INSTITUTO POLITÉCNICO NACIONAL

Escuela Superior de Ingeniería Mecánica y Eléctrica
Unidad Profesional Adolfo López Mateos
Sección de Estudios de Posgrado e Investigación

Receptor GPS utilizando Dispositivos FPGA

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS EN INGENIERÍA DE TELECOMUNICACIONES

P R E S E N T A:

Ing. Juan Manuel Castro Arvizu

DIRECTORES DE LA TESIS:

M. en C. Miguel Sánchez Meraz
Dr. Amadeo José Argüelles Cruz

México D.F. Diciembre 2011

SIP-14

INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D. F. siendo las 11:00 horas del día 5 del mes de Diciembre del 2011 se reunieron los miembros de la Comisión Revisora de Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación de ESIME-ZACATENCO para examinar la tesis titulada:

"RECEPTOR GPS UTILIZANDO DISPOSITIVOS FPGA"

Presentada por el alumno:

CASTRO	ARVIZU	JUAN MANUEL
Apellido paterno	Apellido materno	Nombre(s)

Con registro:

B	0	9	1	8	6	3

aspirante de:

MAESTRO EN CIENCIAS EN INGENIERÍA DE TELECOMUNICACIONES

Después de intercambiar opiniones, los miembros de la Comisión manifestaron *APROBAR LA DEFENSA DE LA TESIS*, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores(a) de tesis

M. en C. MIGUEL SÁNCHEZ MERAZ
Presidente

DR. AMADEO JOSÉ ARGÜELLES CRUZ
Segundo Vocal

DR. MAURO ALBERTO ENCISO AGUILAR
Tercer Vocal

DR. AMADEO JOSÉ ARGÜELLES CRUZ
Secretario

M. en C. MARCO ANTONIO ACEVEDO MOSQUEDA

DR. JORGE ROBERTO SOSA PEDROZA

PRESIDENTE DEL COLEGIO DE PROFESORES

DR. JAIME ROBLES GARCÍA

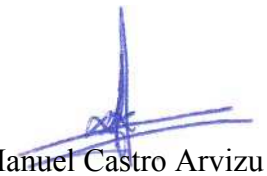
INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México D.F el día 06 del mes Diciembre del año 2011, el que suscribe Juan Manuel Castro Arvizu alumno del Programa de Maestría en Ciencias en Ingeniería de Telecomunicaciones con número de registro B091863, adscrito a la Escuela Superior de Ingeniería Mecánica y Eléctrica unidad Zacatenco, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección del M. en C. Miguel Sánchez Meraz y el Dr. Amadeo José Argüelles Cruz y cede los derechos del trabajo intitulado Receptor GPS utilizando dispositivos FPGA, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección juanma.arvizu@gmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.



Juan Manuel Castro Arvizu

Nombre y firma

Resumen

En este trabajo se presenta la implementación de un receptor GNSS que procesa los diferentes datos de navegación contenidos en la señal GPS. Este prototipo se basa en el uso de una plataforma FPGA integrando los diferentes módulos de un receptor de señales GPS basado en una arquitectura de Radio Definido por Software.

La implementación de los algoritmos sobre dispositivos lógicos programables FPGA es para poder tener un receptor con características programables el cual permita simular otros sistemas de navegación sin necesidad de cambiar dispositivos de hardware sino con solo actualizar algunos parámetros de configuración dentro de la aplicación de software desarrollada.

Abstract

This paper presents the implementation of a GNSS receiver that process the navigation data contained in the GPS signal. This prototype is based using a FPGA Platform integrating all modules included in a GPS Receiver that is based in a Software Defined Radio architecture.

The implementation of algorithms on FPGAs programmable logic devices has the purpose of develop a receiver with programmable characteristics which can implement other navigation systems without changing hardware devices and instead updating only a few settings within the developed software application.

Agradecimientos

A mis padres Juan Manuel y María Aurora por todo el apoyo y confianza que me han brindando en cada peldaño que he subido hasta ahora. Sus enseñanzas siempre las tengo presentes en cada etapa de mi formación como profesionista y persona.

A mis hermanas Cristina y Myriam por estar conmigo en los buenos y malos ratos. Por todos los momentos que pasamos y la dicha que me hace el verlas crecer junto conmigo.

A mis compañeros y maestros que he tenido en *"La Cantera"* por lo que me han enseñado a través de la danza: que la constancia, tenacidad, disciplina y paciencia hacen que uno pueda lograr lo que se proponga.

Agradezco profundamente a mis asesores el M. en C. Miguel Sánchez Meraz y al Dr. Amadeo José Argüelles Cruz por su apoyo, conocimiento y tiempo que invirtieron en mí para obtener mi grado de Maestro en Ciencias.

Al Instituto Politécnico Nacional y CONACYT por todo el apoyo que me brindaron para concluir mi Maestría tanto en el aspecto educativo como en el económico.

Índice General

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1 Antecedentes

El hombre como ser social que es, necesita de la comunicación, ya que de lo contrario se estaría completamente aislado haciendo que la comunicación fuera evolucionando hasta llegar a la más sofisticada tecnología y que el cielo no solo sea habitado con productos del alma como en la antigüedad, sino físicamente con máquinas que impasibles y desde la enorme ventaja que les reporta la altitud en la que se mueven intentan con su funcionamiento hacer la vida del ser humano lo más llevadera posible.

La navegación es el arte de ir de un lugar a otro, de forma segura y de manera eficiente. Cuando se encuentra una tienda dentro de un centro comercial o se camina del trabajo a la casa se están usando las herramientas de los primeros navegadores al usar puntos de referencia. Pero, sí uno se encuentra en medio del océano se deben contar con ciertos artefactos que hagan más fácil la navegación.

Los primeros grandes botes que fueron eficientes para llevar grandes cargas se remonta alrededor del año 3500 A.C. y esto haría el nacimiento de la navegación. Estos primeros navegadores permanecían cerca de la costa con vista a marcas hechas en la tierra o caracteristícas que pudieran reconocer y usualmente sólo viajaban de día. Cuando se quisieron aventurar fuera de la vista de la tierra, era necesario determinar su *Latitud* (posición norte/sur con respecto al Ecuador) observando la altura del sol durante el día y la estrella polar durante la noche.

Una de las primeras herramientas hechas por el hombre para la navegación fue el compás magnético en el siglo XIII que fue usado inicialmente cuando el clima ocultaba al sol o a la estrella polar. El navegador frotaba una aguja de metal contra una magnetita y la colocaba en un corcho dentro de un balde de agua. La aguja apuntaba a la dirección norte. Los primeros marineros encontraban al compás inconsistente ya que no comprendían la diferencia de que apuntaba al polo norte mangético y no al norte verdadero (esto es llamado variación) por

lo que no pudieron explicar estas variaciones y no tuvieron confianza en las lecturas cuando navegaban en áreas desconocidas.

Los marineros al mismo tiempo usaron el astrolabio (creado en el año 1484 por Martin Behaim) para medir el ángulo con respecto al horizonte del sol y las estrellas para medir la *latitud*. También servía para medir la *altitud* del sol o las estrellas. La *Altitud* es la distancia vertical a un origen determinado, considerado como nivel cero.

La primera representación exacta de la superficie esférica de la tierra fue la Proyección Mercator (Gerardus Mercator, 1569). Un gran valor para los navegadores ya que la orientación de la brújula era mostrada como una línea recta (y así podían ver la distancia más corta entre dos puntos), pero el problema de determinar la longitud retardó el uso de estas proyecciones cartográficas por casi 70 años después de haber sido introducidas.

La *Longitud* se define como que tan lejos del este u oeste se encuentra un objeto localizado. La clave para determinar la longitud fue la invención de un sistema de custodía del tiempo exacto. Ya se sabía que la tierra era un globo y que rotaba una revolución completa al sol cada 24 horas. Los navegadores sabían que el sol alcanzaba su máxima altitud al medio día sin importar en que parte de la tierra se estuviese. Si ellos pudiesen determinar a que tiempo exacto era en la longitud de 0°, ellos podrían fácilmente calcular la longitud de su posición actual por la diferencia entre los dos tiempos (una hora equivale a 15° de longitud).

Este método fue tan importante que varios paises ofrecieron ofertas de dinero por la invención de un cronómetro exacto. John Harrison en 1764 fue el que ganó la oferta británica por su cronómetro maritímo. James Cook usó el cronómetro de Harrison para dar la vuelta al mundo y cuando volvió en 1779, sus cálculos de longitud fueron correctos a 8 millas que con ayuda de un científico complementó los cálculos hechos por Harrison y detalló los mapas terrestres durante su viaje que cambió el curso de la navegación y sus mapas fueron usados alrededor del mundo.

En 1884, por acuerdo internacional, el meridiano de Greenwich, Inglaterra fue adoptado como el Primer Meridiano (0°).

El siglo XX ha visto grandes avances en herramientas de navegación. El ímpetu por estos desarrollos ya no fue más por los viajes y la exploración sino para el uso en la guerra. Sin embargo, muchos de estos instrumentos y tecnologías han sido adaptados para usos pacíficos.

El científico británico Robert Watson-Watt produjó el primero radar práctico (Radio Detection And Ranging en sus siglas en inglés). Es usado para localizar objetos dentro de un rango de visión proyectando ondas de radio contra él. El Radar puede determinar la precencia, el rango de un objeto, su posición en el espacio, sus dimensiones, su forma, su velocidad

y su dirección.

El sistema de navegación hiperbólica conocido como LORAN (Long Range Navigation) fue desarrollando en los Estados Unidos entre 1940 y 1943. LORAN es un sistema de ayuda a la navegación electrónico que utiliza el intervalo transcurrido entre la recepción de señales de radio transmitidas desde tres o más transmisores para determinar la posición del receptor. La versión más moderna es LORAN-C funciona en frecuencias del espectro electromagnético entre los 90 y 100 Khz.

No fue sino hasta principios de los 60 que el Departamento de Defensa, Departamento de transporte y la Agencia Espacial Norteamericanas (DoD, DoT y NASA respectivamente) tomaron interés en desarrollar un sistema para determinar la posición basado en satélites.

El sistema debía cumplir los requisitos de globalidad, abarcando toda la superficie del planeta; continuidad, con funcionamiento constante sin ser afectado por las condiciones atmosféricas; altamente dinámico, para posibilitar su uso en aviación.

El sistema TRANSIT también conocido como NAVSAT (Navy Navigation Satellite System con sus siglas en inglés) fue el primero en su especie. Los TRANSIT fueron una serie de satélites de navegación de los Estados Unidos. El primer TRANSIT (primer satélite de navegación del mundo), fue lanzado en abril de 1960 y estaba constituido por una constelación de seis satélites en órbita polar baja, a una altura de 1,074 km. Tal configuración conseguía una cobertura mundial pero no constante. La posibilidad de posicionarse era intermitente, pudiéndose acceder a los satélites cada 1.5 horas. Para poder calcular la posición se requiere engancharse al satélite durante quince minutos continuamente [1].

Pero aun así TRANSIT tenía muchos problemas y en esa época la URSS contaba con su propio sistema llamado TSICADA los norte americanos buscaban la forma de crear un sistema efectivo que dejara a la URSS atrás. Por lo que se concibió un sistema formado por 24 satélites en orbita media, con capacidad de cobertura global y continua.

Figura 1.1: *Satélite TRANSIT.*

El primer satélite se lanzó en 1978, y se planeó el funcionamiento total en ocho años hasta que en diciembre de 1983 se declaró la fase operativa inicial del Sistema de Posicionamiento Global (GPS, en sus siglas en inglés).

El objetivo del sistema GPS era ofrecer a las fuerzas de los E.E.U.U. la posibilidad de ubicarse (disponer de la posición geográfica) de forma autónoma o individual, de vehículos o de armamento, con un costo relativamente bajo, con disponibilidad global y sin restricciones temporales. La iniciativa, financiación y explotación corrieron a cargo del Departamento de Defensa de los Estados Unidos (DoD) y el Sistema de Posicionamiento Global (GPS) se concibió como un sistema militar estratégico [2] .

En 1984, el vuelo civil de Korean Airlines fue derribado por la Unión Soviética al invadir por error su espacio aéreo. Por éste motivo la administración Reagan se vío en la necesidad de ofrecer a los usuarios civiles cierto nivel de uso del sistema GPS, llegando finalmente a ceder el uso global y sin restricciones temporales, de esta forma se conseguía un retorno a la economía de los Estados Unidos, inimaginables unos años atrás [2].

Desde 1984, con muy pocos satélites en órbita, aparecieron tímidamente fabricantes de receptores GPS destinados al mundo civil (Texas Instruments, Trimble Navigation y Motorola Semiconductors).

Estos satélites transmiten señales con espectro en microondas, así permitiendo a un receptor GPS determinar su localización, velocidad, dirección y tiempo. Este sistema fue desarrollado por el DoD y nombrado como NAVSTAR GPS. El Sr. Juan Walsh fue el encargado de dirigir éste gran sistema [3].

El sistema GPS consiste en tres segmentos:

1 Espacial.

2 Control.

3 Usuario.

La figura 1.2 muestra un dibujo de los segmentos del Sistema GPS.

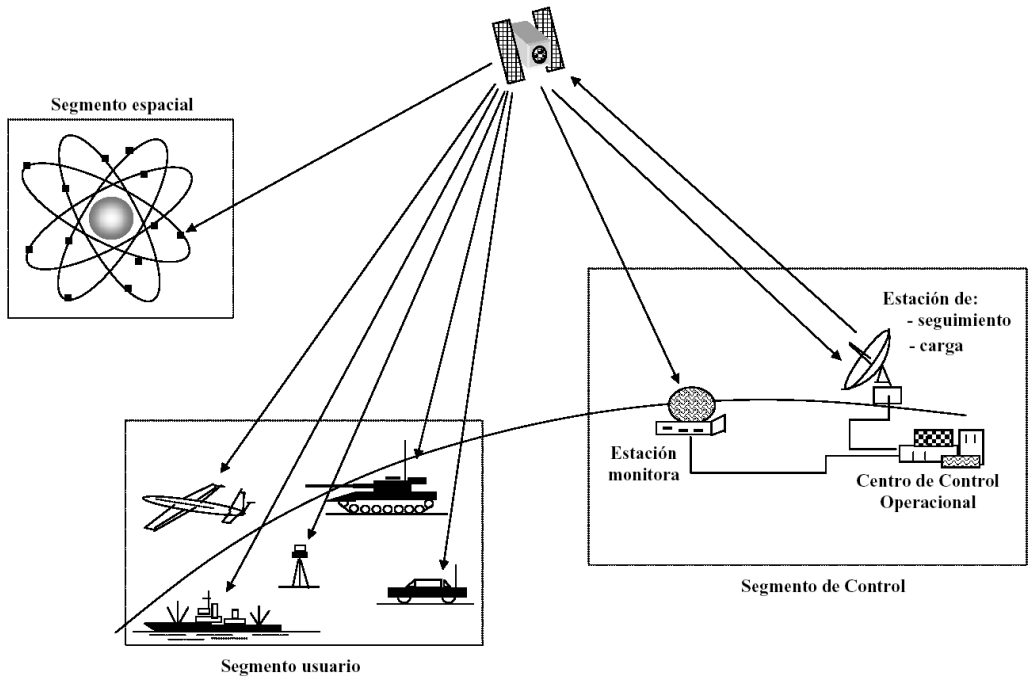


Figura 1.2: *Segmentos del Sistema GPS.*

El Segmento del Espacio se compone de:

⋆ 24 Satélites en 6 planos orbitales

⋆ 4 satélites en cada plano

⋆ 20.180 Km. de altura.

⋆ 60 grados de inclinación.

La figura 1.3 muestra un dibujo de los segmentos del Sistema GPS.

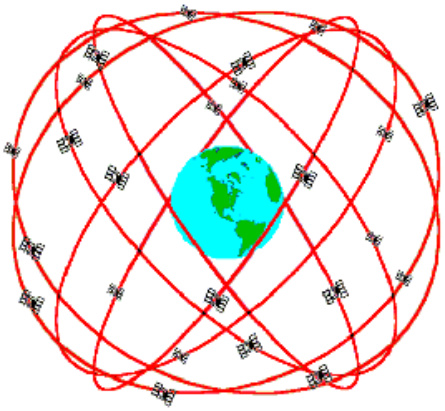


Figura 1.3: *Constelación GPS.*

Normalmente, hay un número determinado de satélites para reemplazar satélites viejos o dañados y se encuentran en órbitas de emergencia. Estos satélites tienen una vida media aproximada de siete años y medio.

Los satélites están situados a 20,180 Km. de altura desplazándose a una velocidad de 14.500 km/h. Las órbitas son casi circulares y se repite el mismo recorrido sobre la superficie terrestre (mientras la tierra gira a su vez sobre su eje) de esta forma se tiene en promedio el valor de un día (aproximadamente 24 horas menos cuatro minutos) un satélite vuelve a pasar sobre el mismo punto de la tierra. Los satélites quedan situados sobre seis planos orbítales (con un mínimo de cuatro satélites cada uno), espaciados equidistantes a 60 grados e inclinados unos 15 grados con respecto al plano ecuatorial. Esta disposición permite que desde cualquier punto de la superficie terrestre sean visibles entre cinco y ocho satélites [3].

El *segmento de control*, consiste en un sistema estaciones de seguimiento localizadas alrededor del mundo. La estación maestra de control (MCS) está situada en Falcon AFB en Colorado Springs. Las estaciones de control miden las señales procedentes de los satélites y son incorporadas en modelos orbitales para cada satélite. Los modelos calculan datos de ajuste de órbita (efemérides) y correcciones de los relojes de cada satélite. La estación maestra envía las efemérides y correcciones de reloj a cada satélite. Cada satélite envía posteriormente subconjuntos de estas informaciones a los receptores de GPS en forma de mensaje de navegación [4].

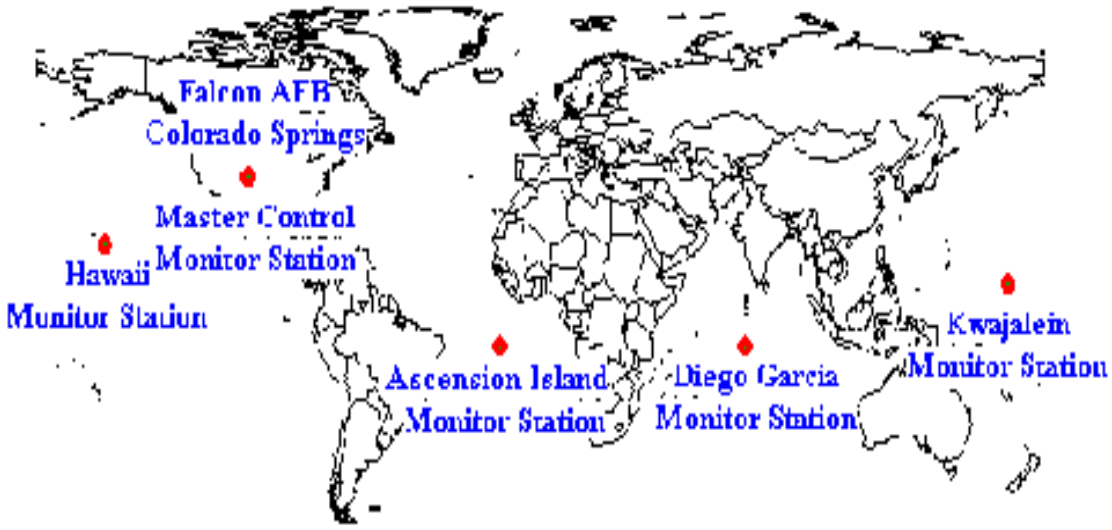


Figura 1.4: *Estaciones de Control Alrededor del Mundo.*

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

- ⋆ La navegación en tres dimensiones, es la función principal del GPS. Se construyen receptores GPS para aeroplanos, embarcaciones, vehículos terrestres y equipos portátiles de pequeño tamaño.
- ⋆ El posicionamiento preciso es posible usando receptores en posiciones de referencia proporcionando datos de corrección y posicionamiento relativo a receptores remotos. Vigilancia, control geodésico y estudios de las placas tectónicas, son ejemplos.
- ⋆ Las aplicaciones de tiempo y estabilización de frecuencia se basan en la precisión de los relojes que incorporan los satélites y que son monitorizados continuamente por las estaciones de control. Los satélites actuales incorporan cuatro relojes atómicos, dos de Rubidio y otros dos de Cesio que ofrecen una estabilidad de frecuencia equivalente a un error de un segundo en 30,000 años. (Hay que tener en cuenta que un error de 30ns provoca un error de 30cm.). Los observatorios astronómicos, sistemas de telecomunicaciones, sincronización de centrales eléctricas y laboratorios de certificación,

pueden obtener señales de tiempo y frecuencia de alta precisión mediante receptores especiales de GPS. Las señales de GPS han sido utilizadas para medir parámetros atmosféricos [5].

Un receptor del GPS calcula simplemente la distancia entre sí mismo y los satélites. Con la ayuda de un reloj atómico, el satélite transmite continuamente ciertos datos que contienen su hora exacta, la localización de las órbitas satelitales y su almacenamiento de trayectoria. El receptor GPS entonces mide el tiempo de la recepción de la señal. De ésta forma se obtiene el cálculo de la distancia entre los satélites y el receptor GPS. Obteniendo la información de cuatro distancias, se forma un tetraedro, obteniendo la posición de usuario en 3D por la intercepción de cuatro esferas. El cuarto satélite, funciona como referencia en la variable del tiempo.

La capacidad del GPS de calcular la velocidad y la orientación locales, es extremadamente útil. La transferencia del tiempo es posible debido a su capacidad de sincronizar el reloj. Un ejemplo extensamente usado del uso del GPS es el teléfono dígital de la célula de CDMA (Acceso Múltiple por División de Código). Cada base, utiliza un receptor GPS para sincronizar a los usuarios con diversas estaciones base para apoyar así llamadas telefónicas de emergencia y otros muchos usos. El equipo del GPS también ha revolucionado el campo de la tectónica midiendo el movimiento de las placas durante terremotos [4].

1.2 Planteamiento del Problema

Una de las diferencias notables de esta tesis con los receptores GPS que se encuentran en el mercado es la reconfigurabilidad. Esto quiere decir, que para cada necesidad del usuario se puede acoplar un desplegado de mensajes de navegación diferentes o adicionales a estos. Los modelos que se encuentran ahora en el mercado, despliegan un mensaje de navegación definido por el fabricante, pero en el caso de esta tesis, el programador, al cambiar la palabra de control del código fuente, lleva a la lectura un mensaje de navegación que sea de interés particular para el usuario haciéndolo un receptor GPS propio para cada tarea específica.

Cabe señalar que actualmente, aprovechando la característica de que las señales GNSS comparten similtudes unas a otras, es posible emigrar de un sistema de navegación a otro desde un mismo receptor sin necesidad de cambiar algun elemento en hardware. Esto hace evidente que un receptor que se diseñe, tenga la capacidad de ser configurable para trabajar bajo cualquier sistema de navegación ya sea GPS como Galileo o GLONASS.

Es por eso que esta tesis da una apertura a una amplia gama de trabajo e investigaciones futuras donde se involucre el apartado de posicionamiento y al hacerlo programable facilitará la realización de estos.

1.3 Justificación

El sistema de navegación europeo Galileo va a contribuir a la cobertura mundial prevista por GNSS en términos de nevegación, posición y aplicaciones de tiempo. Mirando a un futuro cercano, Galileo proveerá un gran conjunto de señales, las cuales dependerán de los servicios a los que sean asignados.

No sólo mejorará el número total de señales en el espacio (SIS), sino también dará a los usuarios nuevas capacidades como integridad en sistemas, mejorando la calidad de las señales en un entorno urbano. Señales de banda ancha como la Portadora Binaria con Offset (BOC) y la alternativa BOC (AltBOC) serán lanzadas alrededor del mundo, haciendo a los usuarios capaces de procesar diferentes señales moduladas de alto rendimiento. Por otro lado, GPS va a introducir nuevas señales civiles para proveer a los usuarios mejor rendimiento en términos de posicionamiento [6] .

