsTramaHex(15,2));
concate = strcat(a,b);
X_Velocity = hex2dec(concate)/8;
clear concate a b

a = strcat(valorsTramaHex(16,1), valorsTramaHex(16,2));
b = strcat(valorsTramaHex(17,1), valorsTramaHex(17,2));
concate = strcat(a,b);
Y_Velocity = hex2dec(concate)/8;
clear concate a b

a = strcat(valorsTramaHex(18,1), valorsTramaHex(18,2));
b = strcat(valorsTramaHex(19,1), valorsTramaHex(19,2));
concate = strcat(a,b);
Z_Velocity = hex2dec(concate)/8;
clear concate a b

HDOP = valorsTrama(21)/5;

a = strcat(valorsTramaHex(23,1), valorsTramaHex(23,2));
b = strcat(valorsTramaHex(24,1), valorsTramaHex(24,2));
```

```
concate = strcat(a,b);
GPS_Week = hex2dec(concate);
clear concate a b

a = strcat(valorsTramaHex(25,1), valorsTramaHex(25,2));
b = strcat(valorsTramaHex(26,1), valorsTramaHex(26,2));
c = strcat(valorsTramaHex(27,1), valorsTramaHex(27,2));
d = strcat(valorsTramaHex(28,1), valorsTramaHex(28,2));
concate = strcat(a,b,c,d);
GPS_TOW = hex2dec(concate)/100;
clear concate a b c d

SV_in_Fix = valorsTrama(29);
CH1_Prn = valorsTrama(30);
CH2_Prn = valorsTrama(31);
CH3_Prn = valorsTrama(32);
CH4_Prn = valorsTrama(33);
CH5_Prn = valorsTrama(34);
CH6_Prn = valorsTrama(35);
CH7_Prn = valorsTrama(36);
CH8_Prn = valorsTrama(37);
CH9_Prn = valorsTrama(38);
CH10_Prn = valorsTrama(39);
CH11_Prn = valorsTrama(40);
CH12_Prn = valorsTrama(41);
vPosition = [X_Position, Y_Position, Z_Position];
vVelocity = [X_Velocity, Y_Velocity, Z_Velocity];
vCH = [CH1_Prn, CH2_Prn, CH3_Prn, CH4_Prn, CH5_Prn, CH6_Prn, CH7_Prn, CH8_Prn,
CH9_Prn, CH10_Prn,CH11_Prn, CH12_Prn];
```

1.2. PollAlmanac.m

```
function [SV_ID, Week, Status, Eccentricity,Time_of_Applicability,...
    Orbital_Inclination, Rate_of_Right_Ascen, Raiz_A,...
    Right_Ascen_at_Week,Argument_of_Perigee, Mean_Anom, Af0, Af1]...
    = PollAlmanac(elport,velocidad, longitud_buffer)
%          ALMANAC DATA - MESSAGE ID 14
%
% En las variables de entrada de la función hay que poner:
%
%   - elport: puerto de comunicación con el receptor GPS. Se introduce como
%             un string entre comillas simples. Por ejemplo: 'COM5'
%
%   - velocidad: velocidad de transferencia hasta 9600, 19200, 38400 y
%                57600 bauds / s.
%
%   - longitud_buffer: tamaño del buffer. Se aconseja un mínimo de
%                      6000 bits. Por ejemplo: 6000
%
% Ejemplo variables de entrada: [...] = PollAlmanac('COM5',57600,6000)
%
% Esta función devuelve los parametros del Almanaque. Responde a la petición
% Poll Almanac que corresponde al MESSAGE ID 146
%         Poll Almanac =  A0 A2 00 02 92 00 00 92 B0 B3
%
% De los 30 bytes de datos, el 1 es el Message ID, el 2 el SV ID, del 3° y 4°
% byte, los primeros 10 bits son la semana y los 6 bits restantes el estado.
% Finalmente vienen 24 bytes de datos del almanaque. Se reciben en un array
% de longitud 12 con 2 bytes en cada posición (total, 24 bytes).
%
%       Eccentricity - Cuán estirada es una elipse. adimensional
%       Time_Of_Aplicability - El número del segundo en órbita cuando el
%                             almanaque fue generado.  segundos
%       Orbital_Inclination - Inclinación de la órbita. semi-circulos
%       Rate_of_Right_Ascen - Tasa de cambio en la medición del angulo de
%                             ascensión recta. semi-circulos/segundo
%       Raiz_A - Medida desde el centro de la orbita hasta el punto de
%               apogeo o el punto de perigeo. metros^(1/2)
%       Right_Ascen_at_Week - Ascensión Recta angular desde el equinocio
%                           vernal. semi-circulos
%       Argument_of_Perigee - Medida angular a lo largo de la trayectoria
%       orbital, desde el nodo ascendente hasta el punto de perigeo,
%       tomada en la dirección de movimiento del satélite. semi-circulos
%       Mean_Anom - Ángulo a través del nodo ascendente. semi-circulos
%       Af0 - Clock Bias del satélite. segundos
%       Af1 - Deriva del reloj en segundos / segundos

s = serial(elport);      %crear objeto serial
set(s, 'BaudRate', velocidad)
set(s, 'DataBits', 8)
set(s, 'InputBufferSize', longitud_buffer);
set(s, 'Parity','none')
set(s, 'StopBits', 1)
set(s, 'Terminator', 'LF/CR')
set(s, 'FlowControl', 'none')
%set(s, 'TimeOut', 5)
%POLL ALMANAC DATA%

fopen(s);

DataToSend = ['A0';'A2';'00';'02';'92';'00';'00';'92';'B0';'B3'];
DataDec = hex2dec(DataToSend);

fwrite(s,DataDec,'uint8');

DataReceive = fread(s);
```

```
DataHex = dec2hex(DataReceive);
fclose(s);
delete(s);
clear s

nel = numel( DataReceive);
valorsTrama = [];
x = 0;
i=1;
m=33;
j=1;
checksumTrama = [];
for i=m:nel
%     var = DataReceive(i);
    if DataReceive(i) == hex2dec(('A0'))

        if DataReceive(i+1) == hex2dec(('A2'))

            if DataReceive(i+2) == hex2dec(('00'))

                if DataReceive(i+3) == hex2dec(('1E'))

                    if DataReceive(i+4) == hex2dec(('0E'))