En este escenario complejo, es claro que el diseño de los receptores GNSS se está conduciendo a la realización de avanzados algoritmos de procesamiento de señales, aún si el impacto de la implementación en hardware tiene que ser tomada en consideración. Como es posible encontrar en la literatura, una gran variedad de soluciones para hardware están disponibles para el desarrollo de alguna plataforma de prueba, donde la flexibilidad juega un papel muy importante en el diseño al igual que los aspectos de potencia.

Un FPGA (del inglés *Field Programmable Gate Array*) es un dispositivo semiconductor que contiene bloques de lógica cuya interconexión y funcionalidad puede ser configurada mediante un lenguaje de descripción especializado. La lógica programable puede reproducir desde funciones tan sencillas como las llevadas a cabo por una puerta lógica o un sistema combinacional hasta complejos sistemas en un chip (SoC).

Las FPGAs tienen las ventajas, a comparación de los ASICs (Circuitos Integrados de Aplicación Específica), de ser reprogramables (lo que añade una enorme flexibilidad al flujo de diseño), sus costes de desarrollo y adquisición son mucho menores para pequeñas cantidades de dispositivos y el tiempo de desarrollo es también menor [6]. Un FPGA permite el manejo de altas tasas de datos, mientras el DSP (Procesador dígital de señales) provee la capacidad del procesamiento de señales y la computación matemática.

Un receptor GNSS debe ofrecer a cualquier usuario la localización de posición y navegeación en cualquier sitio y a cualquier hora. A su vez, dependiendo de la aplicación que se requiera, el receptor tiene que ser flexible, adaptable y reconfigurable para cualquier tipo de señal GNSS ya que debe manejar altas tasas de datos y tener una enorme capacidad para la computación matemática de los algoritmos de navegación y el procesamiento digital de la señal.

Bajo estas consideraciones la elección de un FPGA gestionado por un DSP se considera como una buena elección debido a su peculiaridades ya mencionadas. Al mismo tiempo, hace que la técnica de Radio Definido por Software sea la arquitectura base para el receptor GNSS a desarrollar en este trabajo de tesis.

1.4 Objetivo

Desarrollar un sistema receptor GPS capaz de decodificar los mensajes de altitud, longitud, latitud y tiempo, basado en el uso de dispositivos FPGA.

1.5 Objetivos Específicos

Los alcances esperados una vez concluida esta investigación son:

1 Decodificación de la señal GNSS de los satélites recibidos para recuperar sus mensajes de altitud, longitud, latitud y tiempo.

2 Desarrollo de un algoritmo para el cálculo de la posición del receptor GNSS a partir de los mensajes recuperados desde los satélites.

3 Desarrollar un receptor GNSS sobre un sistema de desarrollo de dispositivos FPGA.

4 Desplegar los mensajes de posición en una pantalla de cristal líquido.

1.6 Organización de la Tesis

En el primer capítulo se explica la introducción al sistema GPS, sus antecedentes y la necesidad del hombre de conocer su posición geográfica así como el objetivo, planteamiento del problema y la justificación.

En el segundo capítulo se menciona el fundamento teórico de los diferentes bloques que conforman al sistema receptor GPS.

El tercer capítulo corresponde específicamente a la metodología utilizada para el cálculo de posición.

El cuarto capítulo describe al diseño en Hardware del receptor GPS utilizando la tarjeta Virtex 6 ML605.

En el quinto capítulo presenta las pruebas y resultados.

El sexto capítulo entrega las conclusiones de este trabajo de tesis, de igual manera que el trabajo a futuro.

Capítulo 2

Fundamentos Teóricos

2.1 Radio Definido por Software

Un radio es cualquier tipo de dispositivo que transmite en forma inalámbrica o recibe señales en la frecuencia de radio (RF) de parte del espectro electromagnético para facilitar la transferencia de información. En el día de hoy, existen radios en una multitud de elementos tales como teléfonos celulares, computadoras, aparatos para abrir la puerta del coche, vehículos y televisores.

A su vez, el procesamiento moderno digital de señales cada vez más se afirma en los últimos diseños de sistemas radio-electrónicos como receptores y transmisores de radio. No sólo moduladores y demoduladores analógicos o digitales, sino también los amplificadores de frecuencia intermedia se pueden realizar completamente en forma digital. Este progreso se debe al alto rendimiento de los convertidores analógico-digital y digital-analógico, pero principalmente por el incremento de la potencia computacional de los circuitos digitales programables y los procesadores digitales de señales. La última tecnología digital permite el procesamiento de señales con un ancho de banda en el rango de diez a miles de MHz. Los convertidores analógicos-digital modernos permiten procesar señales en frecuencias intermedias estándar de 20 a 140 MHZ. [7]

El receptor o transmisor, donde gran parte del procesamiento de la señal se realiza digitalmente usando circuitos programables digitales, es llamado un SDR (Radio definido por software en sus siglas en inglés) o llamado en forma corta Radio Software. La arquitectura del radio definido por software (SDR) está ganando popularidad debido a la capacidad de reprogramación a trabajar bajo estándares diferentes a los cambios mínimos de hardware [8].

Los circuitos digitales realizan el siguiente procesamiento de señal: Filtrado, sincronización, detección, ecualización, modulación, control de error FEC, etc. El procesamiento subsequente puede también ser programado en los circuitos digitales.

El hardware tradicional basado en sistemas de radio sólo puede ser modificado a través de intervención física. Esto se traduce en altos costos de producción y poca flexibilidad para soportar diferentes estándares de frecuencias. Por el contrario, la tecnología de Radio Definido por Software proporciona una solución eficiente y de bajo costo para este problema, permitiendo que sistemas completos puedan ser mejorados por medio de actualizaciones en software [9] .

2.1.1 Características de SDR

A continuación se presentan las principales características de la tecnología SDR

- **Reconfigurabilidad.** El SDR permite la coexistencia de distintos módulos de software adecuado para ejecutar diferentes estándares en un mismo sistema permitiendo una configuración dinámica del sistema con sólo seleccionar el módulo de software adecuado para correr. La tecnología SDR facilita la aplicación en servicios múltiples, multi-modo y multi-banda y terminales multi-estándar.

- **Interoperabilidad.** La arquitectura SDR facilita el uso de sistemas abiertos en arquitectura de radio. En el campo de telefonía móvil, los usuarios finales pueden utilizar sin problemas aplicaciones innovadoras de otros fabricantes en sus teléfonos móviles como en un sistema de PC. Ésto aumenta el atractivo y la utilidad de los teléfonos. La arquitectura SDR hace que un sistema pueda emigrar de una aplicación específica a otra sin necesidad de cambios en hardware.

La arquitectura SDR define una colección de tecnologías en hardware y software donde algunas o todas las funciones de operación de radio (también conocidas como procesamiento de la capa física) se ejectuan a través de software modificable. Estos dispositivos incluyen Arreglos en Compuertas Programables en Campo (dispositivos FPGA) , Procesadores Digital de Señales (DSP), Procesadores de Propósito General (GPP, cuyas siglas provienen del inglés *General Purpose Processor*), Sistema Programable en Chip (SoC, cuyas siglas provienen del inglés *System-on-Chip*) u otras aplicaciones específicas de procesadores programables. El uso de estas tecnologías permite nuevas funciones inalámbricas y capacidades que se añaden a los sistemas de radio ya existentes sin necesidad de nuevo hardware.

2.1.2 Arquitectura de receptor GNSS basado en Software

La arquitectura general de un receptor GNSS (Sistema Global de Navegación por Satélite, GNSS en su acrónimo inglés) basado en software se muestra en la figura 2.1. Después de la sección RF, la cual incluye la antena y el *front end* de RF, la señal análoga se manda a un

convertidor analógico-digital (ADC), el cual convierte la señal al dominio digital. Luego, las etapas de adquisición, seguimiento y posicionamiento posteriores, permiten obtener la señal que provee la solución de Posición, Velocidad y Tiempo (PVT).

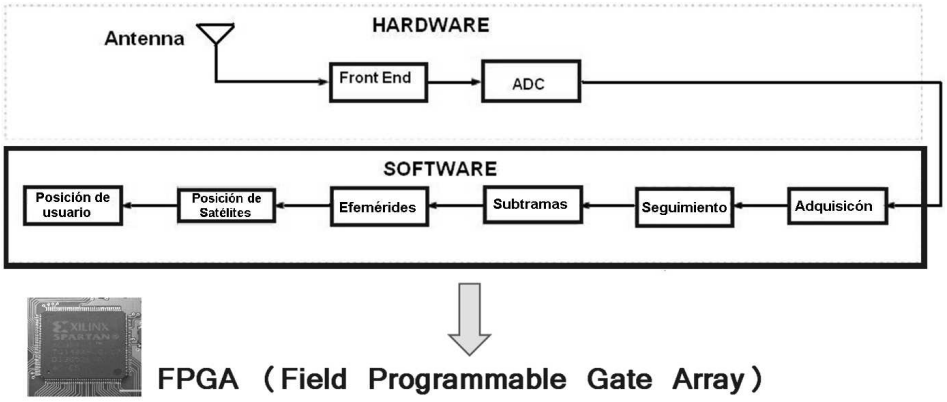


Figura 2.1: *Arquitectura general de un receptor GNSS.*

La línea de separación entre un *front-end* (FE) analógico y digital (AFE y DFE respectivamente) debe ser trazada para clasificar las posibles arquitecturas que se muestran en la figura 2.2:

- Receptor completamente basado en Hardware, en el cual todo el procesamiento digital de señales es realizado vía secciones no reconfigurables (HW).
- Receptor controlado por Software, en el cual, el procesamiento de señales es realizado vía hardware pero es controlado vía software.
- SDR, en el cual, todo el procesamiento es realizado utilizando total o parcialmente dispositivos re-programables después de un front-end en hardware seguido por un ADC.
- Receptor "ideal" en software, en el cual el ADC es colocado justo después de la antena y del amplificador de bajo ruido (LNA), lo que implica que todo el procesamiento sea realizado vía software [10] .

Hablando en general, en un radio SDR, todas las funciones de procesamiento de señales son realizadas vía Hardware digital en lugar de Hardware analógico. Este enfoque provee reducciones en costos, la versatilidad se incrementa, y se reduce el tiempo para ser lanzado al mercado, esto debido a que disminuye el tiempo necesitado en el diseño, en compilación y en pruebas. Gracias a la filosofía SDR, una terminal SDR puede permitir la coexistencia de varios módulos en software, implementando diferentes estándares de comunicación y funciones (multi-modo, multifunciones, terminales multi-servicio) [11]. A diferencia de radio controlado

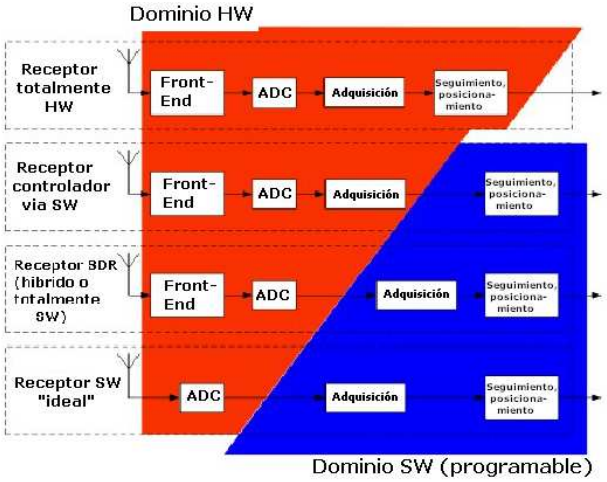


Figura 2.2: *Diferentes niveles de partición en HW/SW de los receptores GNSS*

vía software (terminales operadas sobre circuitos integrados de aplicación específica, ASICs), las terminales SDR tienen la característica clave de reconfigurabilidad.

Como se puede observar en la figura 2.2, las familias de plataformas en Hardware que actualmente pueden ser usadas para un receptor GNSS en arquitectura SDR se pueden clasificar en dos grupos:

- Arquitectura totalmente basada en Software, y
- Arquitecura híbrida Hardware/Software.

Las arquitecturas totalmente basada en Software son usadas cuando un alto grado de flexibilidad es requerida. Por ejemplo, las soluciones basadas en PC son particularmente adecuadas para fines de investigación; los sistemas embebidos direccionan al mercado hacia sistemas portátiles.

Las arquitecturas basadas en FPGA (Arquitectura híbrida HW/SW) combinan tamaño y eficiencia energética. De hecho, ofrecen una reconfiguración completa algorítmica y arquitectónica, allanando el camino para una entrada importante de la tecnología SDR en el mercado de masas [12].

Las investigaciones recientes en arquitectura basada en SDR se deben a la búsqueda de ventajas económicas, lo que implica baja complejidad y arquitecturas de bajo consumo, capaces de ofrecer un rendimiento mejorado a precios asequibles.

Cuando se trata de soluciones de radio basadas en una arquitectura híbrida, una metodología de diseño, conocida como Arquitectura de Comunicaciones en Software (SCA en sus siglas en inglés), es a menudo adoptada. La SCA es una estructura abierta de arquitectura en Software que sugiere a los diseñadores como los elementos en hardware y software deben operar en armonía dentro de un SDR, lo que permite un pleno uso de la flexibilidad del SDR. La principal característica de la SCA es la capacidad de proveer interfaces directamente transferibles a los componentes de radio, garantizando así una reutilización muy amplia para los componentes de Software [7] .

El uso de un receptor GNSS embebido en un solo microprocesador (o un procesador de señal digital, DSP) se está convirtiendo en un método habitual en el mercado de masas, aparentemente impulsado por el mercado de aplicaciones móviles. Por otra parte, se espera que los receptores de software, estén disponibles en las plataformas basadas en PC en un futuro próximo. De igual manera, las señales de un ancho de banda grande en las bandas de L1/l2 ya pueden ser transferidas y procesadas en tiempo real en la PC. El desarrollo del USB de triple frecuencia para *front-ends* para GPS y Galileo están también actualmente en desarrollo[13]. Por otra parte, los receptores integrados en Software de gama alta definitivamente superan a las soluciones en hardware en términos de costos de desarrollo y de integración, aunque el consumo de energía sigue siendo una cuestión que queda abierta.

Para evaluar mejor los parámetros críticos de un diseño adecuado SDR para receptores GNSS, es útil examinar las cuestiones de implementación que ya se han enfrentado en proyectos pasados [14] [5]. Diversas actividades para los receptores GNSS-SDR se han identificado en todo el mundo, con diferentes opciones de arquitectura. Las terminales SDR más utilizadas son aquellas que dependen de tablas personalizadas con soluciones de interfaz específicas para la etapa de conectar la RF a las secciones digitales. Sin embargo, también se han aprovechado las características disponibles de los diseños de interfaz estándar (PCI, SCA, PC/104) para conectar el RF y/o las tarjetas digitales [7] .

También, la elección entre un DSP y procesadores de propósito general (GPPs) depende de la arquitectura seleccionada. Mientras los DSP se montan típicamente en chips específicos en los tableros digitales personalizados, los GPPS generados por software permiten que el procesador seleccionado sea sintetizado en uno de los FPGAs disponibles.

Las consideraciones previas sobre las aplicaciones GNSS permiten muchas oportunidades de negocios para la tecnología de los GNSS en un futuro a corto y mediano plazo, particularmente cuando la arquitectura SDR es utilizada para las terminales de usuario y receptores. Hoy en día la tecnología SDR no es suficiente para cubrir todas las aplicaciones posibles. Sin embargo, existen perspectivas prometedoras para la integración de los servicios GNSS (tiempo, posicionamiento, navegación) y otras tecnologías con arquitectura SDR en un futuro próximo.

La adopción de la filosofía SDR como la tecnología central en receptores GNSS proveerá las características de modularidad y flexibilidad, también, puede allanar el camino para el estudio y la realización de la próxima generación de receptores integrados multi-estándar y de navegación/comunicación (NAV/COM) [10] .

2.2 Estructura de la señal GPS

Las señales GPS son transmitidas en dos radiofrecuencias en la banda UHF. La banda UHF cubre la banda de frecuencias de 300 MHz a 3 GHz. Estas frecuencias son conocidas como L1 y L2 y son derivadas de una frecuencia en común,$f_0 = 10.23MHz$

$$f_{L1} = 154 f_0 = 1575.42\ MHz$$
$$f_{L2} = 120 f_0 = 1227.60\ MHz$$

Las señales están compuestas de las siguientes tres partes:

Portadora: La frecuencia de la portadora.

Datos de navegación: Los datos de navegación contienen información con respecto a las órbitas satelitales. Esta información es actualizada desde todos los satélites a las estaciones terrestres en el Segmento de Control GPS. Los datos de navegación se transmiten a una tasa de transmisión de 50 bps.

Secuencia dispersa: Cada satélite tiene dos únicas secuencias dispersas o códigos. El primero es el código de adquisición aproximado (C/A), y el otro es el código de precisión encriptado (P(Y)). El código C/A es una secuencia de 1023 pps (número de pulsos por segundo). El código es repetido cada milisegundo, logrando una frecuencia de 1.023 MHz. El código P es un código más largo ($\approx 2.35x10^4$ pps) con una tasa de 10.23 MHz. Este se repite cada semana comenzando a principios de cada semana GPS la cual empieza la noche del sábado para domingo. El código C/A es solamente modulado sobre la portadora L1 mientras el código P(Y) es modulado tanto con la portadora L1 como con la portadora L2.

La señal C/A se modula con una portadora con una frecuencia de 1575.42 MHz (la frecuencia L1) usando BPSK (Modulación por desplazamiento en fase, con sus siglas en inglés) para luego ser mandada a través del canal de comunicación.

La señal de entrada en el lado del receptor para el *k-ésimo* satélite se puede representar como:

$$S^{(k)} = A^k C^k(t-\tau) D^k(t-\tau) cos[2\pi(f_{L1} + f_D^k)t + \theta^k] + n(t) \quad (2.1)$$

donde:

A	es la amplitud de la señal L1 de entrada.
C	es la señal de código C/A *"gold code"* .
$D(t)$	es la señal con los datos de navegación.
τ	es el retardo en tiempo.
f_{L1}	es la frecuencia de la portadora L1.
f_D	es el efecto Doppler.
θ	es el cambio de fase de la portadora.
$n(t)$	es el término para el ruido.

La portadora L_2 se modula en bifase por código P (normalmente) o por código C/A. Los mismos datos de navegación a 50 bps pueden ir con L_2 al igual que con L_1.

$$s_{L2}^{(k)}(t) = A^{(k)} C^{(k)}(t-\tau) D^{(k)}(t-\tau)\, cos\left[2\pi(f_{L2} + f_D^{(k)})t + \theta^{(k)}\right] + n(t)$$

Satélite k — Amplitud de L2 — Código P del satélite k

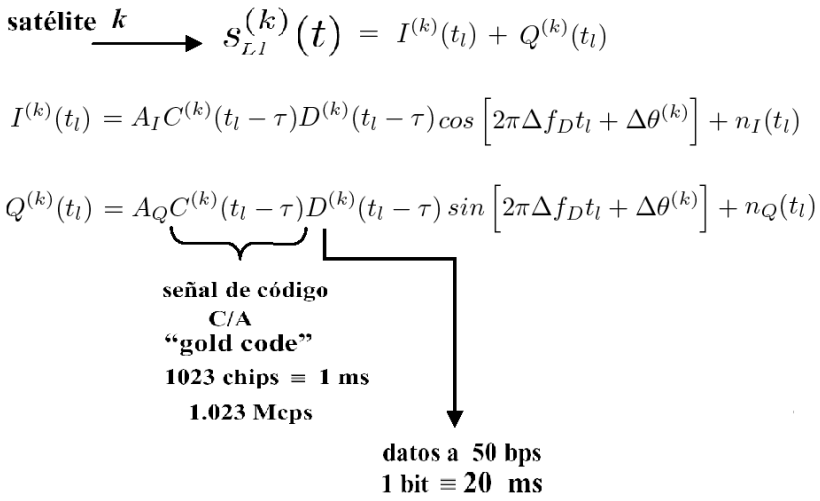
A veces, L2 se modula con código P y sin datos de navegación, lo cual mejora las prestaciones de los sistemas de -seguimiento de la señal- en recepción, mejorando la caracterítica ruido/interferencia. Propiamente, ésta modulación es utilizada en aplicaciones militares [4].

La señal es procesada en el *front-end* del receptor. Esto incluye las etapas de amplificación, mezclado con frecuencia de un oscilador local y filtrado. La señal se convierte en,

$$S_{IF}^{(t)} = A_{IF}^k C^k(t-\tau) D^k(t-\tau) cos[2\pi(f_{IF} + f_D^k)t + \delta\theta^k] + n_{IF}(t) \quad (2.2)$$

donde:
A_{IF} es la amplitud luego de la etapa IF.
f_{IF} es la frecuencia IF,
$\delta\theta$es la diferencia entre la fase de la portadora recibida y la fase de la portadora IF,
n_{IF} es el ruido luego de la etapa de IF.

La señal IF es muestreada y multiplicada por las formas de onda de referencia coseno y seno para remover la componente de la portadora creando así las componentes *En-Fase* (I) y *Cuadratura* (Q),



donde
$\Delta f_D = f_d - \hat{f}_d$, es la diferencia entre la frecuencia doppler de entrada y la generada internamente,
$\Delta\theta = \delta\theta - \hat{\theta}$ es la diferencia entre la fase de la portadora de entrada y la generada internamente.

Para la adquisición de la señal, se correlaciona la señal de entrada con una versión generada en el receptor del código C/A (o PRN) hasta que un pico sea detectado. Un Bucle Cerrado de Retardo (DLL) es utilizado para alinear el código de fase mientras un Bucle Cerrado de Fase (PLL) se usa para alinear la fase de la portadora. Una vez que la señal de un satélite específico ha sido determinada, entonces se calcula la fase de código y el efecto Doppler para luego ser usados en el seguimiento de la señal.

2.3 Módulos de un receptor GNSS

2.3.1 Antena

La antena será diseñada para inducir un voltaje de las ondas de radio que se propagan en la frecuencia L1 GNSS o a 1575.42 MHz. Además, el diseño debe acomodar el ancho de banda apropiado a la señal deseada. Esto usualmente se especifica al utilizar dos parámetros de antenas adicionales; *Relación de Onda Estacionaria de Voltaje* (VSWR, en sus siglas en inglés) y la *impedancia*. Prácticamente todos los componentes del *front-end* GNSS tienen una impedancia de 50 Ω, la cual es la impedancia típica para la mayoría de los diseños en radio-frecuencia. El VSWR es una medida de cuanta de la potencia incidente será absorbida y cuanta sera reflejada. Y, por supuesto, esta es una función de la frecuencia. El VSWR se encuentra en el orden de 2.0:1, lo cual equivale a la absorción de energía del 90% en todo el

ancho de banda de frecuencias deseadas.

La señal GPS es RHCP(*Rigth-Hand Circularly Polarized*, en sus siglas en inglés), así que la antena del receptor debe ser también RHCP para compaginar con la señal de entrada. El patrón de la antena es esencialmente hemisférico en la mayoría de las aplicaciones.

El patrón de la antena describe la directividad de la antena. La idea básica para el patrón de una antena sería que recibiera señales en todas direcciones (que se conoce por antena isotrópica). Sin embargo, tal patrón de ganancia uniforme no tiene sentido para GNSS. Ya que la fuente de la señal, los satélites GNSS, están por encima de todos los posibles receptores. El patrón de antena preferido debería ser hemisférico, diseñado para recibir señales en ángulos positivos de elevación en todas las direcciones azimutales. Esto permitirá el seguimiento de los satélites desde el zenit hasta el horizonte (típicamente 90° - 5°). Dado el problema de multitrayecto y que la mayoría de los rayos multitrayecto llegan desde ángulos de elevación bajos, el patrón de la antena debe diseñarse para recibir señales solamente por encima de los 10-20 grados de elevación [15].

Existe una gran variedad de antenas; la más común es la antena de parche. Una visión completa de antenas de superficie se describe en [16].