                        valorsTrama(:,j) = DataReceive(i + 4: i + 4 + 29);
                        checksumTrama(:,j) = DataReceive(i + 4 + 30 : i + 4 +
30+ 1);
                        i=i+m;
                        j=j+1;
                    end
                end
            end
        end
    end
end

for i=1:30
    for k=1:(j-1)
     valorsTramaHex{i,k} = dec2hex(valorsTrama(i,k),8);
    end
end

for i=1:30
    for j=1:k

         valorsBin{i,j} = dec2bin(valorsTrama(i,j),8);

    end

end


Message_ID = [];
SV_ID = [];

for t=1:k
    Message_ID = dec2hex(valorsTrama(1,:));
    SV_ID = valorsTrama(2,:);
end

valorsTramaBin_1 = dec2bin(valorsTrama(3,:));
valorsTramaBin_2 = dec2bin(valorsTrama(4,:));
valorsTramaBin_3 = strcat(valorsTramaBin_1,valorsTramaBin_2);
valorsWeek = [];

for i=1:10
    for j=1:k
        valorsWeek = strcat(valorsTramaBin_3(:,1:10));
```

```
        Week = bin2dec(valorsWeek);

    end
end

Status = bin2dec(strcat(valorsTramaBin_3(:,11:16)));

valorsTramaBin_5 = dec2bin(valorsTrama(5,:),8);
valorsTramaBin_6 = dec2bin(valorsTrama(6,:),8);
valorsTramaBin_7 = dec2bin(valorsTrama(7,:),8);
valorsTramaBin_8 = dec2bin(valorsTrama(8,:),8);
valorsTramaBin_9 = dec2bin(valorsTrama(9,:),8);
valorsTramaBin_10 = dec2bin(valorsTrama(10,:),8);
valorsTramaBin_11 = dec2bin(valorsTrama(11,:),8);
valorsTramaBin_12 = dec2bin(valorsTrama(12,:),8);
valorsTramaBin_13 = dec2bin(valorsTrama(13,:),8);
valorsTramaBin_14 = dec2bin(valorsTrama(14,:),8);
valorsTramaBin_15 = dec2bin(valorsTrama(15,:),8);
valorsTramaBin_16 = dec2bin(valorsTrama(16,:),8);
valorsTramaBin_17 = dec2bin(valorsTrama(17,:),8);
valorsTramaBin_18 = dec2bin(valorsTrama(18,:),8);
valorsTramaBin_19 = dec2bin(valorsTrama(19,:),8);
valorsTramaBin_20 = dec2bin(valorsTrama(20,:),8);
valorsTramaBin_21 = dec2bin(valorsTrama(21,:),8);
valorsTramaBin_22 = dec2bin(valorsTrama(22,:),8);
valorsTramaBin_23 = dec2bin(valorsTrama(23,:),8);
valorsTramaBin_24 = dec2bin(valorsTrama(24,:),8);
valorsTramaBin_25 = dec2bin(valorsTrama(26,:),8);
valorsTramaBin_26 = dec2bin(valorsTrama(27,:),8);
valorsTramaBin_27 = dec2bin(valorsTrama(27,:),8);
valorsTramaBin_28 = dec2bin(valorsTrama(28,:),8);
valorsTramaBin_29 = dec2bin(valorsTrama(29,:),8);
valorsTramaBin_30 = dec2bin(valorsTrama(30,:),8);

Eccentricity = (bin2dec(strcat(valorsTramaBin_6,valorsTramaBin_7)))*2^(-21);

Time_of_Applicability = (bin2dec(strcat(valorsTramaBin_8)))*2^(12);

Orbital = (strcat(valorsTramaBin_9,valorsTramaBin_10));
val = [];
value = [];
Orbital_Inclination = [];

for i=1:k

val = bin2dec(Orbital(i,:));
y = sign(2^(16-1)-val)*(2^(16-1)-abs(2^(16-1)-val));
    if ((y == 0) && (val ~= 0))
        value{i} = -val;
    else
        value{i} = y;
    end
end
dates = (cell2mat(value));
Orbital_Inclination = (dates*2^(-19)+0.30)*pi;


Rate_of_Right_Ascen = [];
RateAscen = (strcat(valorsTramaBin_11,valorsTramaBin_12));
val = [];
value = [];
for i=1:k

val = bin2dec(RateAscen(i,:));
y = sign(2^(16-1)-val)*(2^(16-1)-abs(2^(16-1)-val));
    if ((y == 0) && (val ~= 0))
        value{i} = -val;
    else
```

```
        value{i} = y;
    end
end
dates = (cell2mat(value));
Rate_of_Right_Ascen = (((dates*2^(-38)))*pi);

Raiz = bin2dec(strcat(valorsTramaBin_14,valorsTramaBin_15,valorsTramaBin_16));
Raiz_A = Raiz*2^(-11);


Right_Ascen_at_Week = [];
RightAscenWeek =
(strcat(valorsTramaBin_17,valorsTramaBin_18,valorsTramaBin_19));
val = [];
value = [];
for i=1:k

val = bin2dec(RightAscenWeek(i,:));
y = sign(2^(24-1)-val)*(2^(24-1)-abs(2^(24-1)-val));
    if ((y == 0) && (val ~= 0))
        value{i} = -val;
    else
        value{i} = y;
    end
end
dates = (cell2mat(value));
Right_Ascen_at_Week = (((dates*2^(-23)))*pi);

Argument_of_Perigee = [];
omega = (strcat(valorsTramaBin_20,valorsTramaBin_21,valorsTramaBin_22));
val = [];
value = [];
for i=1:k

val = bin2dec(omega(i,:));
y = sign(2^(24-1)-val)*(2^(24-1)-abs(2^(24-1)-val));
    if ((y == 0) && (val ~= 0))
        value{i} = -val;
    else
        value{i} = y;
    end
end
dates = (cell2mat(value));
Argument_of_Perigee = (((dates*2^(-23)))*pi);

Mean_Anom = [];
Mean = (strcat(valorsTramaBin_23,valorsTramaBin_24,valorsTramaBin_25));
val = [];
value = [];
for i=1:k

val = bin2dec(Mean(i,:));
y = sign(2^(24-1)-val)*(2^(24-1)-abs(2^(24-1)-val));
    if ((y == 0) && (val ~= 0))
        value{i} = -val;
    else
        value{i} = y;
    end
end
dates = (cell2mat(value));
Mean_Anom = (((dates*2^(-23)))*pi);


Af0 = [];
ff = (strcat(valorsTramaBin_26, valorsTramaBin_28(:,4:6)));
val = [];
value = [];
```

```
for i=1:k
val = bin2dec(ff(i,:));
y = sign(2^(11-1)-val)*(2^(11-1)-abs(2^(11-1)-val));
    if ((y == 0) && (val ~= 0))
        value{i} = -val;
    else
        value{i} = y;
    end
end
dates = (cell2mat(value));
Af0 = dates*2^(-20);

Af1 = [];
ff = (strcat(valorsTramaBin_27, valorsTramaBin_28));
val = [];
value = [];
for i=1:k
     f1 = ff(:,1:11);
end
%
for i=1:k
val = bin2dec(f1(i,:));
y = sign(2^(11-1)-val)*(2^(11-1)-abs(2^(11-1)-val));
    if ((y == 0) && (val ~= 0))
        value{i} = -val;
    else
        value{i} = y;
    end
end
dates = (cell2mat(value));
Af1 = dates*2^(-38);
```

1.3. PollEpheme.m

```
function [SV_ID, IODE, C_rs,A_n, M_0, C_uc, eccentricity, C_us, sqrt_A,...
    T_oe, C_ic, w0, C_is, i_0, C_rc, omega, omegaDOT, Idot]...
    = PollEpheme(elport,velocidad,longitud_buffer)
%          EPHEMERIS DATA - MESSAGE ID 15
%
% En las variables de entrada de la función hay que poner:
%
%   - elport: puerto de comunicación con el receptor GPS. Se introduce como
%             un string entre comillas simples. Por ejemplo: 'COM5'
%
%   - velocidad: velocidad de transferencia hasta 9600, 19200, 38400 y
%                57600 bauds / s.
%
%   - longitud_buffer: tamaño del buffer. Se aconseja un mínimo de
%                      6000 bits. Por ejemplo: 6000
%
% Ejemplo variables de entrada: [...] = pollEpheme('COM5',57600,6000)
%
% Esta función devuelve los parametros de las Efemerides. Responde
% a la petición Poll Almanac que corresponde al MESSAGE ID 147
%
%         Poll Almanac =  A0 A2 00 03 93 00 00 00 93 B0 B3
%
% De los 92 bytes de datos, el 1 es el Message ID, el 2 el SV PRN code, los
% 90 restantes son los datos que conforman las efemerides.
%
%         IODE - Permite detectar cambios en el mensaje de navegación.
%         A_n - Diferencia media de movimiento a partir de un valor
%               calculado. semi-circulos / segundo
%         M_0s - Ángulo a través del nodo ascendente. semi-circulos
%         eccentricity - Cuán estirada es una elipse. adimensional
%         sqrt_A - Medida desde el centro de la orbita hasta el punto de
%                apogeo o el punto de perigeo. metros^(1/2)
%         Time_Of_Applicability - El número del segundo en órbita cuando el
%                                 almanaque fue generado.  segundos
%         w0 - Ascensión Recta angular desde el equinocio
%                             vernal. semi-circulos
%         C_is, C_rc, C_ic, C_uc C_us,C_rs - Diferentes amplitudes de
%                                       términos corregidos referidos a
%                                       los harmónicos del seno y el
%                                       coseno del argumento del perigeo,
%                                       el radio de la órbita y el
%                                       ángulo de inclinación. radianes
%         i_0 - Ángulo de inclinación. semi-circulos
%         omega - Medida angular a lo largo de la trayectoria
%                 orbital, desde el nodo ascendente hasta el punto de
%                 perigeo, tomada en la dirección de movimiento del
%                 satélite. semi-circulos
%         omegaDOT - Tasa de cambio en la medición del angulo de
%                                 ascensión recta. semi-circulos/segundo
%         Idot - Cambio de ángulo de inclinación. semi-circulos / segundo

s = serial(elport);      %crear objeto serial
set(s, 'BaudRate', velocidad)
set(s, 'DataBits', 8)
set(s, 'InputBufferSize', longitud_buffer);
set(s, 'Parity','none')
set(s, 'StopBits', 1)
set(s, 'Terminator', 'LF/CR')
set(s, 'FlowControl', 'none')
%set(s, 'TimeOut', 5)
```

```
fopen(s);

DataToSend = ['A0';'A2';'00';'03';'93';'00';'00';'00';'93';'B0';'B3'];
DataDec = hex2dec(DataToSend);

fwrite(s,DataDec,'uint8');

DataReceive = fread(s);
DataHex = dec2hex(DataReceive);
fclose(s);
delete(s);
clear s

nel = numel( DataReceive);
valorsTrama = [];
x = 0;
i=1;
m=97;
j=1;
checksumTrama = [];

for i=m:nel
%     var = DataReceive(i);
    if DataReceive(i) == hex2dec(('A0'))

        if DataReceive(i+1) == hex2dec(('A2'))

            if DataReceive(i+2) == hex2dec(('00'))

                if DataReceive(i+3) == hex2dec(('5C'))

                    if DataReceive(i+4) == hex2dec(('0F'))

                        valorsTrama(:,j) = DataReceive(i + 4: i + 4 + 91);
                        checksumTrama(:,j) = DataReceive(i + 4 +92 : i + 4 +
92+ 1);
                        i=i+m;
                        j=j+1;
                    end
                end
            end
        end
    end
end

for i=1:92
    for k=1:(j-1)
     valorsTramaHex{i,k} = dec2hex(valorsTrama(i,k));
    end
end