2.3.2 Front-End RF

La señal capturada por la antena se amplifica usando un amplificador de bajo ruido (LNA), luego es filtrada con un filtro pasa banda para eliminar el ruido. Los filtros SAW (*Surface Acoustic wave*), son ampliamente utilizados en receptores actuales [10]. La señal es luego convertida a una IF por medio de un mezclador RF-IF. Este proceso es necesario para convertir la señal analógica a una digital. La portadora de la señal GNSS L1 es 1.57542 Ghz, y necesita ser disminuida a algunas decenas o unos pocos cientos de MHz para poder utilizar chips ADC comerciales. Éste es el fin del proceso de la conversión de bajada.

El ancho de banda en la última etapa de filtro IF se encuentra en el rango de los 2 a los 20 MHz en los receptores de alto desempeño.

2.3.3 Convertidor Analógico-Digital

El convertidor A/D transformará la señal analógica IF a una señal digital para el procesamiento en banda base. Los receptores de alto desempeño utilizan hasta 3 bits de muestreo para anchos de banda de 2-20 MHz. La degradación de la señal debido a los niveles de cuantización finitos de bit depende de dos factores junto con el número limitado de bits. Primero es el ancho de banda IF. Segundo, la razón del umbral máximo con respecto al nivel de ruido RMS [17]. Una vez que la señal es digitalizada entra a la fase del procesamiento digital de la

señal.

El objeto del conjunto de componentes dentro del front-end es acondicionar el voltaje incidente en la antena para que pueda ser muestreado por el ADC. Con el fin de lograr esto para la mayoría de los ADCs, hay tres funciones básicas que deben ser realizadas. Éstas son *amplificación, traslado de frecuencia/conversión de bajada* y *filtro*. Éstos bloques preparan a la señal para la conversión de analógico a digital, lo cual resultará en las muestras a ser procesadas en la parte de software.

Un esquema completo de un *front-end* GNSS L1 se muestra en la figura 2.3 donde se exhiben cada uno de los bloques descritos anteriormente,

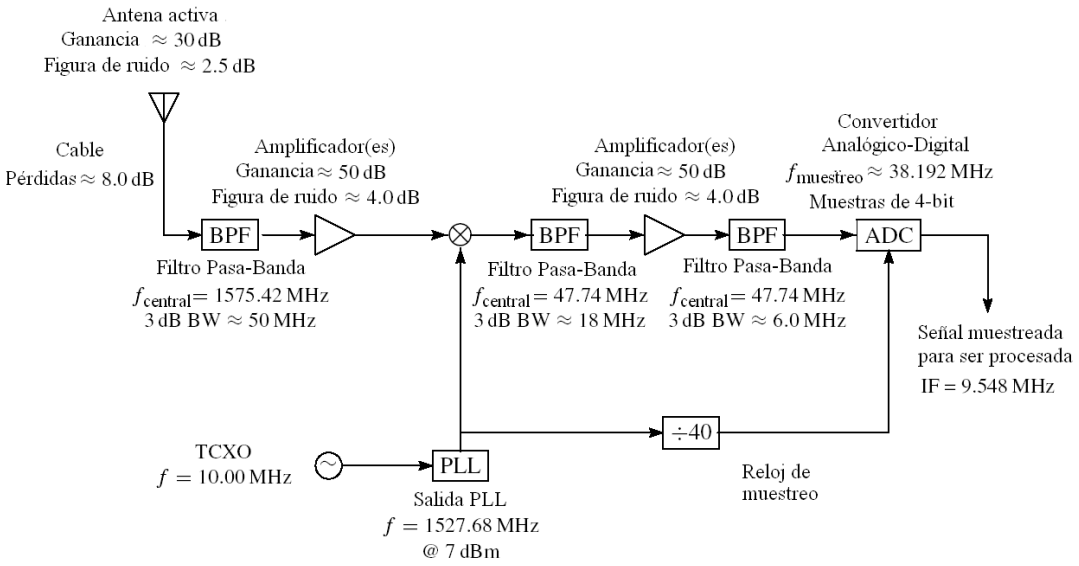


Figura 2.3: *Front-End GNSS L1.*

2.4 Procesamiento Digital de la Señal

2.4.1 Adquisición de la Señal

El propósito de la etapa de adquisición es encontrar los satélites visibles y los valores gruesos de la portadora al igual que la fase del código de esas señales. Los satélites se diferencian por 32 secuencias PRN. La fase del código es el tiempo de alineación del código PRN en el bloque de datos entrante con el que se genera internamente en el receptor. Es necesario la fase del código para poder generar un PRN local que esté perfectamente alineado con el código de entrada. Sólo cuando suceda esto, el código de entrada puede ser removido de la señal.

Si el satélite es visible, la adquisición determinará las siguientes dos propiedades de la señal:

Frecuencia:
La frecuencia de la señal de un satélite puede diferir de su valor nominal. En el caso de conversión de bajada, la frecuencia nominal de la señal GPS en L1 corresponde a la IF. Sin embargo, las señales se ven afectadas por el movimiento relativo del satélite causando un efecto Doppler [11].

Fase del código. La fase del código denota el punto en el bloque de datos actual donde comienza el código C/A. Sí un bloque de datos de 1 ms es examinado, el dato incluye todo un código C/A y por lo tanto el principio del código C/A.

Muchos métodos diferentes han sido usados, pero todos ellos están de una manera u otra basados sobre las propiedades de la señal GPS. Especialmente las propiedades de correlación del código C/A son las más importantes.

La señal recibida s es una combinación de señales de todos los n satélites visibles:

$$s = s^1(t) + s^2(t) + ... + s^n(t) \quad (2.3)$$

Cuando es adquirido un satélite k, la señal entrante s es multiplicada con el código C/A generado localmente correspondiente al satélite k.

La correlación cruzada entre códigos C/A para diferentes satélites implica que las señales de otros satélites son casi eliminadas por este procedimiento. Para evitar eliminar el componente de las señal deseada, el código C/A generado localmente debe ser correctamente alineado en tiempo, este se hace con la fase del código correcta.

Después de la multiplicación con el código generado localmente, la señal será mezclada con una onda portadora generada localmente. Esto es hecho para eliminar la onda portadora de la señal recibida. Para eliminar la onda portadora de la señal, la frecuencia de la señal generada localmente debe ser lo más cercana posible a la frecuencia de la señal portadora. Luego de la mezcla, todos los componentes de la señal son elevados al cuadrado y sumados, proporcionando un valor numérico.

El efecto Doppler es un parámetro muy importante que debe ser considerado. Debido a la velocidad del satélite, hay un efecto Doppler que resulta en valores de frecuencia más altos o más bajos en la señal entrante. En el peor de los casos, la frecuencia se puede desviar $\pm$ 10kHz de la frecuencia nominal. Para identificar sí un satélite es o no visible, basta con buscar valores de frecuencias en rangos de 500 Hz resultando en 41 frecuencias diferentes en el caso

de receptores de movimiento rápido y 21 en el caso de receptores estáticos [4]. Es importante conocer la frecuencia de la señal para poder generar una portadora de señal local. En la etapa de adquisición son usadas la correlación circular y la Transformada Rápida de Fourier (FFT).

La etapa de adquisición funciona como una etapa de búsqueda. Para cada una de las diferentes frecuencias, se prueban 1023 diferentes códigos de fase. Luego de probar todas las posibilidades para el código de fase y frecuencia, se busca un valor máximo. Sí el valor máximo excede un determinado umbral, el satélite es adquirido con su frecuencia y fase correspondiente. La figura 2.4 muestra un gráfico típico para un satélite visible. La figura 2.4 muestra que las señales generadas desde el PRN 21 están presentes en la señal recibida. Esto se ve en el pico del gráfico que viene relacionado con el código de fase C/A y la frecuencia de la señal.

Existen dos modelos para la etapa de adquisición de la señal que se implementan en receptores GPS: Adquisición de Búsqueda Serial y Adquisición de Búsqueda Paralela [4].

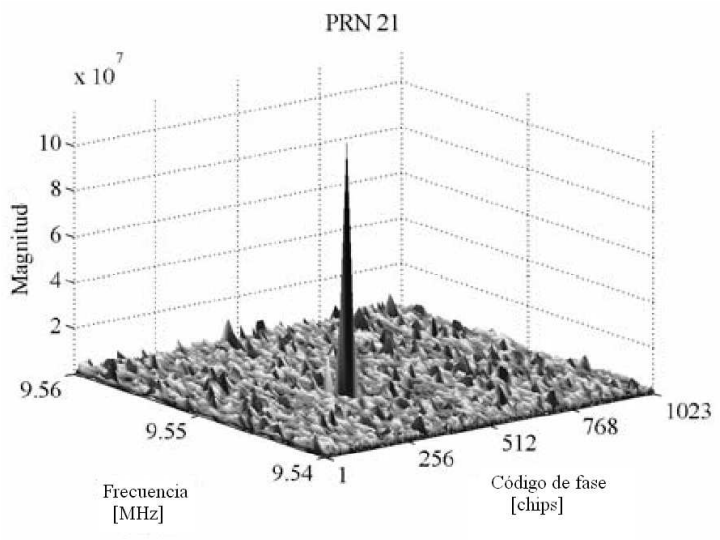


Figura 2.4: *Gráfica de adquisición. Las señales generadas desde el PRN 21 están presentes en la señal recibida.*

2.4.2 Seguimiento

El propósito principal del seguimiento es refinar los valores gruesos del código de fase y la frecuencia para mantener un seguimiento de como las propiedades de la señal cambian a través del tiempo.

La exactitud del valor final del código de fase está conectada a la exactitud con la cual se calculará la pseudodistancia posteriormente. El bloque de seguimiento contiene dos partes: El seguimiento de código y seguimiento de portadora/código

Seguimiento de Código
El seguimiento de código se implementa como un Bucle de Retardo Cerrado (DLL) donde tres códigos locales (réplicas) son generadas y correlacionadas con la señal de entrada. Éstas tres réplicas se refieren a las replicas *early, prompt* y *late*. Éstos tres códigos están a menudo separadas por medio chip.

Seguimiento de Portadora / Código
El otro bloque es el seguimiento de la portadora. Este proceso se puede realizar de dos maneras: ya sea por el seguimiento de la fase de la señal o por el seguimiento en la frecuencia.

Para demodular los datos de navegación exitosamente hay que generar una réplica de la portadora. Para el seguimiento de la portadora, se utiliza un Bucle de Fase Cerrado (PLL) o un Bucle de Frecuencia Cerrado (FLL). Un diagrama que muestra el bloque de seguimiento se observa en la figura 2.5.

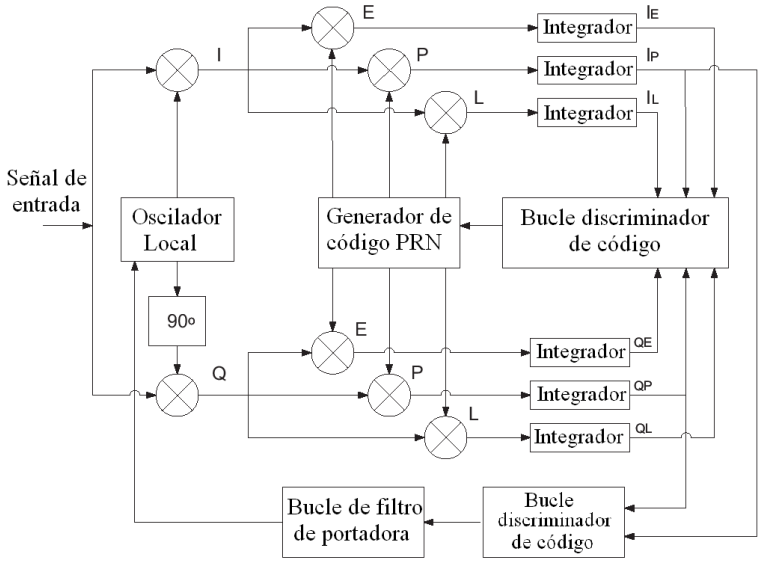


Figura 2.5: *Implementación del ciclo de seguimiento.*

2.4.3 Extracción de los datos de navegación.

Luego del bloque de seguimiento, el código C/A y la portadora pueden ser removidos de la señal, dejando sólamente los bits de datos de navegación. Luego de leer alrededor de 30 segundos de datos, el comienzo de una subtrama se debe encontrar en el orden del tiempo de cuando los datos fueron transmitidos desde el satélite.

Cuando se encuentra el tiempo de transmisión, se decodifican las efemérides de cada uno de los satélites a la vista. Estos datos se utilizan luego para calcular la posición del satélite en el tiempo de transmisión.

El último proceso antes de realizar el cálculo de posición es obtener las pseudodistancias con cada uno de los satélites. La pseudodistancia es la distancia medida desde cada uno de los satélites hacia el receptor. Las pseudodistancias se calculan en base al tiempo de transmisión desde el satélite y el tiempo en que llega al receptor. El tiempo de llegada se basa en el comienzo de cada subtrama.

2.4.4 Cálculo de posición

La tarea final del receptor es calcular la posición de usuario. La posición se calcula en base a las pseudodistancias y a las posiciones de lo satélites calculadas de las efemérides. La figura 2.6 muestra una idea del método para el cálculo de posición. Conociendo SVN_k y la distancia de viaje ρ_k de la señal, se calcula la posición de usuario. El capítulo 3 da una descripción detallada para el algoritmo computacional.

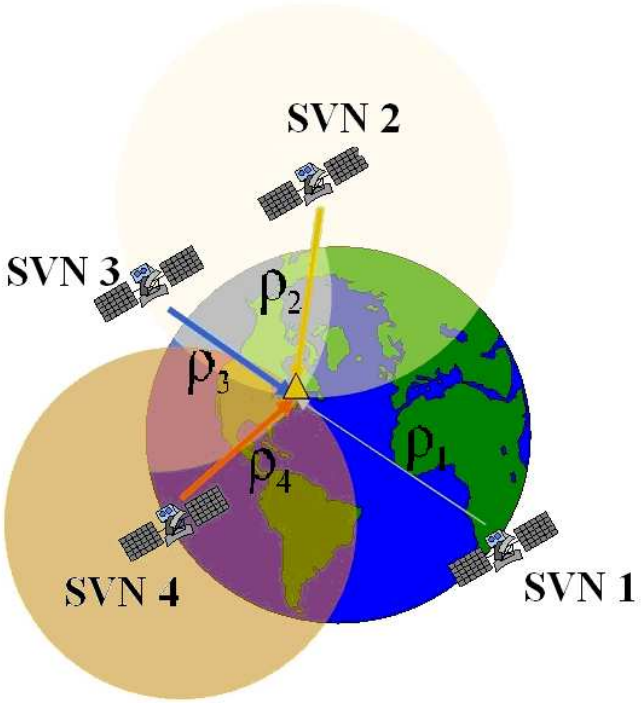


Figura 2.6: *Principio básico para el cálculo de posición.*

Capítulo 3

Extracción de mensajes de navegación

3.1 Datos de navegación

Los datos de navegación son transmitidos sobre la frecuencia L1. Esta sección describe la estructura y el contenido de los datos de navegación. La figura 3.1 muestra la estructura general de un mensaje de navegación completo.

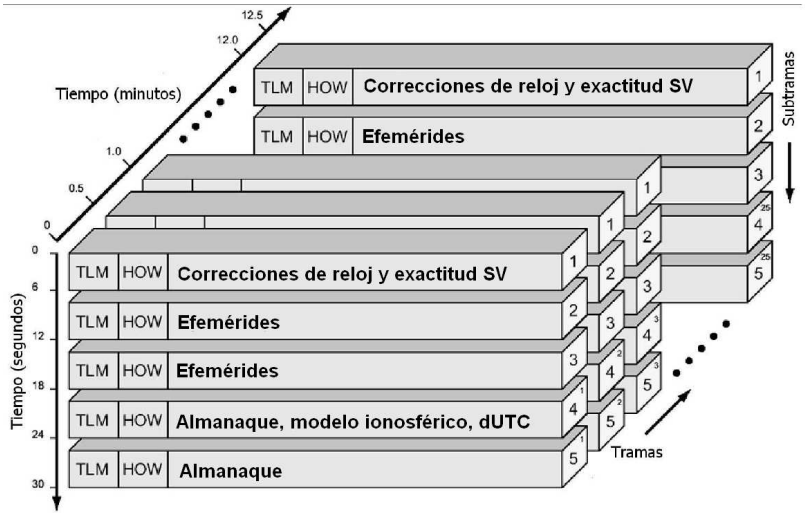


Figura 3.1: *Estructura de los datos de navegación GPS.*

La longitud de una trama es de 1500 bits que contiene 5 subtramas las cuales, tienen 300 bits de longitud cada una. Una subtrama contiene 10 palabras, donde cada palabra tiene 30 bits. Las subtramas 1, 2 y 3 se repiten en cada trama. Las últimas subtramas, 4 y 5, tienen 25 versiones (que tienen la misma estructura pero diferentes datos) y se refieren como las páginas 1 a 25, así que una transmisión completa de mensajes de datos requiere de 25 tramas

completas. La transmisión de una subtrama dura 6 segundos, una trama dura 30 segundos y un mensaje de navegación entero dura 12.5 minutos.

3.1.1 Palabras de Telemetría y Hand Over

Las subtramas comienzan siempre con dos palabras especiales: la de telemetría (TLM) y la de Hand Over (HOW).

La palabra TLM es la primera palabra de cada subtrama y se repite cada 6 segundos. Contiene un preámbulo de 8 bits seguido por 16 bits reservados y de paridad. El preámbulo debe ser usado para la sincronización de la trama.

La palabra HOW contiene una versión truncada de 17 bits del *Time Of Week* (TOW). Los siguientes tres bits indican el ID (identificador) de la subtrama que indica en cual de las 5 subtramas se encuentre el TOW actual.

3.1.2 Datos en el Mensaje de Navegación

Subtrama 1. Reloj de Satélite y Datos de Salud.

La primera subtrama contiene toda la información de reloj que es necesaria para calcular a qué tiempo el mensaje de navegación es transmitido desde el satélite. También, contiene los datos de salud que indican si los datos son de confianza.

Subtrama 2 y 3. Efemérides del Satélite.

Estas subtramas contienen todos los datos de las efemérides de la órbita del satélite necesarias para calcular la posición del satélite.

Subtrama 4 y 5. Datos de Apoyo.

Como se mencionó anteriormente, las últimas subtramas se repiten cada 12.5 minutos, dando un total de 50 subtramas. Las subtramas 4 y 5 contienen los datos de almanaque. Los datos de almanaque son las efemérides y relojes con precisión reducida. En adición, cada satélite transmite datos de almanaque para todos los satélites GPS cuando sólo transmite datos de efemérides de sí mismo. El resto de las subtramas 4 y 5 contienen varios datos por ejemplo parámetros UTC (Tiempo Universal Coordinado en su acrónimo en inglés), indicadores de exactitud y parámetros ionosféricos.

3.2 Decodificación de los datos de navegación

La codificación de los datos de navegación siguen un esquema definido en [18]. Cuando los bits de navegación son obtenidos a través del bit de sincronización, deben ser decodificados. Los parámetros para las efemérides GPS se describen en esta sección.

3.2.1 Localización de la palabra del Preámbulo

El primer problema en la decodificación de los datos de navegación es determinar el inicio de una subtrama. El comienzo de una subtrama está marcada por un preámbulo de longitud de 8 bits. El patrón de una subtrama es *1 0 0 0 1 0 1 1*. El procedimiento de autentificación verifica si el mismo preámbulo es repetido cada 6 segundos que es el tiempo que corresponde a la transmisión de dos subtramas consecutivas.

La búsqueda del preámbulo se implementa a través de la correlación. La primera entrada a la función de correlación es la secuencia de entrada de los bits con los datos de navegación. La segunda entrada es el preámbulo de 8 bits. Ambas entradas son representadas con -1's y 1's. Cuando se usan los valores de -1 y 1 en lugar de 0 y 1, la salida de la función de correlación es 8 cuando el preámbulo es localizado y -8 cuando existe un preámbulo invertido como se puede observar en la figura 3.2.

Como se observa en la figura 3.2, la función de correlación da una máxima correlación de 8 en diferentes tiempos. En este ejemplo, se tienen seis valores máximos de correlación, por consecuencia debe tener seis subtramas. También se tienen valores máximos con signo negativo, esto es, porque se ha localizado un preámbulo invertido. El método para distinguir cual de los valores máximos corresponde realmente al inicio de una subtrama y no una coincidencia de bits es midiendo el retraso entre dos valores consecutivos. Solamente si el retraso entre dos valores es exactamente 6 segundos y el chequeo de paridad no falla, se indica así que es el inicio de una subtrama.

Cuando se localizan los preámbulos correctos, los datos de cada subtrama pueden ser extraídos. Si la correlación muestra que los preámbulos están invertidos, toda la subtrama debe ser invertida.

Debido al efecto Doppler, la longitud del bit de navegación se puede desviar del valor exacto 20 ms. Sobre ésta pequeña diferencia de tiempo, puede acumular así algún valor significante. El algoritmo permanece igual pero cada bit en el preámbulo de referencia es convertido a 20 valores. Ahora el pico de correlación tendrá un máximo valor de 8x20=160 en lugar de 8. Simultáneamente, este algoritmo modificado también encuentra el tiempo de transición de bit [15] .

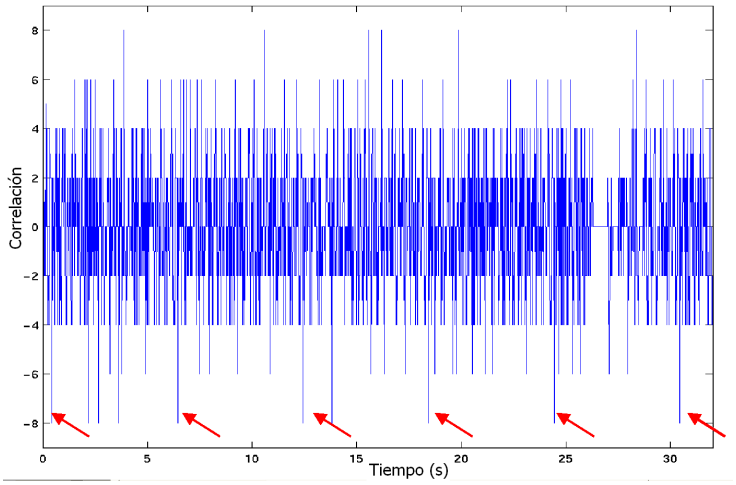


Figura 3.2: *Correlación entre 33 segundos de bits de datos de navegación y 8 bits de preámbulo*

3.2.2 Extracción de los datos de navegación

Cada preámbulo correcto marca el inicio de una subtrama con los datos de navegación. Cada una de las subtramas contiene 300 bits divididos en 10 palabras de 30 bits cada una. La estructura de las dos primeras palabras de una subtrama se muestra en la figura 3.3.

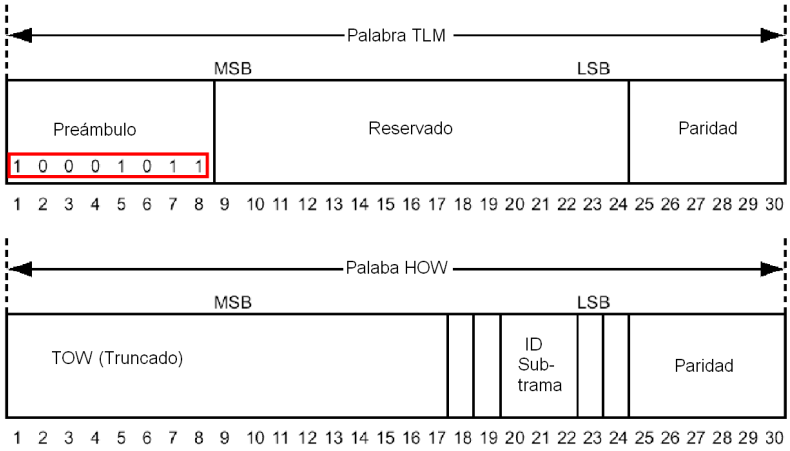


Figura 3.3: *Las primeras dos palabras de cada subtrama correspondientes a la palabra de telemetría (TLM) y la palabra Hand Over (HOW).*

En la figura 3.4 se observa un ejemplo de una subtrama (subtrama 3) y los datos de navegación correspondientes a éste [18] . Este mismo formato lo comparte el resto de las

subtramas. Los parámetros de efemérides son decodificadas de acuerdo a la tabla 3.1 y respetando el factor de escala. Las unidades de semicírculo se multiplican por π para ser convertidas en unidades de radianes.