Message_ID = [];
SV_ID = [];

for t=1:k
    Message_ID = dec2hex(valorsTrama(1,:));
    SV_ID = valorsTrama(2,:);
end

valorsBin = [];

for i=1:92
    for j=1:k

         valorsBin{i,j} = dec2bin(valorsTrama(i,j),8);

    end
```

```
end

IODE = bin2dec(valorsBin(39,:));

Crs = (strcat(valorsBin(40,:), valorsBin(41,:)));
val = [];
value = [];
C_rs = [];
for i=1:k

val = bin2dec(Crs(1,i));
y = sign(2^(16-1)-val)*(2^(16-1)-abs(2^(16-1)-val));
    if ((y == 0) && (val ~= 0))
        value{i} = -val;
    else
        value{i} = y;
    end
end
dates = (cell2mat(value));
C_rs = (dates*2^(-5));

An = (strcat(valorsBin(42,:), valorsBin(43,:)));
val = [];
value = [];
A_n = [];
for i=1:k

val = bin2dec(An(1,i));
y = sign(2^(16-1)-val)*(2^(16-1)-abs(2^(16-1)-val));
    if ((y == 0) && (val ~= 0))
        value{i} = -val;
    else
        value{i} = y;
    end
end
dates = (cell2mat(value));
A_n = (dates*2^(-43));

M0 = (strcat(valorsBin(44,:), valorsBin(45,:),valorsBin(46,:),
valorsBin(47,:)));
val = [];
value = [];
M_0 = [];
for i=1:k

val = bin2dec(M0(1,i));
y = sign(2^(32-1)-val)*(2^(32-1)-abs(2^(32-1)-val));
    if ((y == 0) && (val ~= 0))
        value{i} = -val;
    else
        value{i} = y;
    end
end
dates = (cell2mat(value));
M_0 = (dates*2^(-31));

CUC = (strcat(valorsBin(48,:), valorsBin(49,:)));
val = [];
value = [];
C_uc = [];
for i=1:k

val = bin2dec(CUC(1,i));
y = sign(2^(16-1)-val)*(2^(16-1)-abs(2^(16-1)-val));
    if ((y == 0) && (val ~= 0))
        value{i} = -val;
    else
        value{i} = y;
```

```
    end
end
dates = (cell2mat(value));
C_uc = (dates*2^(-29));

e = (strcat(valorsBin(50,:), valorsBin(51,:),valorsBin(52,:),
valorsBin(53,:)));
eccentricity = bin2dec(e)*2^(-33);

CUS = (strcat(valorsBin(54,:), valorsBin(55,:)));
val = [];
value = [];
C_us = [];
for i=1:k

val = bin2dec(CUS(1,i));
y = sign(2^(16-1)-val)*(2^(16-1)-abs(2^(16-1)-val));
    if ((y == 0) && (val ~= 0))
        value{i} = -val;
    else
        value{i} = y;
    end
end
dates = (cell2mat(value));
C_us = (dates*2^(-29));

raizA = (strcat(valorsBin(56,:), valorsBin(57,:),valorsBin(58,:),
valorsBin(59,:)));
sqrt_A = bin2dec(raizA)*2^(-19);

TOE = (strcat(valorsBin(60,:), valorsBin(61,:)));
T_oe = bin2dec(TOE)*2^(4);

CIC = (strcat(valorsBin(69,:), valorsBin(70,:)));
val = [];
value = [];
C_ic = [];
for i=1:k

val = bin2dec(CIC(1,i));
y = sign(2^(16-1)-val)*(2^(16-1)-abs(2^(16-1)-val));
    if ((y == 0) && (val ~= 0))
        value{i} = -val;
    else
        value{i} = y;
    end
end
dates = (cell2mat(value));
C_ic = (dates*2^(-29));

Omega0 = (strcat(valorsBin(71,:), valorsBin(72,:),valorsBin(73,:),
valorsBin(74,:)));
val = [];
value = [];
w0 = [];
for i=1:k

val = bin2dec(Omega0(1,i));
y = sign(2^(32-1)-val)*(2^(32-1)-abs(2^(32-1)-val));
    if ((y == 0) && (val ~= 0))
        value{i} = -val;
    else
        value{i} = y;
    end
end
dates = (cell2mat(value));
w0 = (dates*2^(-31));
```

```
CIS = (strcat(valorsBin(75,:), valorsBin(76,:)));
val = [];
value = [];
C_is = [];
for i=1:k

val = bin2dec(CIS(1,i));
y = sign(2^(16-1)-val)*(2^(16-1)-abs(2^(16-1)-val));
    if ((y == 0) && (val ~= 0))
        value{i} = -val;
    else
        value{i} = y;
    end
end
dates = (cell2mat(value));
C_is = (dates*2^(-29));

i0 = (strcat(valorsBin(77,:), valorsBin(78,:),valorsBin(79,:),
valorsBin(80,:)));
val = [];
value = [];
i_0 = [];
for i=1:k

val = bin2dec(i0(1,i));
y = sign(2^(32-1)-val)*(2^(32-1)-abs(2^(32-1)-val));
    if ((y == 0) && (val ~= 0))
        value{i} = -val;
    else
        value{i} = y;
    end
end
dates = (cell2mat(value));
i_0 = (dates*2^(-31));

CRC = (strcat(valorsBin(81,:), valorsBin(82,:)));
val = [];
value = [];
C_rc = [];
for i=1:k

val = bin2dec(CRC(1,i));
y = sign(2^(16-1)-val)*(2^(16-1)-abs(2^(16-1)-val));
    if ((y == 0) && (val ~= 0))
        value{i} = -val;
    else
        value{i} = y;
    end
end
dates = (cell2mat(value));
C_rc = (dates*2^(-5));

ome = (strcat(valorsBin(83,:), valorsBin(84,:),valorsBin(85,:),
valorsBin(86,:)));
val = [];
value = [];
omega = [];
for i=1:k

val = bin2dec(ome(1,i));
y = sign(2^(32-1)-val)*(2^(32-1)-abs(2^(32-1)-val));
    if ((y == 0) && (val ~= 0))
        value{i} = -val;
    else
        value{i} = y;
    end
end
dates = (cell2mat(value));
```

```
omega = (dates*2^(-31));

omeDOT = (strcat(valorsBin(87,:), valorsBin(88,:),valorsBin(89,:)));
val = [];
value = [];
omegaDOT = [];
for i=1:k

val = bin2dec(omeDOT(1,i));
y = sign(2^(24-1)-val)*(2^(24-1)-abs(2^(24-1)-val));
    if ((y == 0) && (val ~= 0))
        value{i} = -val;
    else
        value{i} = y;
    end
end
dates = (cell2mat(value));
omegaDOT = (dates*2^(-43));

IODE_2 = bin2dec(valorsBin(90,:));

Idot = [];
ff = (strcat(valorsBin(91,:),'000000'));

for i=1:k
val = bin2dec(ff(1,i));
y = sign(2^(14-1)-val)*(2^(14-1)-abs(2^(14-1)-val));
    if ((y == 0) && (val ~= 0))
        value{i} = -val;
    else
        value{i} = y;
    end
end
dates = (cell2mat(value));
Idot = dates*2^(-43);



valorsTramaBin_1 = dec2bin(valorsTrama(3,:),8);
IODE = valorsTramaBin_1;

valorsTramaBin_2 = dec2bin(valorsTrama(6,:),8);
valorsTramaBin_3 = dec2bin(valorsTrama(7,:),8);
```

1.4. pollNavigation.m

```
function [Altitude, Degraded, Masks, DGPS, LP, UserTasks, Other]...
    = pollNavigation(elport,velocidad,longitud_buffer)
%          NAVIGATION PARAMETERS - MESSAGE ID 19
%
% En las variables de entrada de la función hay que poner:
%
%   - elport: puerto de comunicación con el receptor GPS. Se introduce como
%             un string entre comillas simples. Por ejemplo: 'COM5'
%
%   - velocidad: velocidad de transferencia hasta 9600, 19200, 38400 y
%                57600 bauds / s.
%
%   - longitud_buffer: tamaño del buffer. Se aconseja un mínimo de
%                      6000 bits. Por ejemplo: 6000
%
% Ejemplo variables de entrada: [...] = pollNavigation('COM5',57600,6000)
%
% Esta función presenta los parametros actuales de navegación.
% Responde a la petición Poll Almanac que corresponde al MESSAGE ID 147
%
%         Poll Almanac =  A0 A2 00 02 98 00 00 98 B0 B3
%
% Las variables vienen almacenadas en estructuras:
%
%        Altitude / Degraded: definen el modo de funcionamiento
%                             cuando la recepción óptima (4 satélites)
%                             no está disponible.
%        Masks: Angulos para obviar satélites en caso de que estos
%               esten muy cerca del horizonte.
%        DGPS: Información sobre el GPS diferencial. Controla la activación
%        de SBAS (WAAS en los EEUU y EGNOS en Europa).
%        LP: Información relacionada con el inició rápido del GPS.
%        UserTasks: Este parámetro lo proporciona el software y es
%                   imposible modificarlo.
%        Other: otro tipo de datos relacionados con el tiempo entre
%               fijacion de satélites (segundos), los errores verticales y
%               horizontales(metros), tiempo máximo de respuesta(segundos),
%               los ciclos.

s = serial(elport);      %crear objeto serial
set(s, 'BaudRate', velocidad)
set(s, 'DataBits', 8)
set(s, 'InputBufferSize', longitud_buffer);
set(s, 'Parity','none')
set(s, 'StopBits', 1)
set(s, 'Terminator', 'LF/CR')
set(s, 'FlowControl', 'none')
%set(s, 'TimeOut', 5)
%POLL NAVIGATION PARAMETERS%

fopen(s);
DataToSend = ['A0';'A2';'00';'02';'98';'00';'00';'98';'B0';'B3'];
DataDec = hex2dec(DataToSend);

fwrite(s,DataDec,'uint8');

%pause(45);

DataReceive = fread(s);
DataHex = dec2hex(DataReceive);
fclose(s);
delete(s);
```

```
clear s

nel = numel( DataReceive);
valorsTrama = [];
x = 0;

checksumTrama = [];
for i=1:nel
    var = DataReceive(i);
    if DataReceive(i) == hex2dec(('A0'))

        if DataReceive(i+1) == 162

            if DataReceive(i+2) == 0

                if DataReceive(i+3) == 65

                    if DataReceive(i+4) == 19

                        valorsTrama = DataReceive(i + 4: i + 4 + 64);
                        checksumTrama = DataReceive(i + 4 + 65 : i + 4 + 65 +
1);
                    end
                end
            end
        end
    end
end
valorsTramaHex = dec2hex(valorsTrama);

Message_ID = valorsTrama(1,:);
Altitude_Hold_Mode = valorsTrama(6);
Altitude_Hold_Source = valorsTrama(7);
a = strcat(valorsTramaHex(8,1), valorsTramaHex(8,2));
b = strcat(valorsTramaHex(9,1), valorsTramaHex(9,2));
concate = strcat(a,b);
Altitude_Hold_Input = hex2dec(concate);
clear concate a b

Degraded_Mode = valorsTrama(10);
Degraded_Timeout = valorsTrama(11);
DR_Timeout = valorsTrama(12);
Track_Smooth_Mode = valorsTrama(13);
Static_Navigation = valorsTrama(14);
tresv_Least_Squares = valorsTrama(15);

DOP_Mask_Mode = valorsTrama(20);

a = strcat(valorsTramaHex(21,1), valorsTramaHex(21,2));
b = strcat(valorsTramaHex(22,1), valorsTramaHex(22,2));
concate = strcat(a,b);
Navigation_Elevation_Mask = hex2dec(concate);
clear concate a b

Navigation_Power_Mask = valorsTrama(23);

DGPS_Source = valorsTrama(28);
DGPS_Mode = valorsTrama(29);
DGPS_Timeout = valorsTrama(30);