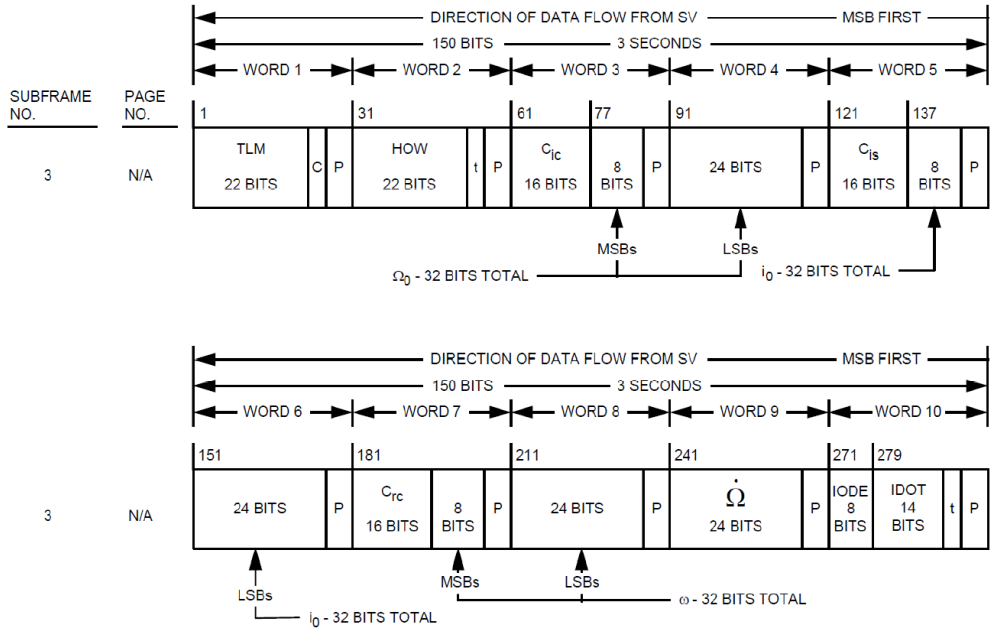


Figura 3.4: *Subtrama 3 con las efemérides contenidas en ella.*

El parámetro IODE es un número de ocho bits que unicamente identifica el conjunto de datos. Todos los parámetros son listados en la tabla 3.2 y explicados en la siguiente sección.

Tabla 3.1: *Esquema de decodificación de los parámetros de efemérides GPS. n* significa que los n bits actuales deben ser decodificados utilizando el complementos a dos.*

Parámetro	**No. de bits**	**factor de Escala (LSB)**	**Unidades**
IODE	8		
C_{rs}	16*	2^{-5}	metros
Δ_n	16*	2^{-43}	semicírculos/seg
M_0	32*	2^{-31}	semi-círculos
C_{uc}	16	2^{-29}	radianes
e	32*	2^{-33}	adimensionales
C_{us}	16*	2^{-29}	radianes
$(A)^{\frac{1}{2}}$	32*	2^{-19}	$metros^{\frac{1}{2}}$
t_{oe}	16	2^4	segundos
C_{ic}	16*	2^{-29}	radianes
$(OMEGA)_0$	32*	2^{-31}	semi-círculos
C_{is}	16*	2^{-29}	semi-círculos
$i0$	32*	2^{-31}	radianes
C_{rc}	16*	2^{-5}	metros
w	32*	2^{-31}	semi-círculos
$OMEGADOT$	24*	2^{-43}	semi-círculos/seg
$IDOT$	14*	2^{-43}	semi-círculos/seg

Tabla 3.2: *Parámetros de efemérides.*

IODE	Conjunto de datos, efemérides
Δn	Correción del movimiento medio
ν_0	Anomalía media a t_{oe}
e	Excentricidad
$\sqrt{a}$	Raíz cuadrada del eje semi mayor
t_{oe}	Tiempo de referencia de las efemérides
Ω_0	Longitud del nodo ascendente a t_{oe}
i_0	Inclinación a t_{oe}
ω	Argumento del perigeo
$\dot{\Omega}$	Tasa de Ω_0
$\dot{i}$	Tasa de i
C_{rs}, C_{rc}	Coeficientes de corrección para términos seno y coseno de r
C_{is}, C_{ic}	Coeficientes de corrección para términos seno y coseno de i
C_{us}, C_{uc}	Coeficientes de corrección para términos seno y coseno de *omega*

3.3 Cálculo de la posición de Satélite.

Esta sección conecta las coordenadas ECEF (Earth-centered and Earth-fixed) X,Y,Z a la posición del satélite descritas por los elementos de órbita keplerianos. Hay que referirse a los elementos orbitales *a,e,w,* Ω*, i y* μ mostradas en la figura 3.5. Los seis elementos orbitales keplerianos constituyen una importante descripción de la órbita, así que se describen de forma esquemática en la tabla 3.3.

El eje X apunta hacia el punto Aries como punto de referencia debido a que en órbitas espaciales, se debe tener un punto de referencia para calcular las mediciones. Debido al movimiento mismo de la tierra e igual nuestro movimiento con respecto al sol, estos elementos se descartan por lo que se recurre a constelaciones estelares que se encuentran a millones de años luz y por lo cual, se pueden considerar "fijas" a nuestro punto de vista. El eje Z coincide con el eje de giro de la Tierra. El eje Y es ortogonal a estas dos direcciones formando así un eje coordenado en tres dimensiones.

El plano orbital intersecta al ecuador de la Tierra por la *línea de nodos*. La dirección en la cual el satélite se mueve del sur al norte se llama *Nodo Ascendente K*. El ángulo entre el plano ecuatorial y el plano orbital es la *inclinación i*. El ángulo en el centro de la Tierra entre el eje X y el nodo ascendente K se llama Ω; es en ascención derecha. El ángulo en el centro de la Tierra entre K y el perigeo P se llama *argumento del perigeo w*; se incrementa en dirección contraria a las manecillas del reloj.

Debido al achatamiento de la tierra, se producen dos rotaciones del plano orbital [4]:

- Modificación del Ω. Conocido como regresión de nodos ya que los nodos parecen que se deslizan a lo largo del ecuador, y eso hace que la línea de nodos, la cual está en el plano ecuatorial, gire alrededor del centro de la tierra.

- El segundo efecto es la rotación de ápsides (eje mayor y eje menor de la órbita satelital) en el plano orbital, lo que por consecuencia se tiene una modificación en el valor del argumento del perigeo, ω.

Estos efectos dependen del movimiento medio n , el eje semimayor a y la excentricidad e, valores que vienen actualizados dentro de las efemérides del satélite.

Tabla 3.3: *Elementos de la órbita keplerieana: Posición del satélite.*

a e	eje semi-mayor excentricidad	Forma y tamaño de la órbita
ω Ω i	argumento del perigeo Nodo ascendente de ascenso recto inclinación	Plano orbital en el aparente sistema
μ	Anomalía media	Posición en el plano

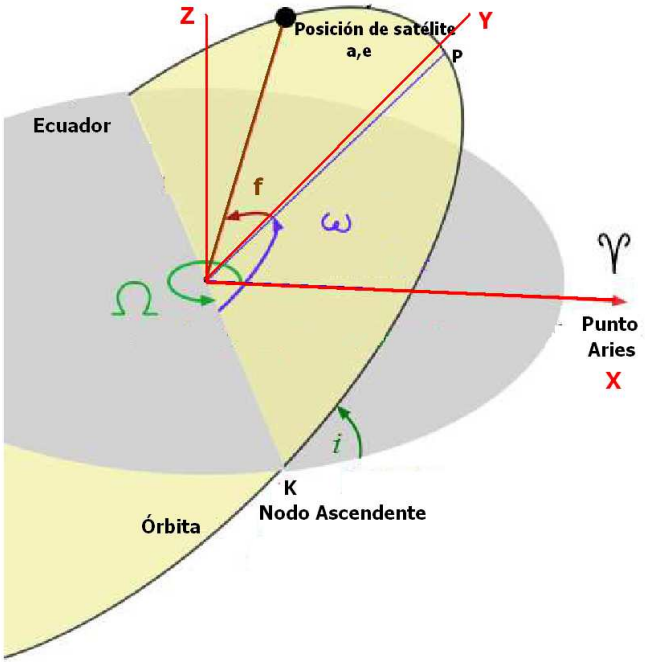


Figura 3.5: *Los elementos de la órbita kepleriana.*

La figura 3.6 muestra un sistema coordenado en el plano orbital con origen en el centro de la Tierra C. El eje ξ apunta al perigeo y el eje η hacia el nodo descendente. El eje ξ es perpendicular al plano orbital. De la figura 3.6 se lee la anomalía excentrica E y la anomalía verdadera ν y por consecuencia se obtiene del gráfico:

$$\xi = rcos\nu = acosE - ae = a(cosE - e)\eta = rsen\nu = \frac{b}{a}asenE = bsenE = a\sqrt{a - e^2}senE \tag{3.1}$$

Así, la posición del vector **r** del satélite con respecto al centro de la Tierra C es:

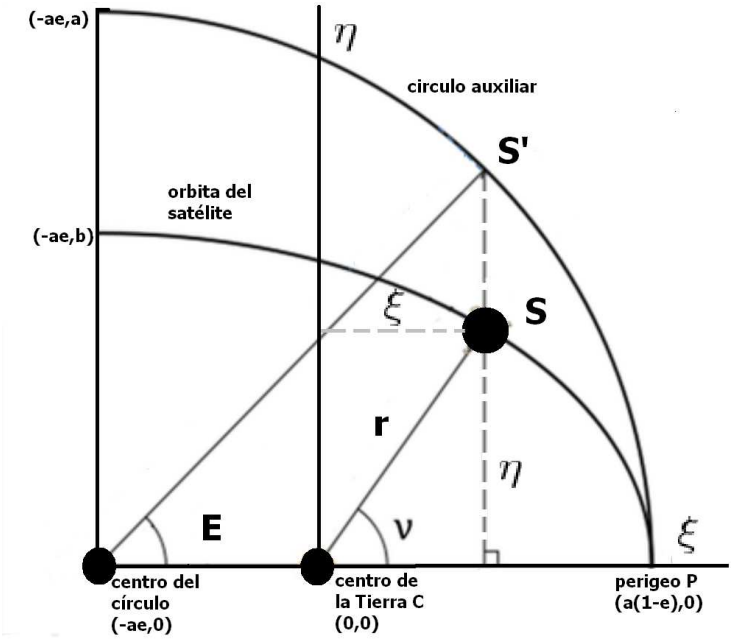


Figura 3.6: *La órbita eliptica con coordenadas (ξ , η). La anomalía verdadera υ en C.*

$$\mathbf{r} = \xi\eta\zeta = aa(cosE - e)a\sqrt{1-e^2}senE0 \tag{3.2}$$

Usando simple trigonometría se obtiene la norma de **r**:

$$\|\mathbf{r}\| = a(1 - ecosE) \tag{3.3}$$

En general, E varía con el tiempo t mientras a y e son casi constantes. Hay que recordar que $\|\mathbf{r}\|$ es la distancia geométrica entre el satélite S y el centro de la Tierra C=(0,0).

El movimiento medio n es la velocidad angular promedio del satélite. Si el período de una revolución del satélite es T, entonces:

$$n = \frac{T}{2\pi} = \sqrt{\frac{GM}{a^3}} \tag{3.4}$$

El producto GM es una la constante gravitacional WGS-84[1] y tiene el valor de $3.986005 \times 10^{14} m^3/s^2$.

Ahora sea t_0 el tiempo en que el satélite pasa por el perigeo, por lo tanto $\mu(t) = n(t - t_0)$. La famosa ecuación de Kepler relaciona la anomalía media μ y la anomalía excentríca E.

$$E = \mu + esenE \tag{3.5}$$

Del vector de posicion $\mathbf{r}$, finalmente se obtiene:

$$\nu = arctan\frac{\nu}{\xi} = arctan\frac{\sqrt{1-e^2}senE}{cosE - e} \tag{3.6}$$

Así, se tiene conectada la anomalía verdadera ν, la anomalía excentrica E y la anomalía media μ. Estas relaciones son básicas para el cálculo de posición de cada satélite. La forma para la solución de esta ecuación es por medio de iteraciones.

Los parámetros elegidos para la descripción de la órbita actual de un satélite GPS y sus perturbaciones son similares a los elementos orbitales keplerianos. Éstos parámetros están a un tiempo específico. Cada satélite transmite sus datos de efemérides únicos. Éstas efemérides se utilizan también para predecir la siguiente parte de la órbita. Las efemérides tienen una exactitud de 1-2 metros. Para aplicaciones geodésicas, se necesita una mejor aproximación a valores reales. Una posibilidad es obtener *efemérides precisas* pre-procesadas, las cuales tienen una exactitud en niveles de *dm* [15] .

El uso destinado de las efemérides va desde el tiempo de referencia t_{oe} que se cuenta en segundos de la semana GPS [2]. Las efemérides están destinadas para ser usadas durante este período. Sin embargo, describen a la órbita dentro de la exactitud especificada 2 horas después [4]. Las efemérides incluyen los parámetros mostrados en la tabla 3.2. Los coeficientes C_w, C_r y C_i, corrigen el argumento del perigeo, el radio de la órbita y la inclinación de ésta que son debidas a la inevitable perturbación de la órbita teórica causada por variaciones en el campo de gravedad de la Tierra, la presión y albedo del sol, la atracción del sol y la luna.

Las efemérides que son obtendias de cada uno de los canales sirven para el cálculo de posición del satélite en el sistema geocéntrico X-Y-Z. La tabla 3.4 va sumarizando, a partir de las efemérides, el procedimiento para obtener las coordenadas resultantes para el satélite k.

[1] El WGS84 es un sistema de coordenadas cartográficas mundial que permite localizar cualquier punto de la Tierra (sin necesitar otro de referencia) por medio de tres unidades dadas. WGS84 son las siglas en inglés de World Geodetic System 84 (que significa Sistema Geodésico Mundial 1984).

[2] La semana GPS es el tiempo GPS iniciado la media noche del Sábado/Domingo 6 de enero de 1980. La semana GPS es el número de semanas completas desde la hora GPS cero.

Tabla 3.4: *Elementos del sistema coordenado ECEF.*

$GM = 3.986005 \times 10^{14}$	Valor WGS-84 del parámetro gravitacional universal para usuarios GPS.
$\Omega_e = 7.2921151467 \times 10^{-}5$	Valor WGS-84 de la velocidad de rotación de la Tierra.
$A = (\sqrt{A})^2$	Eje semi-mayor
$n^0 = \sqrt{GM/A^3}$	Movimiento medio (rad/seg)
$t^k = t - {t_{oe}}^2$	Tiempo desde el tiempo de referencia de las efemérides
$n = n_0 - \Delta n$	Movimiento medio corregido
$\mu_k = \mu_0 + nt_k$	Anomalía media
$\pi = 3.1415926535898$	π GPS de referencia
$\mu_k = E_k - esenE_k$	Ecuación de Keppler para la anomalía excentríca (se resuelve por medio de iteraciones) (rad).
$\nu_k = tan^{-1}\left\{\frac{sen\nu_k}{cos\nu_k}\right\} = tan^{-1}\left\{\frac{\sqrt{1-e^2}senE_k}{cosE_k - e}\right\}$	Anomalía verdadera
$E_k = cos^{-1}\left\{\frac{e-cos\nu_k}{1+ecos\nu_k}\right\}$	Anomalía excentríca
$\Phi_k = \nu_k + \omega$	Argumento de Latitud
$\partial u_k = C_{us}sen2\Phi_k + C_{uc}cos2\Phi_k$ $\partial r_k = C_{rs}sen2\Phi_k + C_{rc}cos2\Phi_k$ $\partial i_k = C_{is}sen2\Phi_k + C_{ic}cos2\Phi_k$	Correción de argumento de Latitud Corrección del radio Corrección de la inclinación
$u_k = \Phi_k + \partial u_k$ $r_k = A(1 - ecosE_k) + \partial r_k$ $i_k = i_0 + \partial i_k + (IDOT)t_k$	Argumento de Latitud corregido Radio corregido Inclinación corregida
$x'_k = r_k cosu_k$ $y'_k = r_k senu_k$	Posición en el plano orbital
$\Omega_k = \Omega_0 + (\Omega - \Omega_e)t_k - \Omega_e t_{oe}$	Longitud del nodo ascendente corregido
$x_k = x'_k cos\Omega_k - y'_k cosi_k sen\Omega_k$ $y_k = y'_k cos\Omega_k - y'_k cosi_k sen\Omega_k$ $z_k = y'_k seni_k$	Coordenadas ECEF

3.4 Estimación de la pseudodistancia

Las estimaciones para la pseudodistancia se pueden dividir en dos conjuntos para su computación que consta en encontrar el conjunto inicial de pseudodistancias y las pseudodistancias subsecuentes.

3.4.1 El conjunto inicial de pseudodistancias

Para encontrar las pseudodistancias entre el receptor y los satélites, al menos se necesitan 12 segundos de señal. Cuando el receptor tiene más de 12 segundos de datos, se correlacionan con el preámbulo de la palabra TLM. La palabra TLM se localiza en el incio de cada subtrama [15].

Luego de que el preámbulo es identificado, el inicio de cada subtrama se encuentra para cada uno de los satélites visibles. En la figura 3.7 se muestra un ejemplo de la llegada de una subtrama de cada canal y las diferencias de tiempo entre cada una de ellas.

Se sabe que el tiempo de viaje de los satélites a la tierra es de 65 a 83 ms. Esto es usado para establecer el pseudorango inicial. El satélite más cercano a la tierra es el satélite con la subtrama con llegada más pronta [15]. Por ejemplo, en la figura 3.7, el canal 2 tiene un tiempo de viaje de 67 ms. El tiempo de viaje del los canales restantes se calcula con respecto al canal 1. Para calcular la pseudodistancia de cada canal, basta con multiplicar el tiempo transcurrido por la velocidad de la luz (299,792,458 m/s).

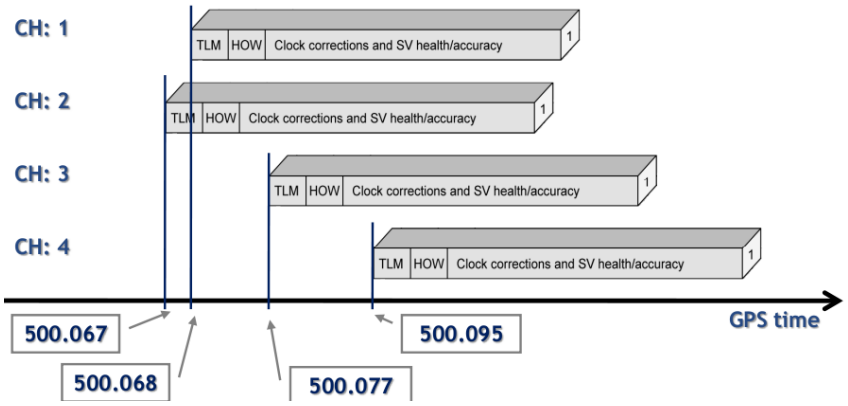


Figura 3.7: *Tiempo de transmisión e inicio de cada subtrama para cuatro canales.*

Para que la pseudodistancia sea más precisa, el ciclo de seguimiento necesita encontrar el inicio del código C/A dentro de la trama. Esto significa que el tiempo de resolución es

el tiempo de muestreo. La frecuencia de muestreo hace que la pseudodistancia logre una exactitud de 8 metros.

Con la pseudodistancia inicial, la posición del receptor ya puede ser computada. La salida de la computación es la posicion del receptor (X,Y,Z) y el offset del reloj del receptor *dt*. El offset del reloj puede ser usado para ajustar el tiempo de viaje desde el satélite de referencia.

3.4.2 Estimación de pseudodistancias subsecuentes.

La sección anterior describe como las pseudodistancias iniciales son estimadas. Esta subsección describe como las pseudodistancias subsecuentes son calculadas.

Cuando se computan las pseudodistancias subsecuentes, se tienen dos cuestiones. La primera cuestión es la diferencia en mili segundos entre el incio de una subtrama comparada con el satélite de referencia. La segunda cuestión es el inicio del código C/A, la cual da la pseudodistancia exacta para el canal.

Cuando el receptor está computando las pseudodistancias subsecuentes, el receptor mueve todos los indices 100 ms. Esto se hace si se ha establecido al receptor de computar posiciones 10 veces por segundo. Luego de esto, se encuentra el incio del código C/A para todos los nuevos indices en los canales de seguimiento. De esta manera es posible computar pseudodistancias cada milisegundo y el receptor hace la computación de la posición 1000 veces por segundo.

3.5 Computación de la posición de usuario.

3.5.1 Linealización de la ecuación de observación

El algoritmo más comúnmente usado para el cálculo de posición es el método de mínimos cuadrados. Este método es usado desde las pseudodistancias para cuatro o más satélites.

Sea el rango geométrico entre el satélite k y el receptor i denotado como ρ_i^k, c la velocidad de la luz, dt^k el desplazamiento (offset) del tiempo del satélite, dt_i el desplazamiento (offset) del reloj del receptor, T_i^k el retardo troposférico, I_i^k sea el retardo ionosférico y e_i^k sea el error de observación del pseudorango. Entonces la ecuación de la pseudodistancia es:

$$P_i^k = \rho_i^k + c(dt_i + dt_k) + T_i^k + I_i^k + e_i^k \tag{3.7}$$

El rango geométrico ρ_i^k entre el receptor y el satélite se calcula como:

$$\rho_i^k = \sqrt{(X^k - X_i)^2 + (Y^k - Y_i)^2 + (Z^k - Z_i)^2} \tag{3.8}$$

Por lo tanto,

$$P_i^k = \sqrt{(X^k - X_i)^2 + (Y^k - Y_i)^2 + (Z^k - Z_i)^2} + c(dt_i + dt_k) + T_i^k + I_i^k + e_i^k \tag{3.9}$$

De las efemérides (las cuales incluyen información del desplazamiento del reloj de satélite dt^k), se puede computar la posición del satélite. Los errores troposféricos e ionosférico T_i^k y I_i^k se calculan por medio de modelos a priori. Así, la ecuación contiene cuatro incógnitas X_i, Y_i, Z_i y dt_i; el error e_i^k se minimiza usando el método de mínimos cuadrados. Para la computación de la posición del receptor, al menos cuatro pseudodistancias se necesitan.

El siguiente paso, es linealizar la ecuación para la pseudodistancia con respecto a la posición de usario (X_i, Y_i, Z_i, dt_i) para poder así ser usada posteriormente dentro del método de mínimos cuadrados.

Se analiza el término no lineal

$$f(X_i, Y_i, Z_i) = \sqrt{(X^k - X_i)^2 + (Y^k - Y_i)^2 + (Z^k - Z_i)^2} \tag{3.10}$$

La linealización comienza al encontrar una posición inicial del receptor $(X_{i,0}, Y_{i,0}, Z_{i,0},)$. La cual a menudo se elige como el centro de la tierra (0,0,0).

Los incrementos $\Delta X, \Delta Y, \Delta Z$ se definen como

$$X_{i,1} = X_{i.0} + \Delta X \tag{3.11}$$

$$Y_{i,1} = Y_{i.0} + \Delta Y \tag{3.12}$$

$$Z_{i,1} = Z_{i.0} + \Delta Z \tag{3.13}$$

Éstas, actualizan las coordenadas aproximadas del usuario. Así, la expansión Taylor de $f(X_{i.0} + \Delta X, Y_{i.0} + \Delta Y, Z_{i.0} + \Delta Z)$ queda como:

$$f(X_{i.1} + \Delta X, Y_{i.1} + \Delta Y, Z_{i.1} + \Delta Z) =$$

$$f(X_{i.0}, Y_{i.0}, Z_{i.0}) + \frac{\delta f(X_{i.0}, Y_{i.0}, Z_{i.0})}{\delta X_{i.0}} \Delta X_i + \frac{\delta f(X_{i.0}, Y_{i.0}, Z_{i.0})}{\delta Y_{i.0}} \Delta Y_i + \frac{\delta f(X_{i.0}, Y_{i.0}, Z_{i.0})}{\delta Z_{i.0}} \Delta Z_i \tag{3.14}$$

Esta ecuación incluye términos de primer orden. Las derivadas parciales son:

$$\begin{aligned}\frac{\delta f(X_{i.0}, Y_{i.0}, Z_{i.0})}{\delta X_{i.0}} &= -\frac{X^k - X_{i,0}}{\rho_i^k} \\ \frac{\delta f(X_{i.0}, Y_{i.0}, Z_{i.0})}{\delta Y_{i.0}} &= -\frac{Y^k - Y_{i,0}}{\rho_i^k} \\ \frac{\delta f(X_{i.0}, Y_{i.0}, Z_{i.0})}{\delta Z_{i.0}} &= -\frac{Z^k - Z_{i,0}}{\rho_i^k}\end{aligned} \tag{3.15}$$

Ahora, $\rho_{i,0}^k$ sea el rango computado de la posición aproximada del receptor, la ecuación de primer orden linealizada queda como:

$$P_i^k = \rho_{i,0}^k - \frac{X^k - X_{i,0}}{\rho_{i,0}^k} - \frac{Y^k - Y_{i,0}}{\rho_{i,0}^k} - \frac{Z^k - Z_{i,0}}{\rho_{i,0}^k} - c(dt_i - dt^k) + T_i^k + I_i^k + e_i^k \tag{3.16}$$

donde,

$$\rho_{i,0}^k = \sqrt{(X^k - X_{i,0})^2 + (Z^k - Z_{i,0})^2 + (Z^k - Z_{i,0})^2} \tag{3.17}$$

3.5.2 Usando el método de mínimos cuadrados

Existe un problema de mínimos cuadrados cuando un sistema A$\mathbf{x}$=$\mathbf{b}$ no tiene solución. A tiene m filas y n columnas, con $m < n$; hay más observaciones $b_1, ..., b_m$ que paramétros libres $x_1, ..., x_n$ [15]. La mejor solución, $\dot{x}$, es minimizando el vector de error $\dot{e}$=b-A$\dot{x}$, así que $\|\dot{e}\|^2 = (\mathbf{b} - A\mathbf{x})^T(\mathbf{b} - A(x))$ es la suma de cuadrados de los m errores separados, minimizando esta cuadrática, da las ecuaciones normales

$$A^T A\dot{x} = A^T\mathbf{b} \tag{3.18}$$

$$\dot{x} = (A^T A)^{-1} A^T\mathbf{b} \tag{3.19}$$

y el vector de error es:

$$\dot{x} = \mathbf{b} - A\dot{x} \tag{3.20}$$

La ecuación de observación linealizada puede ser escrita en la forma de un vector

$$P_i^k = \rho_{i,0}^k + \left[-\frac{X^k - X_{i,0}}{\rho_{i,0}^k} - \frac{Y^k - Y_{i,0}}{\rho_{i,0}^k} - \frac{Z^k - Z_{i,0}}{\rho_{i,0}^k} 1 \right] [\Delta X_i \Delta Y_i \Delta Z_i c dt_i] - c dt^k - T_i^k + I_i^k + e_i^k \tag{3.21}$$

Se reacomoda, para tener la forma usual de un problema de mínimos cuadrados A$\mathbf{x}$=$\mathbf{b}$:

$$\left[-\frac{X^k - X_{i,0}}{\rho_{i,0}^k} - \frac{Y^k - Y_{i,0}}{\rho_i^k} - \frac{Z^k - Z_{i,0}}{\rho_{i,0}^k} 1 \right] [\Delta X_i \Delta Y_i \Delta Z_i c dt_i] = P_i^k - \rho_{i,0}^k + c dt^k + T_i^k - I_i^k - e_i^k \tag{3.22}$$

Una única solución no puede ser encontrada de una sola ecuación. Se hace

$$b_i^k = P_i^k - \rho_{i,0}^k + c dt^k + T_i^k - I_i^k - e_i^k \tag{3.23}$$

Entonces, la solución final viene dada como:

$$A\mathbf{x} = \begin{bmatrix} \frac{X^1 - X_{i,0}}{\rho_{i,0}^1} & \frac{Y^1 - Y_{i,0}}{\rho_i^1} & \frac{Z^1 - Z_{i,0}}{\rho_{i,0}^1} & 1 \\ \frac{X^2 - X_{i,0}}{\rho_{i,0}^2} & \frac{Y^2 - Y_{i,0}}{\rho_i^2} & \frac{Z^2 - Z_{i,0}}{\rho_{i,0}^2} & 1 \\ \frac{X^3 - X_{i,0}}{\rho_{i,0}^3} & \frac{Y^3 - Y_{i,0}}{\rho_i^3} & \frac{Z^3 - Z_{i,0}}{\rho_{i,0}^3} & 1 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \frac{X^m - X_{i,0}}{\rho_{i,0}^m} & \frac{Y^m - Y_{i,0}}{\rho_i^m} & \frac{Z^m - Z_{i,0}}{\rho_{i,0}^m} & 1 \end{bmatrix} \begin{bmatrix} \Delta X_{i,1} \\ \Delta Y_{i,1} \\ \Delta Z_{i,1} \\ c dt_{i,1} \end{bmatrix} = \mathbf{b} - \mathbf{e} \tag{3.24}$$

Si $\mathbf{m} \geq 4$ hay una única solución: $\Delta X_{i,1}, \Delta Y_{i,1}, \Delta Z_{i,1},$. Ésta debe ser sumada a la posición aproximada del receptor para obtener así la siguiente posición aproximada:

$$\begin{aligned} X_{i,1} &= X_{i.0} + \Delta X_{i,1} \\ Y_{i,1} &= Y_{i.0} + \Delta Y_{i,1} \\ Z_{i,1} &= Z_{i.0} + \Delta Z_{i,1} \end{aligned} \tag{3.25}$$

La siguiente iteración es cuando los subíndices $_{i,0}$ son substituidos por $_{i,1}$. Estas iteraciones continuan hasta que la solución $\Delta X_{i,1}, \Delta Y_{i,1}, \Delta Z_{i,1}$ es a nivel de metros. A menudo de dos a tres iteraciones son suficientes para obtener esta meta [19] .

3.6 Transformación de coordenadas

El marco de referencia actual para la navegación terrestre es conocida como el sistema coordenado ECEF-g (Earth centered-Earth fixed geodetic, en sus siglas en inglés). Las coordenadas para este sistema usualmente son escritas como π, ϕ, h para latitud, longitud y altitud respectivamente. La altitud es definida como la distancia perpendicular por encima de una forma de la tierra conocida como geoide la cual es matemáticamente un elipsoide con una revolución promedio se aproxima a la verdadera forma de la Tierra [20].

Algunas veces es necesario definir un marco de referencia alterno conocido como sistema ECEF rectangular (ECEF-r). En este esquema, el origen se encuentra en el centro de la Tierra, el eje-Z está dirigido hacia el polo norte, el eje-X es perpendicular al primer meridiano y el eje-Y se elige para completar un sistema coordinado rectangular X-Y-Z, como se puede observar en la figura 3.8.

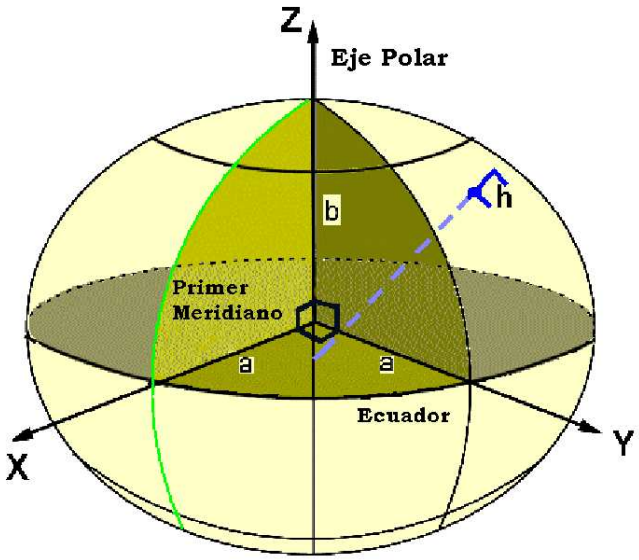


Figura 3.8: *Forma elipsoidal de la Tierra en ECEF.*

El resultado inmediato del método de mínimos cuadrados para la posición de usuario es en sistema ECEF-r con coordenadas en X-Y-Z. En navegación se utiliza el sistema ECEF-g con lectura en latitud, longitud y altura. Un proceso matemático se realiza para la transformación de un sistema a otro.

Para lograr esto, se debe conocer la forma exacta del elipsoide de la Tierra. La forma definida (también conocida como el *datum*), usada en los sistemas de navegación por satélite actuales, es conocida como *WGS-84*.

Tabla 3.5: *Parámetros terrestres WGS84*

Constante	*Valor*	*Unidad*	Significado
π	3.1415926535898		Pi
a	6378137.0	m	Eje semi-mayor de la Tierra WGS-84
b	6356752.3142	m	Eje semi-menor de la Tierra WGS-84
ω_{ie}	$7.2921151467x10^{-5}$	r/s	Valor de la rotación de la Tierra WGS-84
GM	$3.986005x10^{-14}$	m^3/s^2	Valor de la constante gravitacional de la Tierra WGS-84

El WGS84 se trata de un estándar en geodesia, cartografía, y navegación, que data de 1984. Tuvo varias revisiones (la última en 2004), y se considera válido hasta una próxima reunión (aún no definida en la página web oficial de la Agencia de Inteligencia Geoespacial). Se estima un error de cálculo menor a 2 cm. por lo que es en la que se basa el Sistema de Posicionamiento Global (GPS).[20].

El WGS-84 consiste en un patrón matemático de tres dimensiones que representa la tierra por medio de un elipsoide, un cuerpo geométrico más regular que la Tierra que tiene parámetros ya definidos (Tabla 3.5). Solamente los primeros tres valores de esta tabla son requeridos para la transformación de coordenadas.

Primero se define el achatamiento del elipsoide

$$f = \frac{a-b}{a} = 3.3528107 \times 10^3 \tag{3.26}$$

La excentricidad se puede encontrar como

$$e = \sqrt{f(2-f)} = 8.1819191 \times 10^2 \tag{3.27}$$

Luego, se calcula la longitud:

$$\pi = atan(Y/X) \tag{3.28}$$

Nótese que este cálculo arroja los ángulos en radianes por los cuales luego deben ser convertidos en grados.

Enseguida, el radio físico del punto y el radio en el plano X-Y se calculan para estimar un valor inicial de la altitud.

$$r = \sqrt{X^2 + Y^2 + Z^2} \tag{3.29}$$

$$p = \sqrt{X^2 + Y^2} \tag{3.30}$$

La latitud geocentrica se calcula de la misma manera y usada como valor inicial en el ciclo de iteración.

$$\phi_c = atan(p/z) \tag{3.31}$$

$$\phi_{now} = \phi_c \tag{3.32}$$

Y, el ciclo es:

$$h = \frac{p}{cos\phi_{now}} - R_n(\phi_{now})\phi_{next} = atan[\frac{z}{p(1 - e^2\frac{R_n}{R_n+h})}] \tag{3.33}$$

donde R_n es el radio de curvatura en el primer vertical y está dado por:

$$R_n = \frac{a}{\sqrt{1 - e^2 sen^2\phi}} \tag{3.34}$$

Este ciclo converge luego de algunas iteraciones (cuatro al menos) [21]. Así, se encuentra la latitud geodésica, ϕ, y como fase final, se calcula la latitud, h.

$$h = \frac{p}{cos\phi} - R_n \tag{3.35}$$

Estas ecuaciones se pueden definir y se basan en la geometría de la figura 3.9.

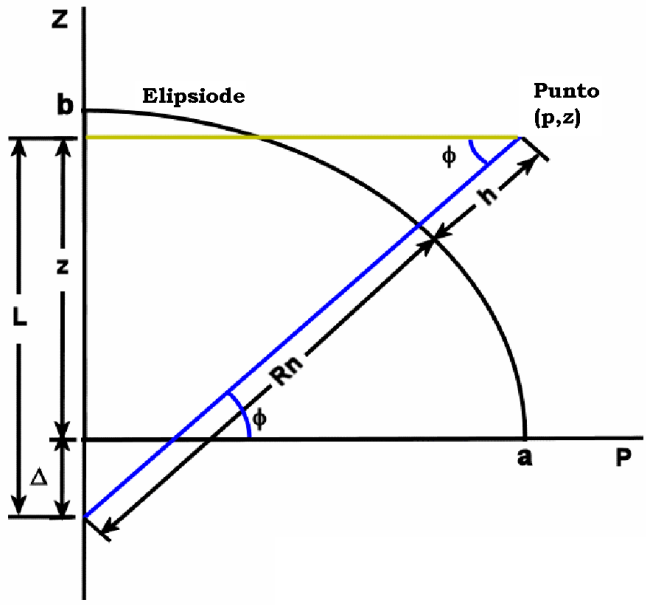


Figura 3.9: *Geometría utilizada para la transformaciones de coordenadas ECEF-g a ECEF-r y viceversa.*

donde:

Punto(x,y,z)

p=	$\sqrt{X^2+Y^2}$
ϕ=	Latitud geodésica.
h=	Altura elipsoidal.
e=	excentricidad=$\sqrt{1-\frac{b^2}{a^2}}$
R_n=	longitud del radio de la curvatura=$\frac{a}{\sqrt{1-e^2 sen^2\phi}}$
Δ =	$e^2\ R_n\ sen\ \phi$
L=	$(R_n+h)sen\phi = z+\Delta$

Capítulo 4

Descripción del sistema desarrollado basado en FPGA.

El ensamble del *front-end* con la FPGA se describe en este capítulo. La tarjeta de desarrollo utilizada en este proyecto de tesis es la ofrecida por Xilinx, Virtex 6 ML605 debido a sus capacidades de almacenamiento en memoria y los multiples periféricos que ofrece. El puerto USB es el que maneja la entrada de datos hacia la tarjeta.

Para lograr la comunicación entre la ML605 y el *front-end SiGe 4120*, se creó un puente de datos USB que sirve comunicación entre los dos chips USB que manejan estos dos dispositivos. Ambos chips son manufacturados por *Cypress*. La plataforma de comunicación se logró modificando el software que ofrece *Cypress* [22] [23] para el servicio de sus productos.

En proyectos de telecomunicaciones satélitales, el portafolio ofrecido por Xilinx satisface de hecho los requerimentos más demandantes en términos de desempeño y robustez [6]. Como consecuencia, el entorno de desarrollo de Xilinx está completamente integrado con las herramientas de diseño de sistemas más comunes (por ejemplo Simulink).

Xilixn desarrolla FPGAs y CPLDs (del acrónimo inglés *Complex Programmable Logic Device*) que son usados en numerosas aplicaciones, como telecomunicaciones, automoción, productos de consumo, industria militar y otros campos. Por lo tanto, ofrece un ambiente de desarrollo propio para sus productos. La unión entre hardware y software del sistema desarrollado en este trabajo de tesis se realizó utilizando esta herramienta, el Kit de Desarrollo Embebido EDK (en sus siglas en inglés *Embedded Development Kit*). La versión utilizada de EDK en este trabajo de tesis fue la 13.2 debido a que esta versión tiene caracterizada a la ML605.

4.1 Kit de Evaluación Virtex 6 ML605

El Kit de Evaluación Virtex 6 FPGA ML 605 provee un ambiente de desarrollo para el diseño de sistemas que demandan un alto desempeño, conectividad serial y un interfaz avanzado de memoria. En la figura 4.1 se puede observar un esquema general de la tarjeta Virtex 6 ML605.

De las caracteristícas importantes que ofrece el Kit de Evaluación Virtex 6 FPGA ML605, son:

- Memoria DDR3 de 512 MBytes.
- Memoria Flash de 128 MBytes.
- Sistema ACE y Conector *CompactFlash.*
- USB-JTAG
- Generación de reloj. Oscilador de 200 MHZ (diferencial).
- Puerto Ethernet 10/100/1000.
- Puente USB - UART
- Periférico DVI.
- LEDs de estatus.
- Periféricos de entrada/salida para el usuario
  - Grupo 1 de LED de usuario - GPIO (*Entrada/Salida de Propósito General*) (8).
  - Grupo 2 de LED de usuario - direccionales (5)
  - *Pushbuttons* de usuario - direccionales (5)
  - *Pushbutton* de reset CPU.
  - *Interruptor DIP* de usuario - GPIO (8 polos).
  - Pantalla LCD (16x2 líneas).
- Interruptores
  - Interruptor de encendido y apagado
  - Botón de reset del sistema ACE CF.
- Administración de la energía. Monitoreo del voltaje PMBus y la corriente vía un controlador de energía TI.

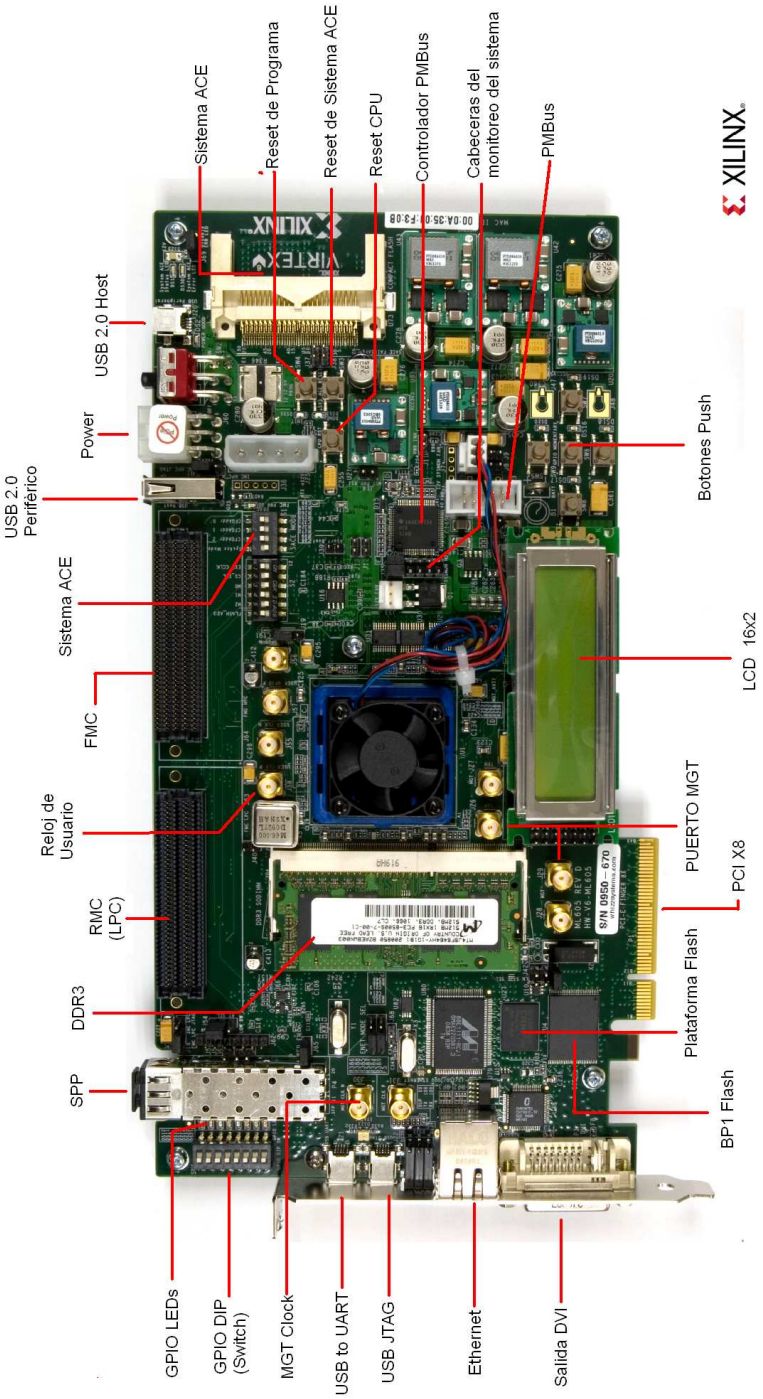


Figura 4.1: *Kit de Evaluación Virtex 6 FPGA ML605.Cortesía de Xilinx.*

La figura 4.2 muestra un diagrama en bloques de cómo los periféricos de ML605 están enlazados al Virtex 6 FPGA.

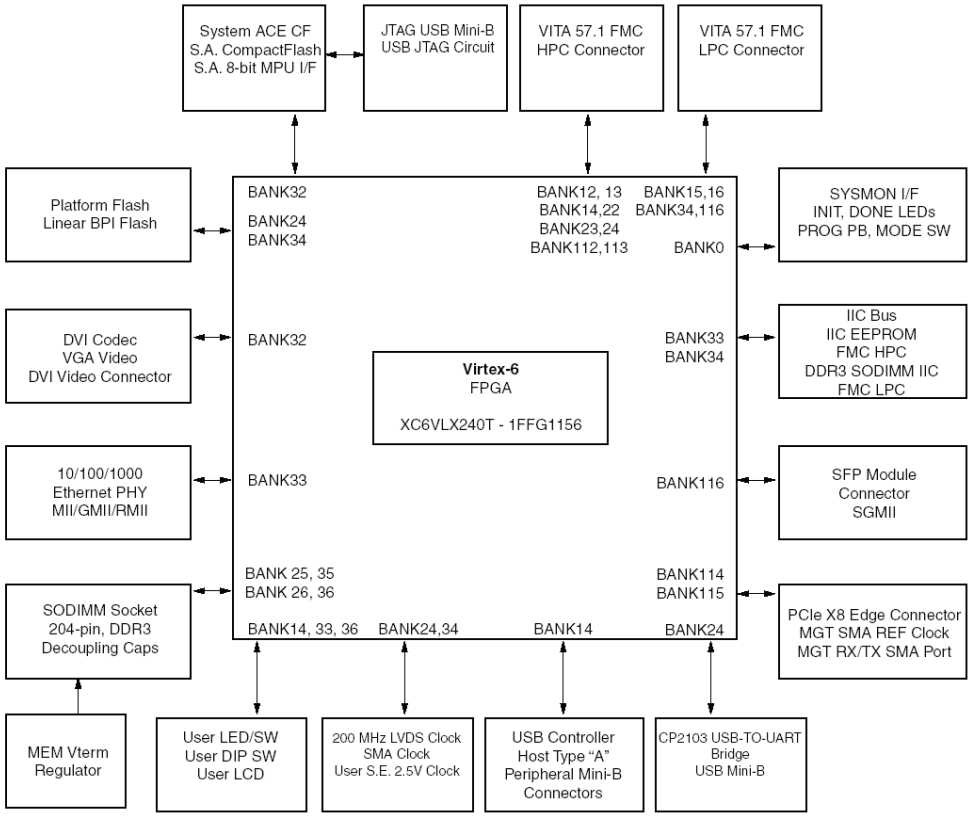


Figura 4.2: *Diagrama en bloques del ML605 con sus periféricos. Cortesía de Xilinx.*

La decisión de elegir este tipo de FPGA y el procesador Microblaze es en su mayoría debido al cómputo poderoso y la capacidad del procesamiento de señales que ofrece [24]. También, porque la tarjeta ML605 ofrece dos puertos USB (*Host y Periférico*) para poder así recibir la señal proveniente del *front-end* y hacer luego su procesamiento. Microblaze es un procesador *softcore* diseñado por Xilinx para sus FPGAs con un amplio conjunto de instrucciones optimizadas para aplicaciones embebidas. Con Microblaze se tiene la solución de una flexibilidad completa en el diseño de sistemas para así, seleccionar la combinación de periféricos, memoria y características de las interfaces y dar un sistema exacto en un solo FPGA. La ML605 a su vez, incluye una versión mas moderna del procesador Microblaze.