LP_Push_to_fix = valorsTrama(35);

a = strcat(valorsTramaHex(36,1), valorsTramaHex(36,2));
b = strcat(valorsTramaHex(37,1), valorsTramaHex(37,2));
c = strcat(valorsTramaHex(38,1), valorsTramaHex(38,2));
d = strcat(valorsTramaHex(39,1), valorsTramaHex(39,2));
concate = strcat(a,b,c,d);
LP_On_Time = hex2dec(concate);
```

```
clear concate a b c d

a = strcat(valorsTramaHex(36,1), valorsTramaHex(36,2));
b = strcat(valorsTramaHex(37,1), valorsTramaHex(37,2));
c = strcat(valorsTramaHex(38,1), valorsTramaHex(38,2));
d = strcat(valorsTramaHex(39,1), valorsTramaHex(39,2));
concate = strcat(a,b,c,d);
LP_On_Time = hex2dec(concate);
clear concate a b c d

a = strcat(valorsTramaHex(40,1), valorsTramaHex(40,2));
b = strcat(valorsTramaHex(41,1), valorsTramaHex(41,2));
c = strcat(valorsTramaHex(42,1), valorsTramaHex(42,2));
d = strcat(valorsTramaHex(43,1), valorsTramaHex(43,2));
concate = strcat(a,b,c,d);
LP_Interval = hex2dec(concate);
clear concate a b c d

User_Tasks_Enabled = valorsTrama(44);

a = strcat(valorsTramaHex(45,1), valorsTramaHex(45,2));
b = strcat(valorsTramaHex(46,1), valorsTramaHex(46,2));
c = strcat(valorsTramaHex(47,1), valorsTramaHex(47,2));
d = strcat(valorsTramaHex(48,1), valorsTramaHex(48,2));
concate = strcat(a,b,c,d);
User_Task_Interval = hex2dec(concate);
clear concate a b c d

LP_Power_Cycling_Enabled = valorsTrama(49);

a = strcat(valorsTramaHex(50,1), valorsTramaHex(50,2));
b = strcat(valorsTramaHex(51,1), valorsTramaHex(51,2));
c = strcat(valorsTramaHex(52,1), valorsTramaHex(52,2));
d = strcat(valorsTramaHex(53,1), valorsTramaHex(53,2));
concate = strcat(a,b,c,d);
LP_Max_Acq_Search_Time = hex2dec(concate);
clear concate a b c d

a = strcat(valorsTramaHex(54,1), valorsTramaHex(54,2));
b = strcat(valorsTramaHex(55,1), valorsTramaHex(55,2));
c = strcat(valorsTramaHex(56,1), valorsTramaHex(56,2));
d = strcat(valorsTramaHex(57,1), valorsTramaHex(57,2));
concate = strcat(a,b,c,d);
LP_Max_Off_Time = hex2dec(concate);
clear concate a b c d

APM_EnabledPower_Duty_Cycle = valorsTrama(58);

a = strcat(valorsTramaHex(59,1), valorsTramaHex(59,2));
b = strcat(valorsTramaHex(60,1), valorsTramaHex(60,2));
concate = strcat(a,b);
Number_of_Fixes = hex2dec(concate);
clear concate a b

a = strcat(valorsTramaHex(61,1), valorsTramaHex(61,2));
b = strcat(valorsTramaHex(62,1), valorsTramaHex(62,2));
concate = strcat(a,b);
Time_Between_Fixes = hex2dec(concate);
clear concate a b

Hor_Ver_Error_Max = valorsTrama(63);
Response_Time_Max = valorsTrama(64);

Altitude.Altitude_Hold_Mode = Altitude_Hold_Mode;
Altitude.Altitude_Hold_Source = Altitude_Hold_Source;
Altitude.Altitude_Hold_Input = Altitude_Hold_Input;
Degraded.Degraded_Mode = Degraded_Mode;
Degraded.Degraded_Timeout = Degraded_Timeout;
```

```
Degraded.DR_Timeout = DR_Timeout;
Masks.DOP_Mask_Mode = DOP_Mask_Mode;
Masks.Navigation_Elevation_Mask = Navigation_Elevation_Mask;
Masks.Navigation_Power_Mask = Navigation_Power_Mask;
DGPS.DGPS_Source = DGPS_Source;
DGPS.DGPS_Mode = DGPS_Mode;
DGPS.DGPS_Timeout = DGPS_Timeout;
LP.LP_Push_to_fix = LP_Push_to_fix;
LP.LP_On_Time = LP_On_Time;
LP.LP_Interval = LP_Interval;
LP.LP_Power_Cycling_Enabled = LP_Power_Cycling_Enabled;
LP.LP_Max_Acq_Search_Time = LP_Max_Acq_Search_Time;
LP.LP_Max_Off_Time = LP_Max_Off_Time;
UserTasks.User_Tasks_Enabled = User_Tasks_Enabled;
UserTasks.User_Task_Interval = User_Task_Interval;

DataFixes.Number_of_Fixes = Number_of_Fixes;
DataFixes.Time_Between_Fixes = Time_Between_Fixes;
Other.Track_Smooth_Mode = Track_Smooth_Mode;
Other.Static_Navigation = Static_Navigation;
Other.Hor_Ver_Error_Max = Hor_Ver_Error_Max;
Other.tresv_Least_Squares = tresv_Least_Squares;
Other.Response_Time_Max = Response_Time_Max;
Other.APM_EnabledPower_Duty_Cycle = APM_EnabledPower_Duty_Cycle;
```

1.5. NLMeasurementData.m

```
function [Channel, Time_Tag, Satellite_ID, GPS_Software, Pseudorange,...
    Carrier_Frequency, Carrier_Phase, Time_In_Track, tablaCanales,...
    Delta_Range_Interval, Mean_Delta_Range_Time, Extrapolation_Time, ...
    Phase_Error_Count, Low_Power_Count] =
NLMeasurementData(elport,velocidad,longitud_buffer)