Para simplificar el proceso del diseño, Xilinx ofrece el Kit de Desarrollo Embebido, EDK (*Embedded Development Kit*), una herramienta que permite el diseño de sistemas completos de procesadores embebidos para su implementación en las tarjetas de Xilinx. EL EDK

contiene diferentes bloques pre-desarrollados con diferentes funciones como:

- ⋆ Operaciones lógicas. Contadores, multiplexores, operaciones matemáticas básicas, memorias RAM, ROM, FIFO, etc.
- ⋆ Procesamiento Digital de Señales. Filtros IIR y FIR, FFT, etc.
- ⋆ Comunicación. Conversión digital de subida y bajada, moduladores digitales, ciertos codificadores y decodificadores FEC, etc.
- ⋆ Cómputo. Procesadores embebidos. Periféricos estándares tales como interruptores, controladores, controladores de buses, UARTs, IOs, etc.

Xilinx también crea núcleos IP (*IP cores*) en lenguaje HDL para permitir a los diseñadores reducir los tiempos de desarrollo. Estos núcleos IP (Propiedad Intelectual), pueden ser utilizados para el diseño de sistemas embebidos basados en FPGA. Al mismo tiempo, en cuanto a las necesidades de sistemas específicos, la creación de núcleos IP personalizados se pueden añadir al sistema. Los núcleos IP ofrecidos por Xilinx van desde funciones simples como contadores, a sistemas complejos como microcontroladores, un ejemplo de esto último es el microprocesador *Microblaze*.

Debido a estas características, todas las funciones relacionadas con los receptores GNSS pueden ser implementadas en una FPGA. El procesador *Microblaze* desempeña diferentes funciones y tareas como:

1 Gestión de los núcleos IP.

2 Control y manejo de los datos recibidos desde el *front-end*.

3 Desplegado de los mensajes de navegación, ya una vez proceada la señal GNSS.

La figura 4.3 muestra los detalles de la arquitectura propuesta para un receptor de señales GPS basado en software. Los algoritmos de adquisición y navegación ya fueron realizados [14], [5]. En esta tesis se presenta el enlace de estos dos bloques con la etapa de cálculo de posición. Para ver más detalles ver el capítulo 3.

El receptor diseñado para esta tesis consiste de tres partes, el *front-end* analógico el cual convierte una señal RF a IF y provee muestras digitalizadas. Estas muestras luego entran a un buffer via USB hacia un equipo host y guardadas en memorias FIFO como se muestra en la figura 4.3. Todos los componentes mostrados en la figura 4.3 están comercialmente disponibles. El puente USB es el protocolo de comunicación entre el chip interno del USB del *front-end* con el de la tarejta ML605 para así tener la conexión de todos los componentes.

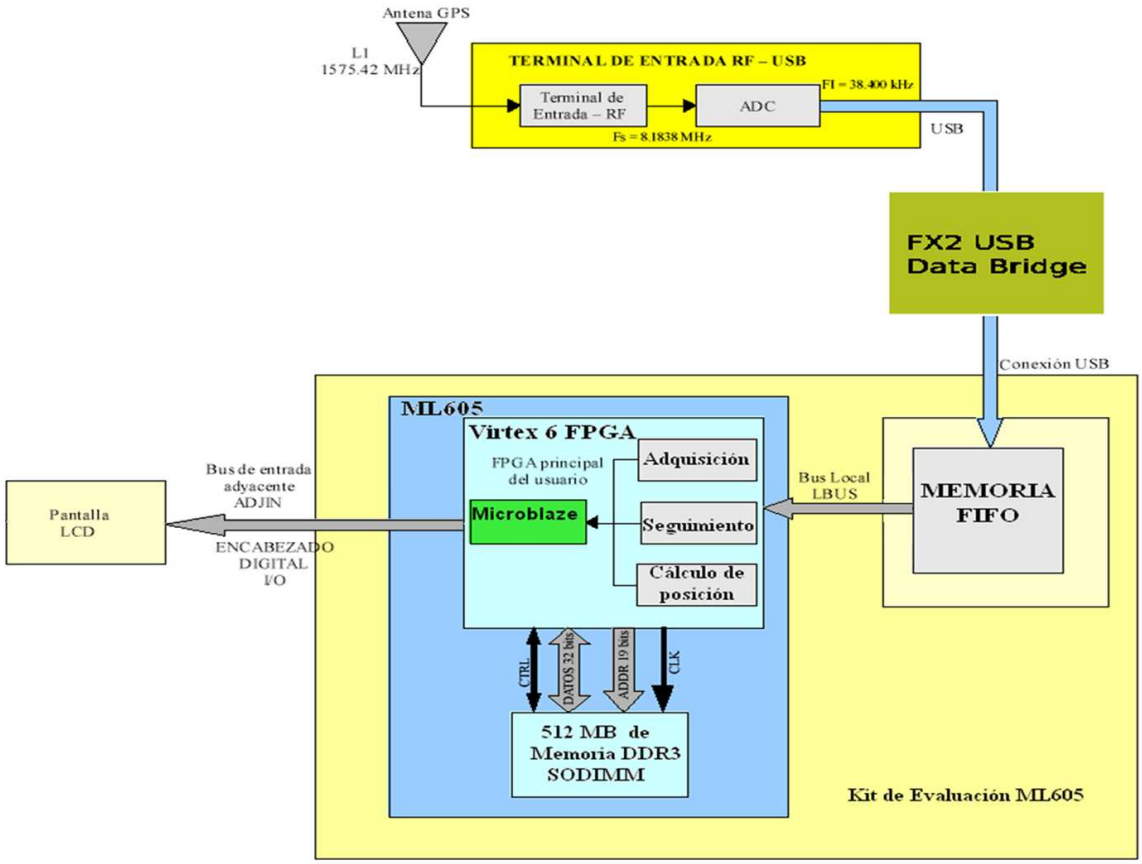


Figura 4.3: *Arquitectura propuesta de un receptor GPS basado en software.*

4.2 Terminal de entrada RF - USB (GN3Sv2)

El GN3S fué construido por la compañia *Sparkfun Electronics* en cooperación con la *Universidad de Colorado* y el *Centro danés GPS DGC*. Este dispositivo provee la solución completa de una posición como los módulos GPS que se encuentran actualmente en el mercado. El GN3S está diseñado para capturar las señales provenientes directamente de los satélites GPS, a través de la antena que incluye. Estas señales GPS son procesadas por este dispositivo y convertidas hacia una frecuencia ntermedia, la cual es muestreada y cuantizada para obtener muestras crudas de la señal en banda base proveniente de la red de satélites GPS. Los datos capturados (ya digitales) pueden ser procesados luego en algún ambiente de desarrollo computacional.

La arquitectura propuesta utiliza la segunda versión del GN3S (GN3s v2.0) la cual está construida sobre el *SiGe 4120 GPS ASIC*. Este dispositivo proporciona un flujo de datos

con una frecuencia de muestreo baja (2.1535MHz) y un par de muestras I/Q. La figura 4.4 muestra la forma física y real que tiene el *front-end* utilizado [25].

Los parámetros específicos de los datos capturados de este módulo son los siguientes:

- Frecuencia de muestreo: 8.1838 MHz.
- Frecuencia intermedia: 38.400 KHz.
- 2 bits de muestras I/Q (1 bit para I y 1 bit para Q) en un formato binario schar [1] (sI0,sQ0,sI1,sQ1,sI2,sQ2..)

Figura 4.4: *Front-end SiGe 4120 GPS ASIC. Cortesía de Xilinx.*

Una ilustración más detallada del interfaz del *front-end* hacia el FPGA vía USB se puede ver en la figura 4.5.

4.3 Puente de datos USB

4.3.1 Cypress EZ-USB FX2LP

Muchos diseños basados en USB están basados en chips de *Cypress* y son muy comunes en modernos FPGAs y puertos USB. El *front-end* comercial y la tarjeta ML605 utilizan el chip *Cypress EZ-USB FX22LP*, el cual es un controlador USB de alta velocidad y baja potencia (281 mW) [26]. Los chips FX2 son bastante rápidos como para proveer un alto desempeño en

[1] schar es un tipo de dato carácter con signo de 8 bits.

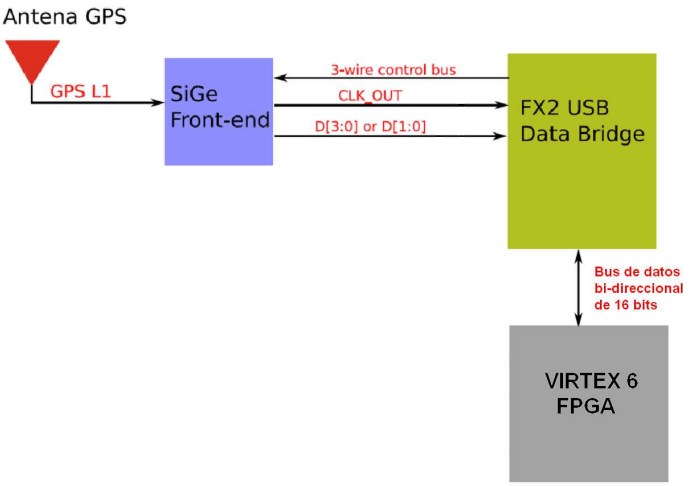


Figura 4.5: *Diagrama en bloques del interfaz al FPGA*

la trasnferencia de datos. La máxima tasa de transferencia de datos es de 96 MB/s. Cuando el *host* no puede manejar altas tasas de datos, la FIFO en el chip FX2 se desborda y la cadena de datos se rompe.

El FX2LP consiste de cinco principales componentes:

- Transceptor USB.
- Motor de Interfaz Serial (SIE).
- Interfaz USB (con FIFOs).
- Microcontrolador 8051.
- GPIF (Máquina de estados programable).

Un diagrama de bloques simplificado del chip EZ-USB se muestra en la figura 4.6
El FX2LP es un chip muy flexible que puede ser configurado para usar un solo canal de datos interno (máximo 96 MB/s) o diferentes tipos de canales de tasas de datos más bajas.

Terminales

El chip EZ-USB tiene tres terminales de 64 bytes para el control USB para la alta transferencia de datos:

- EP0

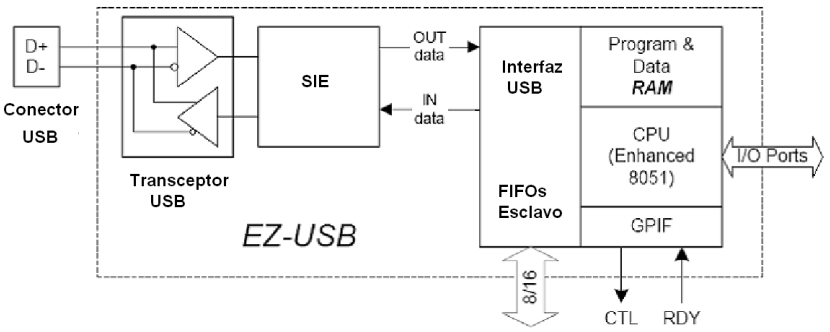


Figura 4.6: *Diagrama de bloques simplificado del EZ-USB.*

- $EP1\_IN$
- $EP1\_OUT$

También hay cuatro terminales de altas velocidades. Cada terminal puede ser configurada como sigue:

- IN / OUT
- Interrupción / Asíncrono
- Doble / Triple / Cuadrúple buffer

Cada una de estas terminales puede ser también configuradas para operar en modo "AUTO" con el GPIF como controlador.

Interfaz Programable General (GPIF)

El GPIF es una máquina de estados de propósito general y puede ser programado para funcionar como interfaz para diferentes periféricos externos con formas de onda ya predefinidas. EL GPIF puede operar a 30 o 48 MHz con un reloj interno a la vez que con un reloj externo (encima de 48 MHz). Cuando una terminal opera en modo AUTO, el GPIF corre de acuerdo con los estatus predefinidos. Para transferencias automáticas de entrada, el GPIF puede ser reprogramado para esperar una bandera de llenado de la FIFO para que luego el GPIF haga todo el trabajo de transferencia de datos a alta velocidad, que es usualmente el caso en que el 8051 es demasiado lento como para interactuar con las cadenas de datos a altas velocidades.

Tabla 4.1: Configuraciones de Arranque de la Firmware

Dirección EEPROM	Contenidos
0x0	Comando de inicio (0xC0)
0x1	Vendor ID (VID) L
0x2	Vendor ID (VID) H
0x3	Product ID (PID) L
0x4	Product ID (PID) H
0x5	Device ID (DID) L
0x6	Device ID (DID) H
0x7	Byte de configuración (0x4)

Opciones de arranque

Hay muchas opciones disponibles de cómo arrancar o iniciar el firmware y solamente las dos más comunes serán descritas. La forma más fácil de iniciar el firmware es conectar la tarjeta y luego descargar el firmware directamente en la RAM. Una forma más práctica es iniciando el firmware desde la EEPROM. Sin embargo, esto requiere que una EEPROM esté presente y depende del tamaño del firmware. El firmware predeterminado en la tarjeta puede manejar solamente comandos de descarga RAM (*Vendor Cypress 0xA0*) y por lo tanto es necesario el firmware capaz para descargar en la EEPROM (Vendor Cypress 0xA9)[26].

Sí no hay ninguna EEPROM presente, la tarjeta arrancará el dispositivo definido por el fabricante, sin embargo, si una EEPROM es detectada, los primeros ocho bytes determinan de como el dispositivo debe ser arrancado. La tabla 4.1 muestra la configuración de arranque.

Un 0xC0 indica que solamente el VID, PID y DID son leídos de la EEPROM. La otra opción es una 0xC2 la cual dice que el dispositivo se cargará desde la EEPROM (y usa VID, PID, y DID desde el firmware). El último byte es para la configuración IIC (I^2C).

Microcontrolador 8051

El microcontrolador 8051 es responsable de inicializar el chip FX2 y monitorear el estatus del GPIF. No participa en la trasferencia de datos a alta velocidad. Durante una transferencia de datos de alta velocidad, el microcontrolador sondea periódicamente el estado del GPIF para asegurarse de que el GPIF no está inactivo (lo que indicaría un desbordamiento de búfer). Estas tareas son todas manejadas por el 8051.

Motor USB

La tarea del motor USB es sencilla: Tan pronto que un búfer de 512 bytes se llene, se preparan los contenidos del búfer para la trasnferencia por USB y luego esperar que el software pida los

datos. El USB tiene un protocolo estricto de maestro/esclavo y el esclavo no puede mandar datos al *host* al menos que el *host* requiera los datos primero.

El puente de datos USB [23] es usado para mover muestras IF de alta velocidad desde el *front-end* al equipo *host*. Este puente de datos puede mover de manera fiable, datos por encima de los 39 M/s en una computadora (1 GHz) moderna. El chip del USB de *Cypress Semiconductor, CY7C68013 FX2*, toma 8 ó 16 bits de datos en el flanco de subida del reloj (CLK_OUT) y los guarda en una FIFO para ser empaquetada por un chip manejador de USB. Ya que el tamaño del paquete más pequeño es de 8 bits, el *front-end* GPS provee datos a 2 bits (configuración de fábrica), esto significa que solamente 4 de cada 8 bits mandados, contienen datos en modo de *2-bit*. Así, los datos son mandados al *host* a 16 Mb/s, donde la tarjeta guarda estos bits a 8 Mb/s.

4.3.2 Firmware

Una porción del firmware que corre en el chip FX2 ha sido creado por desarrolladores de *GNU Radio* [27]. Este firmware has sido modificado ampliamente para acomodar y acoplar los datos entre el *front-end* y el chip FX2 de la tarjeta ML605. El firmware se ha compilado en una computadora con el compilador *sdcc* [28] y subido al firmware vía USB usando la herramienta $fx2\_programmer$ [22].

Ya que el código de *GNU Radio* hace interfaz a diferentes plataformas de hardware, partes cruciales del código ha tenido que ser reescrito. Para un mejor entendimiento de la operación FX2, ver la figura 4.7. El chip FX2 consiste de cuatro partes, el Interfaz Programable General (GPIF), *buffers*, un microcontrolador 8051 y un motor USB.

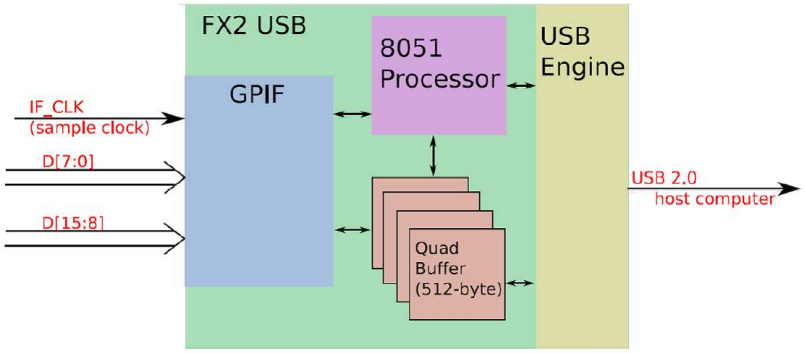


Figura 4.7: *Flujo de datos a través del puente USB.*

4.3.3 Comunicación del *Front-End SiGe 4120* con la tarjeta *ML605*

Cypress ofrece un kit de herramientas para el desarrollo y modificación de aplicaciones vía software para sus dispositivos. *GNU Radio* ha desarrollado un código que puede ser modificado por cualquier usuario para la manipulación de sistemas que tengan integrados el chip de Cypress. Éste es el caso del *front-end SiGe 4120* y la tarjeta *ML605*.

Se necesita arrancar un *firmware* directamente desde la EEPROM del chip Cypress contenido en la ML605 para poder realizar la transferencia de datos entre estos dos dispositivos.

Xilinx ofrece un proyecto de referencia que habilita y utiliza el puerto USB para la tarjeta ML40x. El código fuente para este proceso funciona también para la ML605.

Imagen EEPROM

1 Descargar e instalar las herramientas USB directamente desde la página web de Cypress. Se encuentran como *CY3663-EZ-OTG-/EZ-Host Development kit* en *http://www.cypress.com/design/SD1025*. El registro a Cypress es requerido.

2 Dentro de la carpeta ML40x, contiene un directorio llamado "*eeprom*" y contiene los archivos necesarios para generear la imagen de la EEPROM. Luego, se abre un ambiente *BASH* creado especialmente por *Cypress* para sus dispositivos.

3 Se escribe *"make"* en la pantalla para crear el archivo *eeprom.bin*.

4 Se conecta el cable USB a la tarjeta. Para la ML605 se requiere conectar un puerto USB tipo A a un mini-B entre la PC y la ML605.

5 Ya instaladas el kit de herramientas de *Cypress*, se corre el siguiente comando en una ventana DOS.
$C:/Cypress/USB/OTG-Host/tools/utilities/qtui2c.exe$
$[locationofUSBfiles]/eeprom.binf$

Archivo *.bin Compact Flash

El directorio *keyboard* se ha modificado especificamente utilizando la descripción del fabricante del *Front-End SiGe 4120*. Este directorio contiene los archivos necesarios para generar el archivo **.bin* y colocarlo en la Compact Flash para así realizar la transferencia de datos entre en *Front-End* y la ML605.

Se abre nuevamente un ambiente *BASH* y dentro del directorio, se teclea *make* para crear el archivo **.bin*. Luego, se copia el archivo generador *demo.bin* en la Compact Flash utilizando un lector comercial de memorias y se inserta en la ML605.

4.4 Entorno en Software del Sistema

Un sistema embebido o empotrado es un sistema de computación diseñado para realizar una o algunas funciones dedicadas frecuentemente en un sistema de computación en tiempo real. Se llama embebido ya que incorpora en el mismo dispositivo, hardware, software e incluso partes mecánicas [12]. Los sistemas embebidos controlan muchos dispositivos de uso común hoy en día.

Obtener las porciones de hardware y software en un sistema embebido trae retos adicionales y tiene potencial de convertise muy confuso en términos de diseño [14]. Para simplificar el proceso de diseño, Xilinx ofrece varios conjuntos de herramientas hechos específicamente para sus productos. Es conveniente conocer los nombres de estas herramientas básicas, nombre de los archivos de proyecto, acrónimos y abreviaturas. Los componentes de EDK (EDK - *Embedded Development Kit*) se mencionan en la siguientes líneas.

4.4.1 Entorno de Software Integrado

El entorno de Software Integrado (ISE - *Integrated Software Environment*) es la base para el diseño lógico sobre FPGA de Xilinx. Debido a que el diseño con FPGA puede ser un proceso complicado, Xilinx ha proporcionado herramientas de desarrollo de software que permiten simplificar algunas de estas complejidades. Varias utilidades, como son restricciones de entrada, análisis de tiempo, ruteo lógico y programación de dispositivos han sido integradas en ISE.

ISE es una herramienta de software para la síntesis y análisis de los diseños de HDL, lo que permite al desarollador el sintetizar (compilar) sus diseños, realizar análisis de tiempo, examinar diagramas RTL, simular la reacción de un diseño a diferente estímulos y configurar el dispositivo.

4.4.2 Kit de Desarrollo Embebido

El kit de desarrollo embebido EDK es un conjunto de herramientas y Propiedad Intelectual (IP) que permite en el diseño tener un completo sistema de procesador embebido para la implementación en dispositivos FPGA de Xilinx. El software Xilinx ISE también debe ser instalado para ejecutar EDK.

Plataforma de Estudio de Xilinx

La Plataforma de Estudio de Xilinx (XPS, *Xilinx Platform Studio*) es el ambiente de desarrollo o GUI usada para el diseño de la porción de hardware del sistema de procesador embebido.

Kit de Desarrollo de Software

El Kit de Desarrollo de Software (SDK *Software Development kit*) es un ambiente de desarrollo integrado, complementario a XPS, que es usado para la creación y verificación de aplicaciones de software embebido desarrolladas en C/C++. SDK es construido sobre *Eclipse*, por que lo hace una herramienta de software familiar para varios desarrolladores.

Componentes Adicionales EDK

A continuación se presenta una lista de otros elementos que contiene EDK.

- Hardware IP para el procesador embebido de Xilinx.
- Drivers y librerías para el desarrollo de software embebido.
- Compilador GNU y depurador para desarrollo de software en C/C++ que tiene como finalidad el desarrollo sobre procesadores Microblaze y PowerPC.
- Documentación.
- Ejemplos preeviamente hechos que sirven como referencia en diseños de proyectos.

4.4.3 Herramientas para el proceso de diseño

La figura 4.8 muestra un diagrama de flujo simplificado para un diseño embebido. A continuación se da una descripción general de cómo estas herramientas trabajan juntas para simplificar el proceso del diseño.

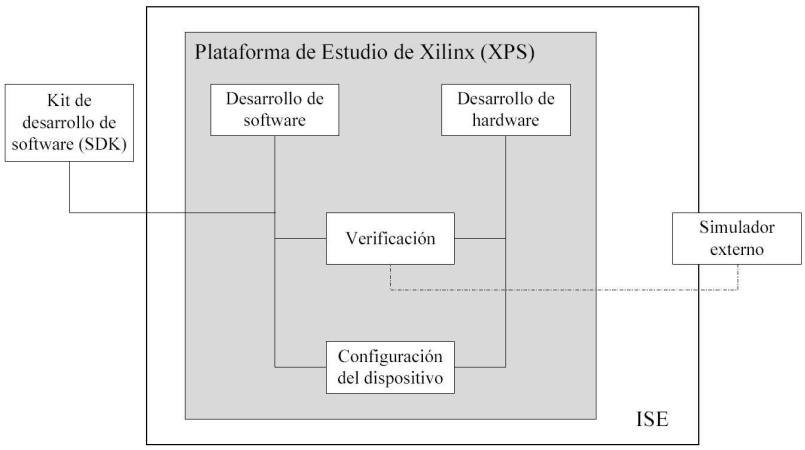


Figura 4.8: *Diagrama de flujo de un proceso embebido básico.*

- El diagrama de flujo de la figura 4.8 recomienda comenzar con un proyecto ISE, y entonces agregar una fuente de procesador embebido al proyecto ISE.
- XPS es usado principalmente en el desarrollo de sistemas hardware de procesador embebido. Configuración del microprocesador, periféricos, y la interconexión de estos componentes, junto con la asignación de sus respectivas propiedades, tiene lugar en XPS.
- SDK es el ambiente de desarrollo de software recomendado para aplicaciones de software simples y complejas. Mientras el desarrollo en software básico podía ser realizado dentro de XPS, esta capacidad fué removida a partir de la versión 13 del *ISE Design* .
- Verificar el correcto funcionamiento de su plataforma de hardware puede ser realizado ejecutando el diseño a través de un simulador de Lenguaje de Descripción de Hardware (HDL - *Hardware Description Lenguage*). XPS facilita los tres tipos de simulación: Situacional, Estructural y Tiempo Real.

XPS instala automáticamente el proceso de verificación, incluyendo archivos HDL para la simulación.

4.4.4 Requerimentos de Instalación

ISE Xilinx

Varias utilidades en EDK usan la funcionalidad que se entrega en las herramientas contenidas en ISE. Entonces, para usar las herramientas de EDK, primero es necesario tener instalado el ISE. Estar seguro que también se tiene instalado el último *Service Pack* de ISE. Para obtener información y descargar la versión más actual para el software ISE, ver [29].

Instalación de EDK

Las instrucciones exactas de instalación varían, dependiendo de si el software se obtuvo a partir de una descarga electrónica o un DVD. Es importante que el ISE y EDK sean la misma versión [29]. Es decir, si se instala la versión 13.2 de EDK, se debe instalar la versión 13.2 de ISE.

Requerimentos de instalación para simulación

Para realizar simulación usando las herramientas de EDK, se deben tener los siguientes pasos completos.

1 Un simulador instalado (ModelSim PE/SE v6.3c o Mentor Graphics IUS v6.1) es requerido para los pasos de simulación.

2 Compilar las librerías de simulación.

4.4.5 Creación del proyecto

El constructor de sistema base (BSB, *Base System Builder*) es un asistente que rápida y eficientemente establece un grupo de trabajo de diseño, que también se puede modificar. El proyecto es creado usando el BSB. Xilinx recomienda el uso del asistente BSB para crear la base de cualquier nuevo proyecto de diseño embebido. BSB puede ser todo lo necesario para crear un diseño, pero si más adecuaciones son requeridas, BSB ahorra mucho tiempo debido a que automatiza configuraciones básicas de la plataforma hardware y software de las tareas comunes de la mayoría de los diseños con procesador. Después de ejecutar el asistente, se tiene un proyecto que contiene todos los elementos básicos necesarios para construir uno o más sistemas personalizados o complejos, si fuera necesario.

Usando el asistente BSB, se puede crear el archivo del proyecto, escoger una tarjeta, seleccionar y configurar un microprocesador y el uso de interfaces I/O, agregar periféricos internos, crear software, y generar un reporte con el resumen del sistema. BSB reconoce los componentes y configuraciones del sistema de la tarjeta seleccionada y establece las opciones adecuadas para nuestra selección [5].

Creando el archivo de nivel superior del proyecto (**.xmp*)

Un archivo llamado Proyecto de Microprocesador de Xilinx (XMP, *Xilinx Microprocessor Project*) es el archivo de nivel superior que da la descripción del sistema embebido durante el desarrollo. Toda la información del proyecto del XPS es guardada en el archivo XMP, incluyendo la ubicación de los archivos de Especificación de Hardware del Microprocesador (MHS, *Microprocessor Hardware Specification*) y la Especificación de Software del Microprocesador (MSS, M*icroprocessor Software Specification*).

El archivo XMP también contiene información sobre código C y archivos de cabecera que XPS compila, así como algún archivo ejecutable que el SDK compile. El proyecto también incluye las familias de arquitecturas FPGA y el tipo de dispositivo para que la herramienta de hardware sea ejecutada.

Para ejecutar el asistente BSB. es necesario primero ejecutar el navegador de proyecto XPS, y crear un proyecto con un sistema de procesador embebido. Al momento de abrir XPS, el software pedirá la creación de un nuevo proyecto utilizando o la apertura de uno ya existente.

Ahora que el asistente BSB ha empezado, se crea un proyecto usando las características descritas en la tabla 4.2.

El resultado final con los periféricos y núcleos habilitados en el proyecto se muestra en la figura 4.9. Adicionalmente, se crearon dos núcleos: Para habilitar la lectura de mensajes en la pantalla LCD de la ML605 (llamado *lcd_ip* en XPS) y el otro que es la FIFO que

Tabla 4.2: Pasos a ejecutarse con el asistente BSB

Pantalla del Asistente	**Propiedades del Sistema**	**Configuración / Comando de uso**
Bienvenido al Constructor de Sistema Base BSB	Opciones del tipo de proyecto	Seleccionar "*I would like to create a new design*"
Seleccionar Tarjeta	Seleccionar la tarjeta de desarrollo	Seleccionar "*I would like to create a system for a custom board*", *Board Vendor*: Xilinx, *Board name*: Vitex 6 ML605, *Board Revision*: D
Configuración del procesador Microblaze	Frecuencias de reloj	System clock frequency : 100 MHz Local Memory, 32 KB
Configuración de Periféricos	Dispositivos para Entrada y Salida	Se deshabilita *Ethernet MAC*, *Push Buttons* y *RS232* a 9600 baudios. *no parity*

alamacenará la información proveniente del *front-end* (llamado *fifogps* en XPS).

ISE ofrece una herramienta llamada *Core Generator* que permite al usuario crear núcleos específicos a sus necesidades. Con *Core Generator* se tiene una amplia gama de tipo de núcleos que van desde simples multiplexores y multiplicadores hasta núcleos de almacenamiento de memoria como es el caso de las FIFOs. La profundidad máxima de escritura en una FIFO es de 8184 bytes. Debido a esta peculiaridad, se serializan FIFOs para ir almacenando y recibiendo la información proveniente del *front-end*. Las tareas de adquisición y seguimiento necesitan el bloque completo de información para poder ser ejecutadas.

El *front-end* arroja los datos vía USB hacia la ML605. Para habilitar como entrada o salida de datos al periférico USB de la ML605, se debe agregar al sistema un Controlador de Periféricos Externo (EPC- External Peripherial Controller). En la figura 4.9 los núcleos que sirven para la comunicación vía USB son *xps_epc_0* y *util_bus_split_0*. Para referencia al proceso de configuración de esta tarea, ver [30].

El chip que contiene la terminal USB de la tarjeta, necesita arrancar el *firmware* desde una localidad de memoria, para poder establecer la comunicación entre el *front-end* y la tarjeta. El archivo **.bin* creado a partir del *fx2programmer*, se guarda en la *Tarjeta de Memoria Compact Flash*. XPS ofrece *SySace*, un interfaz entre el sistema y la *CompactFlash* para la

lectura y escritura de datos en la tarjeta de memoria.

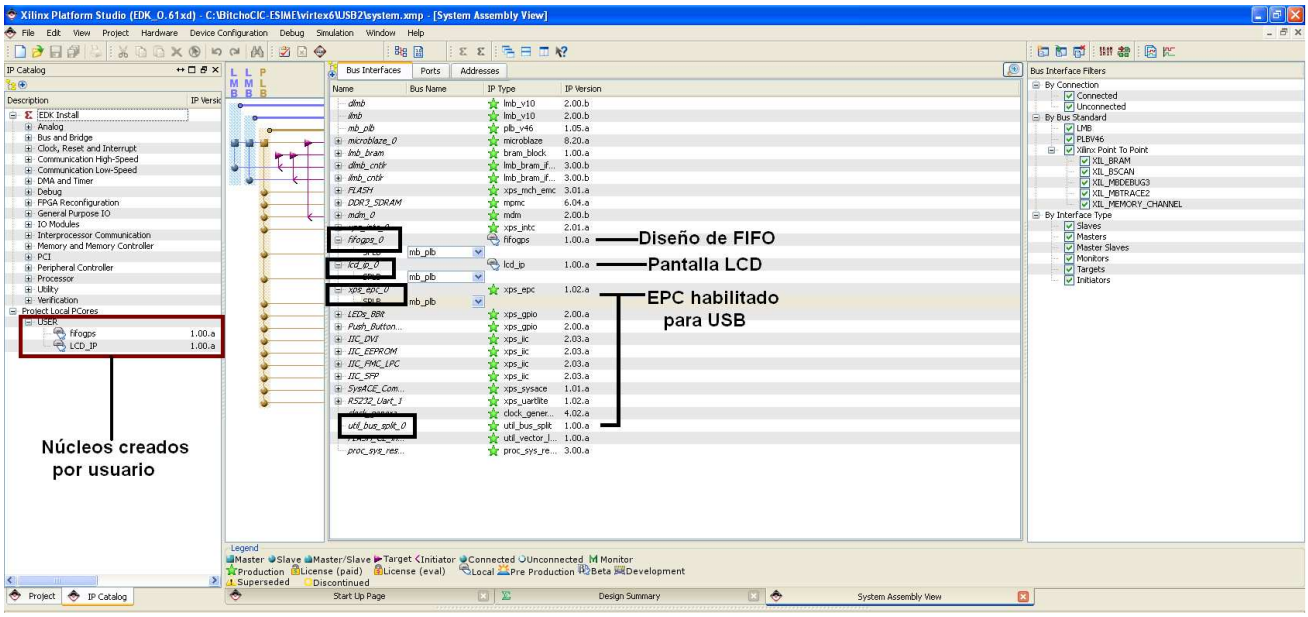


Figura 4.9: *Interface de usuario del XPS.*

Los núcleos IP del sistema se muestran en la figura 4.10. Todos los núcleos IP o los diseñados por el usuario están conectados al Bus Local del Procesador (PLB *Processor Local Bus*). El PLB es un Bus síncrono y contiene buses separados de lectura y escritura.

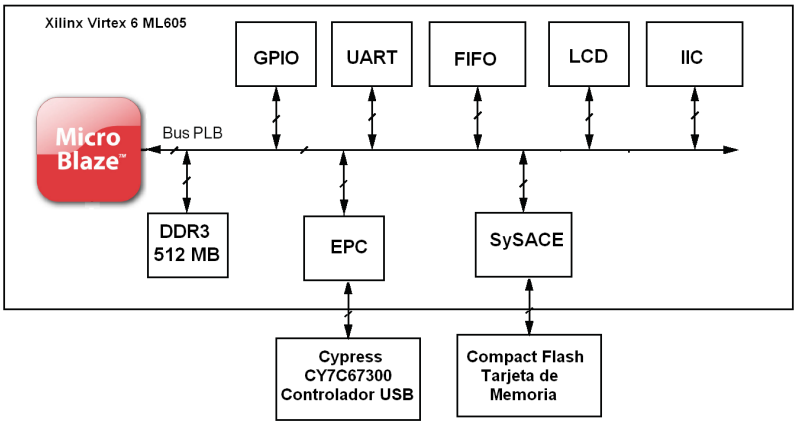


Figura 4.10: *Diagrama de bloques del Sistema.*

Una vez creado el proyecto, se tiene listo un árbol para el proyecto de software embebido. Ahora es momento de utilizar el SDK. En el ambiente XPS se da click en *Project* y luego a

Export Hardware Design to SDK. Aparece una ventana como en la figura 4.11. Se selecciona *Export and Launch SDK*. Al elegir a esta opción, se crea un directorio llamado "SDK" que tiene contiene todas las librerías, el mapa de direcciones de los núcleos presentes en el sistema y la descripción del sistema como se puede observar en la figura 4.12.

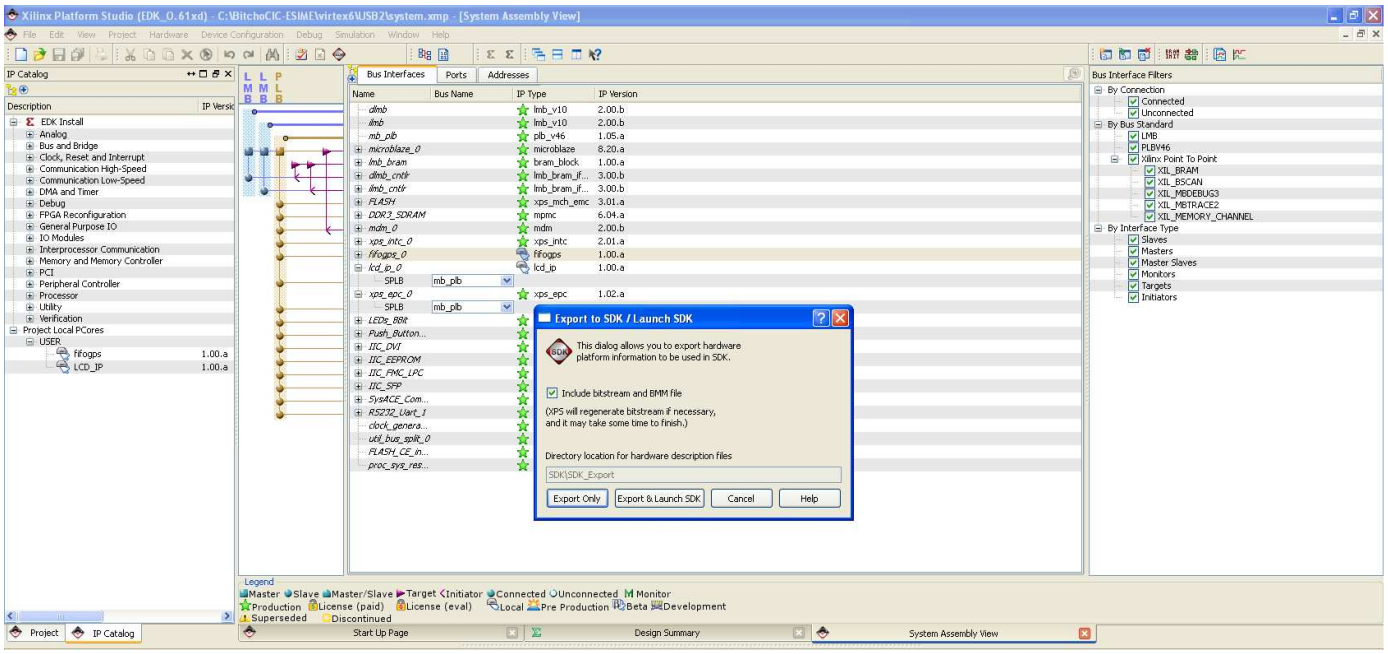


Figura 4.11: *Exportar diseño en Hardaware y abrir SDK*

4.5 Implementación del diseño

Se crea en SDK un proyecto llamado *empty_cpp_0* donde se añade cada uno de los bloques en software del receptor. La figura 4.13 muestra un esquema a groso modo de las librerías de las que depende *main.c* . La librería *fadqh1.h* realiza el proceso de adquisición y *ftrackh1.h* la tarea de seguimiento de la señal GPS.

Al compilar el código, se crea un archivo llamado *empty_cpp_0.elf*. Una vez teniendo completo el diseño, ahora se puede implementar en hardware. Para esto, se vuelve a XPS y en la pestaña de *project*, se añade el archivo **.elf* como se muestra en la figura 4.14.

Se conecta el cable USB-JTAG de la PC a la tarjeta y se selecciona dentro de XPS, *Device Configuration* y luego *Download Bitstream* como se muestra con el número 2 en la figura 4.14.

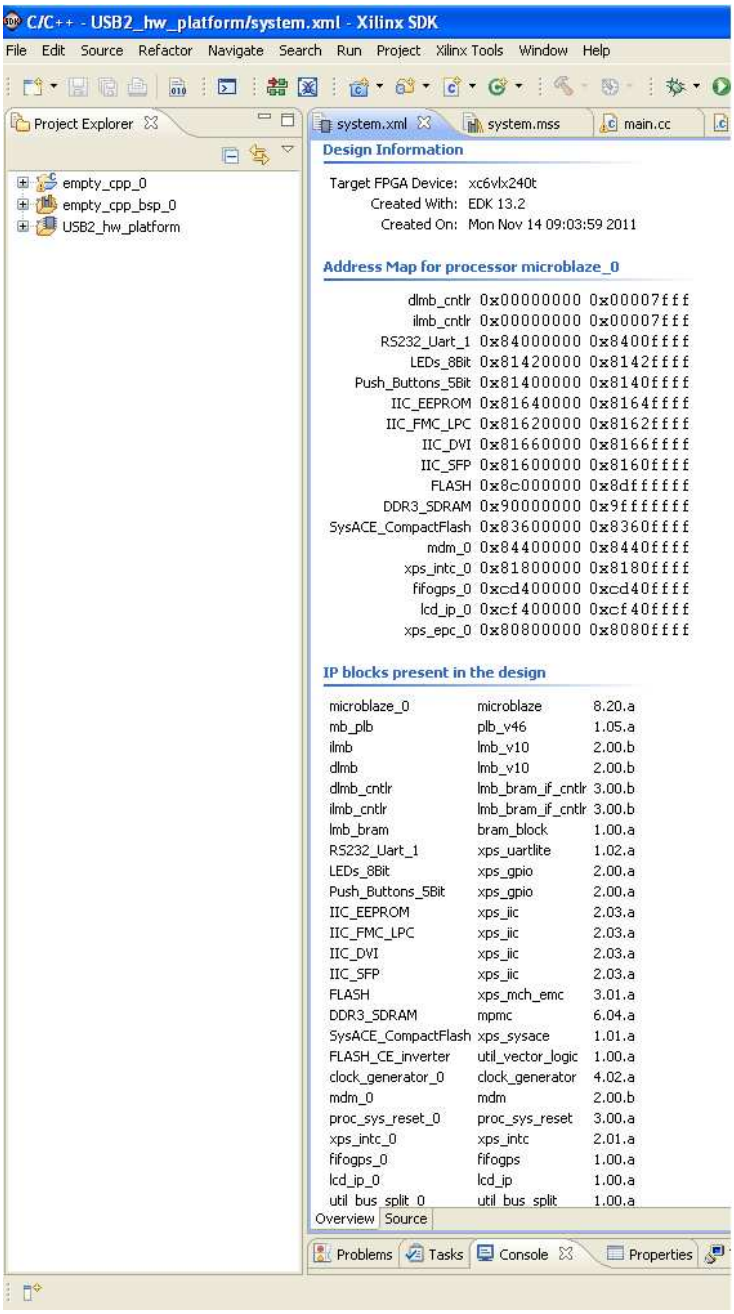


Figura 4.12: *Descripción del sistema en SDK*

Los resultados del cálculo de posición se pueden observar luego en la salida serial de la tarjeta (hacia la PC, utilizando el Hyperterminal) y sobre la pantalla LCD. El capítulo 5 muestra los resultados obtenidos luego de la implementación del diseño en hardware y software.

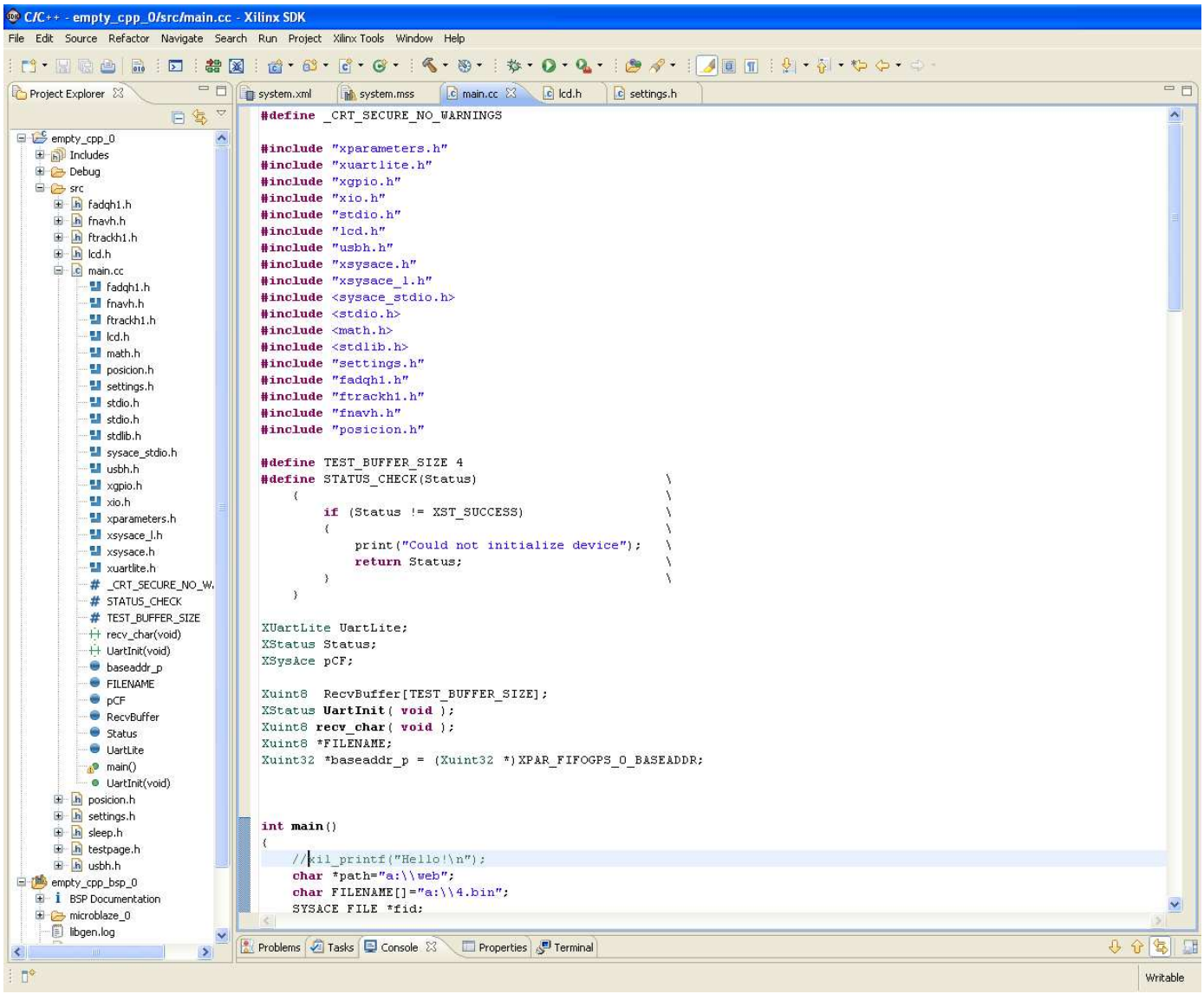


Figura 4.13: *Software del Sistema de Desarrollo*

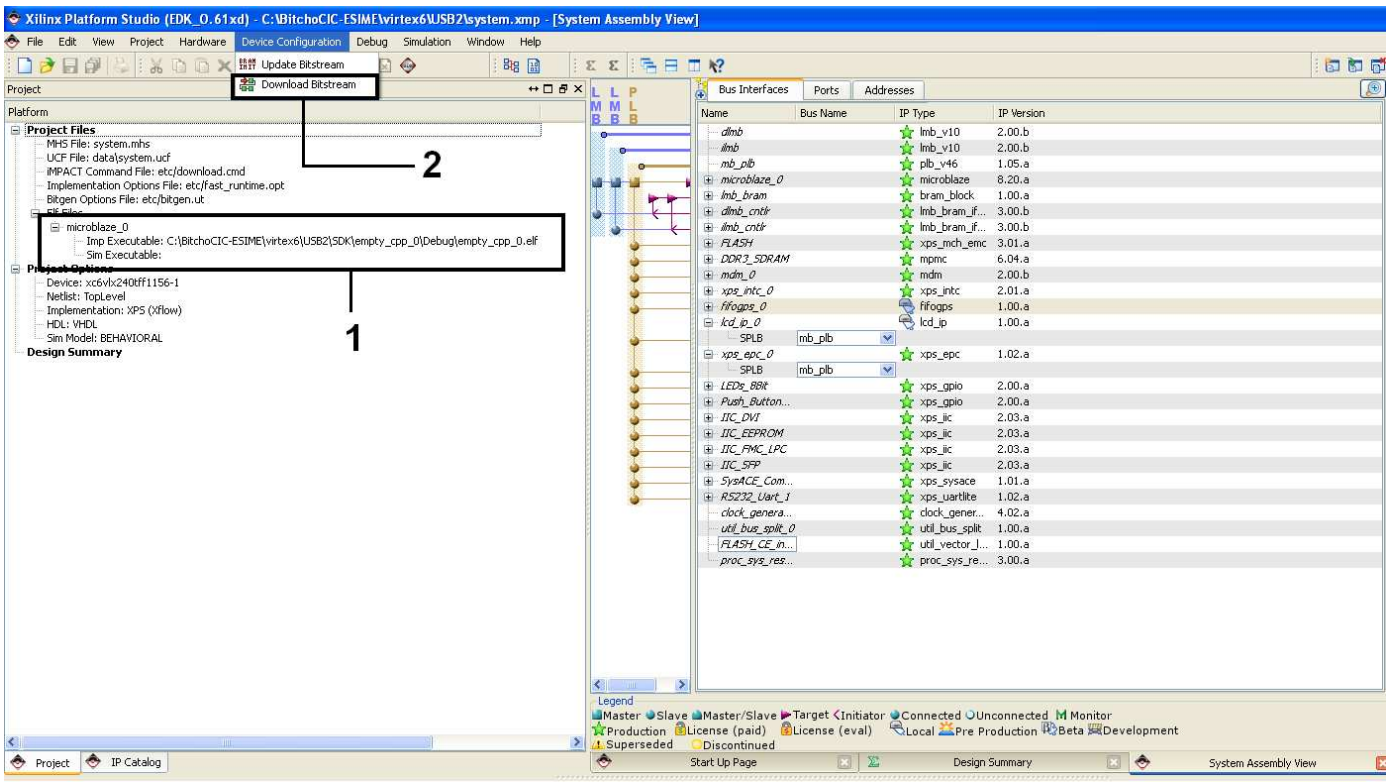


Figura 4.14: *Implementación del diseño en la ML605*

Capítulo 5

Pruebas y Resultados

Las actividades realizadas para anteceder el proyecto de tesis del receptor GNSS se enumeran:

- ⋆ Simulación y pruebas en C++ del algoritmo de cálculo de posición en Matlab.
- ⋆ Acoplar los lazos de adquisición y seguimiento a la tarea de cálculo de posición.
- ⋆ Emigrar cada uno de los bloques del receptor GPS a ambiente C++ para posteriormente asociarlo al FPGA.
- ⋆ Implementar cada uno de los bloques que conforman al receptor GPS hasta la tarea del cálculo de posición, desplegando estos mensajes en una pantalla LCD empotrada en la tarjeta Virtex6 ML605.