%          NAVIGATION LIBRARY MEASUREMENT DATA - MESSAGE ID 28
%
% En las variables de entrada de la función hay que poner:
%
%   - elport: puerto de comunicación con el receptor GPS. Se introduce como
%             un string entre comillas simples. Por ejemplo: 'COM5'
%
%   - velocidad: velocidad de transferencia hasta 9600, 19200, 38400 y
%                57600 bauds / s.
%
%   - longitud_buffer: tamaño del buffer. Se aconseja un mínimo de
%                      12000 bits.
%
% Ejemplo variables de entrada: [...] = NLMeasurementData('COM5',57600,12000)
%
% Esta función presenta parámetros como Pseudodistancias, distancias
% Doppler, tiempo de GPS, etc.
% Tambien realiza la petición Poll Clock Status de forma interna, para poder
% extraer el Clock Drift, necesario para encontrar las distancias Doppler
% de forma correcta.
%
% La petición del Clock Status se realiza mediante el POLL CLOCK STATUS
% MESSAGE ID 144.
%
%         A0 A2 00 02 90 00 00 90 B0 B3
%
%       Time Tag: tiempo en milisegundos de medición del software del
%                 receptor. Se inicia cuando el receptor se enciende o se
%                 reinicia.
%       Satellite_ID: Identificación del satélite
%       GPS_Software: Tiempo de la semana (TOW) estimado en milisegundos.
%       Pseudorange: Medidas de pseudorango para un satélite en particular.
%                    En metros.
%       Carrier_Frequency: Frecuencia de la portadora. En m / s.
%       Carrier_Phase: en este caso y por el tipo de receptor, siempre es 0.
%       Time_in_Track: Para un satélite determinado, el tiempo que ha sido
%                      usado. En milisegundos.
%       tablaGanacia: Relación señal a ruido en 10 franjas
%                     de 100 milisegundos. En dB-Hz
%       Delta_Range_Interval: Intervalo de medición delta-pseudodistancia
%       Mean_Delta_Range_Time: tiempo medio del intervalo en milisengudos.
%                              delta-pseudodistancia en milisegundos
%       Extrapolation_Time: Valor reservado para este receptor.
%       Phase_Error_Count: recuento de errores de fase mayores de 60 grados
%       Low_Power_Count: cuenta de mediciones de baja potencia, menores a
%                        28 dB-Hz.

s = serial(elport);      %crear objeto serial
set(s, 'BaudRate', velocidad)
set(s, 'DataBits', 8)
set(s, 'InputBufferSize', longitud_buffer);
set(s, 'Parity','none')
set(s, 'StopBits', 1)
set(s, 'Terminator', 'LF/CR')
set(s, 'FlowControl', 'none')
% set(s, 'TimeOut', 5)

fopen(s);
```

```
DataReceive = fread(s);
DataReceiveHex = dec2hex (DataReceive);

DataToSend = ['A0';'A2';'00';'02';'90';'00';'90';'B0';'B3'];
DataDec = hex2dec(DataToSend);
fwrite(s,DataDec,'uint8');
DataRecClock = fread(s);
DataRecClockHex = dec2hex (DataRecClock);

fclose(s);
delete(s);
clear s
nel = numel( DataRecClock);
x = 0;
j=1;
m=25;
i=1;
checksumTramaClock = [];
valorsTramaClock = [];
 for i=m:nel
        var = DataReceive(i);
        if DataRecClock(i) == hex2dec(('A0'))

            if DataRecClock(i+1) == hex2dec(('A2'))

                if DataRecClock(i+2) == hex2dec(('00'))

                    if DataRecClock(i+3) == hex2dec(('14'))

                        if DataRecClock(i+4) == hex2dec(('07'))

                            valorsTramaClock(:,j) = DataRecClock(i + 4: i + 4
+ 19);
                            checksumTramaClock(:,j) = DataRecClock(i + 4 + 20
: i + 4 + 20+ 1);
                             i=i+m;
                             j=j+1;
                        end
                    end
                end
            end
        end
 end
columnas = size(valorsTramaClock,2)
for i=1:20
    for j=1:columnas
     valorsClockHex{i,j} = dec2hex(valorsTramaClock(i,j),2);
    end
end

for t=1:columnas
         a = valorsClockHex(9,:);
         b = valorsClockHex(10,:);
         c = valorsClockHex(11,:);
         d = valorsClockHex(12,:);
         concate = strcat(a,b,c,d); %Falta añadir el Clock drift
         numero = char(concate(1,t));
         Clock = hex2dec(concate);
         Clock_Drift = Clock / 1575420000;
         clear numero concate a b c d

end

nel = numel( DataReceive);
x = 0;
j=1;
m=59;
```

```
i=1;
checksumTrama = [];
valorsTrama = [];
 for i=m:5000
        var = DataReceive(i);
        if DataReceive(i) == hex2dec(('A0'))

            if DataReceive(i+1) == hex2dec(('A2'))

                if DataReceive(i+2) == hex2dec(('00'))

                    if DataReceive(i+3) == hex2dec(('38'))

                        if DataReceive(i+4) == hex2dec(('1C'))

                            valorsTrama(:,j) = DataReceive(i + 4: i + 4 + 55);
                            checksumTrama(:,j) = DataReceive(i + 4 + 56 : i +
4 + 56+ 1);
                             i=i+m;
                             j=j+1;
                        end
                    end
                end
            end
        end
 end

 colum = size(valorsTrama,2)
for i=1:56
    for j=1:colum
     valorsTramaHex{i,j} = dec2hex(valorsTrama(i,j),2);
    end
end
Message_ID = [];
Channel = [];

for t=1:colum
    Message_ID = dec2hex(valorsTrama(1,:));
    Channel = valorsTrama(2,:);
    Satellite_ID = valorsTrama(7,:);
    Sync_Flags = valorsTrama(38,:);
    Phase_Error_Count = valorsTrama(55,:);
    Low_Power_Count = valorsTrama(56,:);
end


for t=1:colum
        a = strcat(valorsTramaHex(3,:));
        b = strcat(valorsTramaHex(4,:));
        c = strcat(valorsTramaHex(5,:));
        d = strcat(valorsTramaHex(6,:));
        concate = strcat(a,b,c,d);
        Time_Tag = hex2dec(concate);
        clear concate a b c d

end

for t=1:colum
         a = valorsTramaHex(24,:);
         b = valorsTramaHex(25,:);
         c = valorsTramaHex(26,:);
         d = valorsTramaHex(27,:);
         concate = strcat(a,b,c,d); %Falta añadir el Clock drift
         numero = char(concate(1,t));
         Carrier_Frequency(t) = bin2ieee754 ([numero]) - Clock_Drift(1,:);
         clear numero concate a b c d

end
```

```
for t=1:colum
    a = valorsTramaHex(8,:);
    b = valorsTramaHex(9,:);
    c = valorsTramaHex(10,:);
    d = valorsTramaHex(11,:);
    e = valorsTramaHex(12,:);
    f = valorsTramaHex(13,:);
    g = valorsTramaHex(14,:);
    h = valorsTramaHex(15,:);
    concate = strcat(e,f,g,h,a,b,c,d);
    numero = char(concate(1,t));
    GPS_Software(t) = bin2ieee754 ([numero]);
    clear numero concate a b c d e f g h

end
%
for t=1:colum
    a = valorsTramaHex(16,:);
    b = valorsTramaHex(17,:);
    c = valorsTramaHex(18,:);
    d = valorsTramaHex(19,:);
    e = valorsTramaHex(20,:);
    f = valorsTramaHex(21,:);
    g = valorsTramaHex(22,:);
    h = valorsTramaHex(23,:);
    concate = strcat(e,f,g,h,a,b,c,d);
    numero = char(concate(1,t));
    Pseudorange(t) = bin2ieee754 ([numero]);
    clear numero concate a b c d e f g h

end

for t=1:colum
        a = strcat(valorsTramaHex(28,:));
        b = strcat(valorsTramaHex(29,:));
        c = strcat(valorsTramaHex(30,:));
        d = strcat(valorsTramaHex(31,:));
        e = strcat(valorsTramaHex(32,:));
        f = strcat(valorsTramaHex(33,:));
        g = strcat(valorsTramaHex(34,:));
        h = strcat(valorsTramaHex(35,:));
        concate = strcat(d,c,b,a,h,g,f,e);
        Carrier_Phase = hex2dec(concate);
        clear concate a b c d e f g h
end

for t=1:colum
    a = strcat(valorsTramaHex(36,:));
    b = strcat(valorsTramaHex(37,:));
    concate = strcat(a,b);
    Time_In_Track = hex2dec(concate);
    clear concate a b

end
%
for t=1:colum
        C_No_1 = hex2dec(valorsTramaHex(39,:));
        C_No_2 = hex2dec(valorsTramaHex(40,:));
        C_No_3 = hex2dec(valorsTramaHex(41,:));
        C_No_4 = hex2dec(valorsTramaHex(42,:));
        C_No_5 = hex2dec(valorsTramaHex(43,:));
        C_No_6 = hex2dec(valorsTramaHex(44,:));
        C_No_7 = hex2dec(valorsTramaHex(45,:));
        C_No_8 = hex2dec(valorsTramaHex(46,:));
        C_No_9 = hex2dec(valorsTramaHex(47,:));
        C_No_10= hex2dec(valorsTramaHex(48,:));
```

```
        tablaCanales =
[C_No_1,C_No_2,C_No_3,C_No_4,C_No_5,C_No_6,C_No_7,C_No_8,C_No_9,C_No_10];

end

for t=1:colum
    a = strcat(valorsTramaHex(49,:));
    b = strcat(valorsTramaHex(50,:));
    concate = strcat(a,b);
    Delta_Range_Interval = hex2dec(concate);
    clear concate a b

end

for t=1:colum
    a = strcat(valorsTramaHex(51,:));
    b = strcat(valorsTramaHex(52,:));
    concate = strcat(a,b);
    Mean_Delta_Range_Time = hex2dec(concate);
    clear concate a b

end

for t=1:colum
    a = strcat(valorsTramaHex(53,:));
    b = strcat(valorsTramaHex(54,:));
    concate = strcat(b,a);
    Extrapolation_Time = hex2dec(concate);
    clear concate a b

end

clear i j t
```

1.6. bin2ieee754.m

```
function salida=bin2ieee754(numhex)
N=length(numhex)*4;
numbin = fliplr(h2b(numhex));
for i=1:N,
    numdec(i)=str2num(numbin(i));
end
switch N
    case 32
        signo=numdec(32);
        exponente=sum(numdec(24:31).*[2.^(0:7)]);
        mantisa=sum(fliplr(numdec(1:23)).*[2.^(-[1:23])]);
        salida=(-1)^signo*2^(exponente-127).*(1+mantisa);
    case 64
        signo=numdec(64);
        exponente=sum(numdec(53:63).*[2.^(0:10)]);
        mantisa=sum(fliplr(numdec(1:52)).*[2.^(-[1:52])]);
        salida=(-1)^signo*2^(exponente-1023).*(1+mantisa);
    otherwise
        salida=0;
end
```

1.7. h2b.m

```
function b2 = h2b(h)
n = length(h);
b2=[];
for indice = 1 : n,
switch h(indice)
    case {'0'}
        b = '0000';
       case {'1'}
        b = '0001';
    case {'2'}
        b = '0010';
    case {'3'}
        b = '0011';
    case {'4'}
        b = '0100';
    case {'5'}
        b = '0101';
    case {'6'}
        b = '0110';
    case {'7'}
        b = '0111';
    case {'8'}
        b = '1000';
    case {'9'}
        b = '1001';
    case {'A', 'a'}
        b = '1010';
    case {'B', 'b'}
        b = '1011';
    case {'C', 'c'}
        b = '1100';
    case {'D', 'd'}
        b = '1101';
    case {'E', 'e'}
        b = '1110';
    case {'F', 'f'}
        b = '1111';
end
b2=[b2 b];
end
```

1.8. c2scaled.m

```
function salida=c2scaled(numhex)
N=length(numhex)*4;
numbin = h2b(numhex)
salida=(tc2dec(numbin,N))
```

1.9. tc2dec.m

```
function value = tc2dec(bin,N)
val = bin2dec(bin);
y = sign(2^(N-1)-val)*(2^(N-1)-abs(2^(N-1)-val));
if ((y == 0) && (val ~= 0))
value = -val;
else
value = y;
end
end
```