La figura 5.1 muestra un diagrama de cada etapa que conforma el sistema completo para lograr la meta final que es obtener y desplegar los mensajes de navegación.

5.1 Resultados de Adquisición y Seguimiento

5.1.1 Bloque de Adquisición

El *front-end* utilizado en este proyecto de tesis es el ofrecido por *SparkFun Electronics SiGe GN3S V2* [25]. Este *front end* ofrece un programa de captura basado en Windows que tiene un límite de 600 MB (ó 38.4 segundos) de captura de datos. Esto significa que se puede capturar una trama GPS completa. Debido a los límites de memoria y velocidades de acceso, es renuente pensar que se pueden capturar archivos más grandes a nivel de gigabytes. Las figuras 5.2 y 5.3 muestran la forma indicar el tiempo de captura de señal GPS.

Este archivo de entrada sirve tanto para el algoritmo realizado en Matlab como en C++. También, es de señalar, que es entrada tanto para el bloque de adquisición como el de

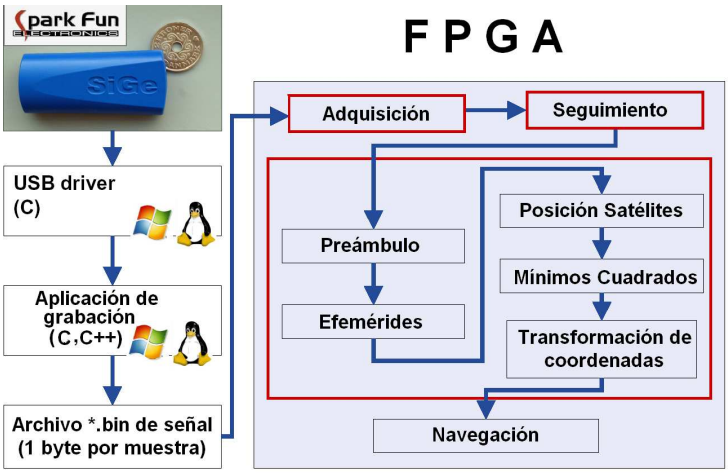


Figura 5.1: *Estructura en software del sistema.*

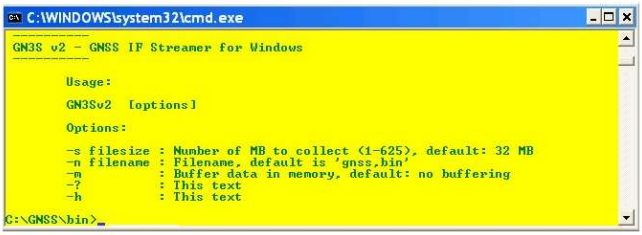


Figura 5.2: *Programa de Captura utilizado para SiGe GN3S V2*

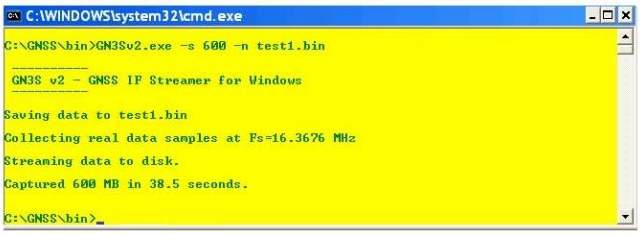


Figura 5.3: *Captura de 38.4 segundos de señal GPS*

seguimiento. Con el *front-end* se capturó 30 segundos de señal GPS para ser procesada, equivalente a 605 MB.

El propósito de la etapa de adquisición es encontar los satélites y valores gruesos de la portadora al igual que la fase del código de estas señales. Los satélites se diferencian por 32 secuencias PRN. Es importante conocer la frecuencia de la señal para poder generar una

portadora de señal local. La correlación circular y el uso de la Transformada Rápida de Fourier (FFT) es usada en este proceso.

La técnica realizada por *Cooley* y *Turkey* para la implementación de FFT en este proceso fué el elegido debido al bajo tiempo de cómputo que ofrece [31].

Utilizando *Matlab*, se visualiza la salida del bloque de adquisición. La figura 5.4 muestra los canales o satélites presentes en la señal, que particularmente corresponden a los PRN 14,18,21,24,29.

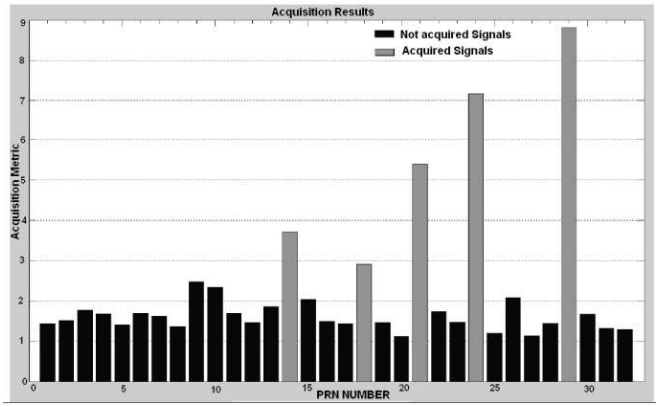


Figura 5.4: *Resultado de la etapa de adquisición de la señal GPS.*

La figura 5.4 indica el valor del máximo para cada fase del código C/A; esta figura muestra en color gris qué códigos PRN alcanzan un valor más alto que un umbral ya definido. Un valor de 2.4 es elegido para discriminar y obtener valores reales PRN contenidos en la señal GPS actual [15]. De acuerdo a este valor, los códigos PRN o satélites son adquiridos. Los códigos PRN en gris oscuro no fueron adquiridos o no están presentes en la señal recibida.

Al graficar la fase del código C/A y frecuencia de la señal, se logra un máximo que corresponde a un valor PRN generado internamente, así se tiene localizado que PRN está presente en la señal. La figura 5.5 es el resultado en Matlab para un sólo satélite visible contenido dentro de la señal GPS. Este proceso se repite con cada valor PRN hasta encontrar los canales o satélites restantes.

Luego de los gráficos en Matlab, a la par se ejecuta el algoritmo de navegación desarrollado en C++. La diferencia entre un ambiente y otro, son las funciones que Matlab ya tiene disponibles para ser ejecutadas. Un ejemplo claro es la función *fft()*. En este proyecto se implementó en C++ el algoritmo desarrollado por *Cooley* y *Turkey* para calcular la FFT.

Los resultados que arroja el bloque de adquisición se pueden apreciar en la figura 5.6. La correlación entre el código PRN generado internamente y el contenido en la señal GPS, da un valor llamado *Peak-Metric*. Este valor da una referencia de qué tan cercanos son ambos códigos. La fase del código y la frecuencia portadora se muestran también en la misma ventana.

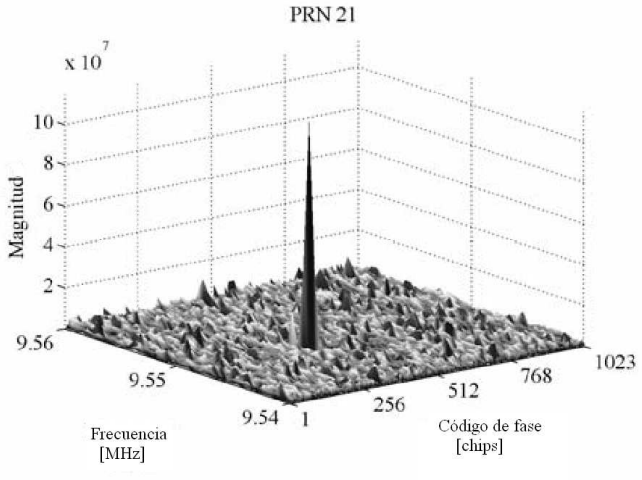


Figura 5.5: *Gráfica de Adquisición. Las señales generadas desde el PRN 21 están presentes en la señal recibida.*

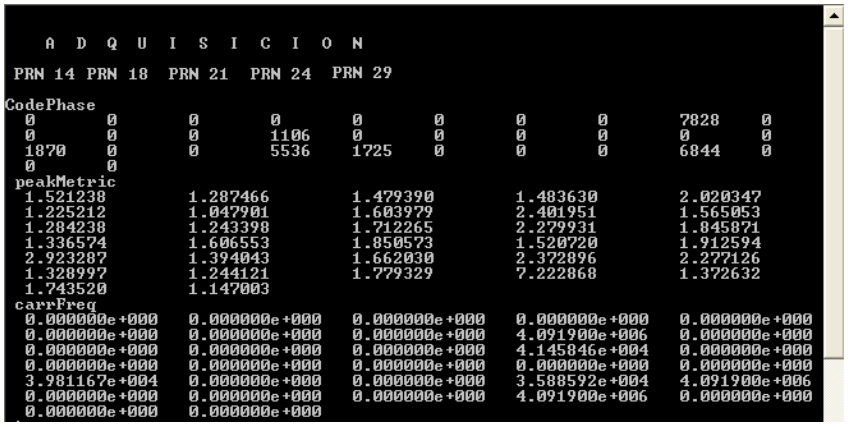
Los 32 valores correspondientes a *CodePhase*, *PeakMetric* y *carrFreq* son guardados dentro de una estructura llamada *acqResults*. Esta estructura, junto con el archivo **.bin* sirven de entrada para el siguiente bloque que es el de seguimiento.

5.1.2 Bloque de Seguimiento

El propósito principal de la etapa de seguimiento es refinar los valores gruesos de la fase del código y la frecuencia para mantener un seguimiento de como las propiedades de la señal cambian a través del tiempo. El bloque de seguimiento contiene dos partes: El seguimiento de código y seguimiento de portadora / código.

El seguimiento está ejecutandose continuamente para seguir los cambios en frecuencia como una función del tiempo. Sí el receptor pierde al satélite, una nueva adquisición se debe realizar para otro en particular.

Para demodular los datos de navegación exitosamente, hay que generar una réplica de la portadora. Para el seguimiento de la portadora, se utiliza un Bucle de Fase Cerrado (PLL) o un Bucle de Frecuencia Cerrado (FLL) [5].



```
   A D Q U I S I C I O N

PRN 14 PRN 18  PRN 21  PRN 24  PRN 29

CodePhase
  0      0       0       0       0       0       0       0       7828    0
  0      0       0       1106    0       0       0       0       0       0
  1870   0       0       5536    1725    0       0       0       6844    0
  0      0
peakMetric
  1.521238        1.287466        1.479390        1.483630        2.020347
  1.225212        1.047901        1.603979        2.401951        1.565053
  1.284238        1.243398        1.712265        2.279931        1.845871
  1.336574        1.606553        1.850573        1.520720        1.912594
  2.923287        1.394043        1.662030        2.372896        2.277126
  1.328997        1.244121        1.779329        7.222868        1.372632
  1.743520        1.147003
carrFreq
  0.000000e+000   0.000000e+000   0.000000e+000   0.000000e+000   0.000000e+000
  0.000000e+000   0.000000e+000   0.000000e+000   4.091900e+006   0.000000e+000
  0.000000e+000   0.000000e+000   0.000000e+000   4.145846e+004   0.000000e+000
  0.000000e+000   0.000000e+000   0.000000e+000   0.000000e+000   0.000000e+000
  3.981167e+004   0.000000e+000   0.000000e+000   3.588592e+004   4.091900e+006
  0.000000e+000   0.000000e+000   0.000000e+000   4.091900e+006   0.000000e+000
  0.000000e+000   0.000000e+000
```

Figura 5.6: *Valores PRN, Fase del Código, Peak-Metric y frecuencia portadora de la señal GPS.*

El módulo de seguimiento entrega diferentes señales para cada canal como se puede observar en la figura 5.7. La información de la señal I_P es la que interesa como entrada hacia la etapa del cálculo de posición. En esta señal, vienen contenidas las efemérides de cada satélite visible.

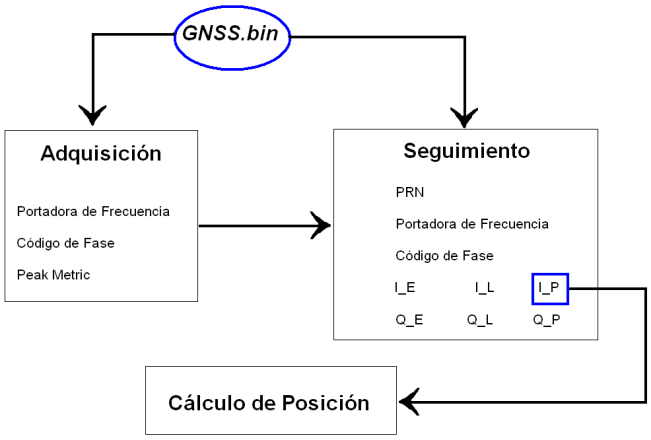


Figura 5.7: *Salida y entrada de datos requeridos para cada uno de los bloques que conforman al algoritmo de navegación*

5.2 Resultados del bloque de Cálculo de Posición

La tarea final del receptor es calcular la posición de usuario. usualmente cualquier receptor GNSS necesita al menos de cuatro satélites visibles para esta tarea. La posición se calcula en base a las pseudodistancias y a las posiciones de los satélites calculadas de las efemérides. El bloque de seguimiento entrega los datos de navegación para ser procesados y así, encontrar las efémetrides de cada uno de los satélites visibles en la señal.

La salida del ciclo de seguimiento es el valor en fase del módulo de seguimiento truncado a valores 1 y -1 como se observa en la figura 5.9. Teóricamente se podría obtener un valor de bit cada ms. Sin embargo, se está trabajando con señales débiles y con ruido, así que un valor promedio para 20 ms es calculado y truncado a -1 o 1. Un bit de navegación dura 20 ms.

La tasa de bits de los datos de navegación es de 50 bps. La tasa de muestreo de la salida del bloque de seguimiento es de 1000 sps, que corresponde a un valor cada ms. Antes de que los datos de navegación puedan ser decodificados, la señal proveniente del bloque de seguimiento debe ser convertido de 1000 sps a 50 bps. Esto significa que 20 valores consecutivos deben ser reemplazados por un 1. Este procedimiento se le conoce como *sincronización de bit*.

La primer tarea en el proceso de la sincronización de bits es encontrar el tiempo en la secuencia cuando una transición de bit ocurre. Primero, se localiza un cruce en cero. Un cruce en cero es donde el valor de bit cambia de 1 a -1 o viceversa. Cuando un cruce de cero es localizado, el tiempo de transición de un bit se ha encontrado. Cuando el tiempo de transición de un bit es conocido, es posible encontrar todos los tiempos de transición de bit. Estos son localizados 20 ms a partir del comienzo de la primera transición de bit detectada. La figura 5.8 muestra todas los tiempos de transición de bit en una secuencia de 200 ms. Los tiempos de transición de bit se han marcado con flechas.

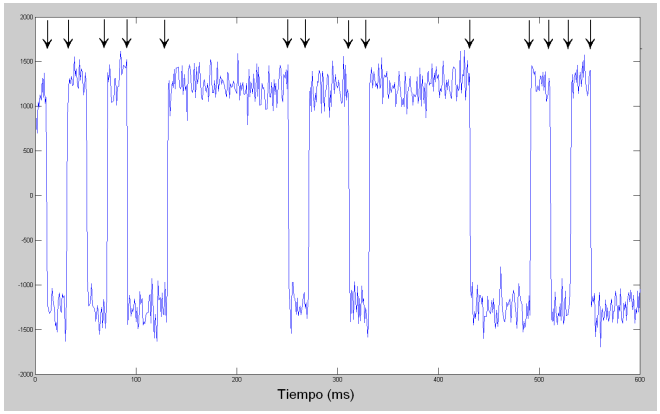


Figura 5.8: *Salida del bloque de Seguimiento. La señal actual es muy fuerte ya que una débil tendría valores cercanos a cero.*

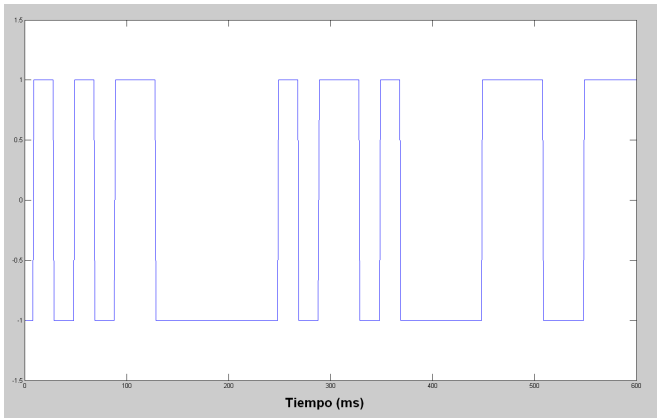


Figura 5.9: *Señal lista para la correlación con los bits de sincronización y encontrar el inicio de cada subtrama GPS.*

La señal se correlaciona con los bits de sincronización de inicio de cada subtrama GPS (preámbulo), dando como resultado la figura 5.10. Luego, se miden las diferencias entre cada uno de los máximos. Esta diferencia debe ser igual a 6000ms, valor de duración de una subtrama.

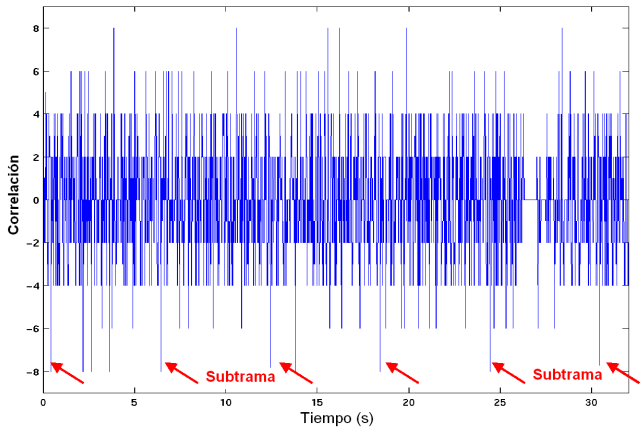


Figura 5.10: *Correlación de los datos de navegación con la palabra de sincronización para localizar cada una de las subtramas contenidas en la señal GPS.*

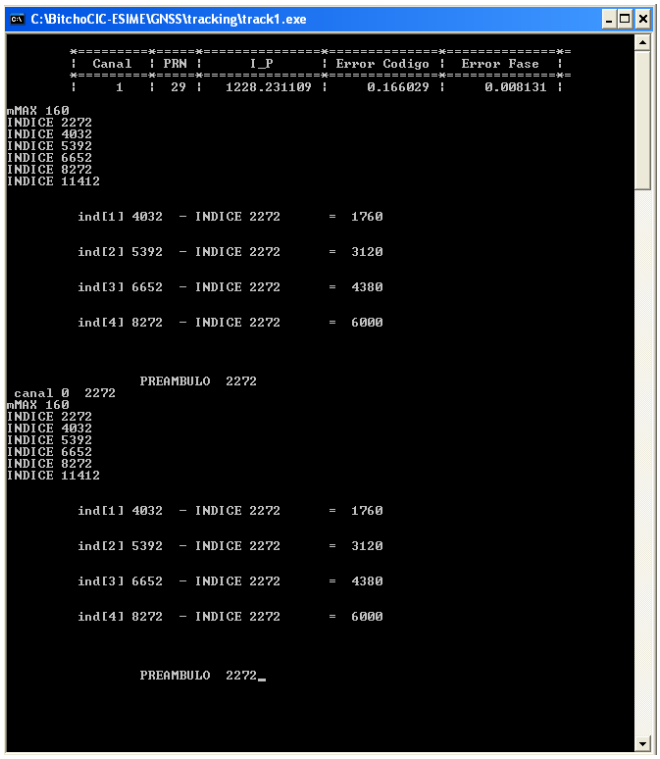


Figura 5.11: *Método para encontrar el incio de una trama GPS luego de la tarea de correlación.*

Los tiempos de inicio de una trama GPS son mostrados en la figura 5.12. Cada uno de

estos tiempos corresponde a cada canal. Este valor, indica el tiempo de llegada de una trama completa GPS. Este valor es importante ya que son base para el cálculo de pseudodistancias.

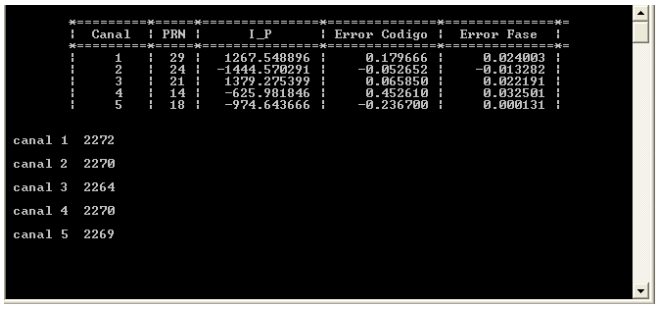


Figura 5.12: *Resultados de la etapa de correlación donde se muestra el comienzo de las tramas de cada canal.*

Una trama GPS la componen 5 subtramas y cada una de ellas contiene 300 bits de información. Luego, para cada canal, se empiezan a extraer las efemérides de cada satélite presente. Para realizar este proceso se requiere del manual *ICD-GPS-200C.pdf* [18]. La figura 5.13 ejemplifica esta tarea para un solo canal. En este ejemplo, se toma el primer canal (PRN 21) y despliega las efemérides del satélite visible. Este proceso se repite para todos los canales restantes. Para saber el significado métrico de cada variable que aparece, hágase referencia a la tabla 3.2.

El manual *ICD-GPS-200C.pdf* indica también el proceso a seguir para la estimación, en coordenadas XYZ (también conocidas como coordenadas ECEF), de la posición de cada uno de los satélites. Tarea que es la ulterior a ser ejecutada. La figura 5.14 muestra las coordenadas correspondientes al satélite 29. También, muestra las correcciones de reloj marcadas como *clk* así como la pseudodistancia del usuario al satélite 29 (*obs*).

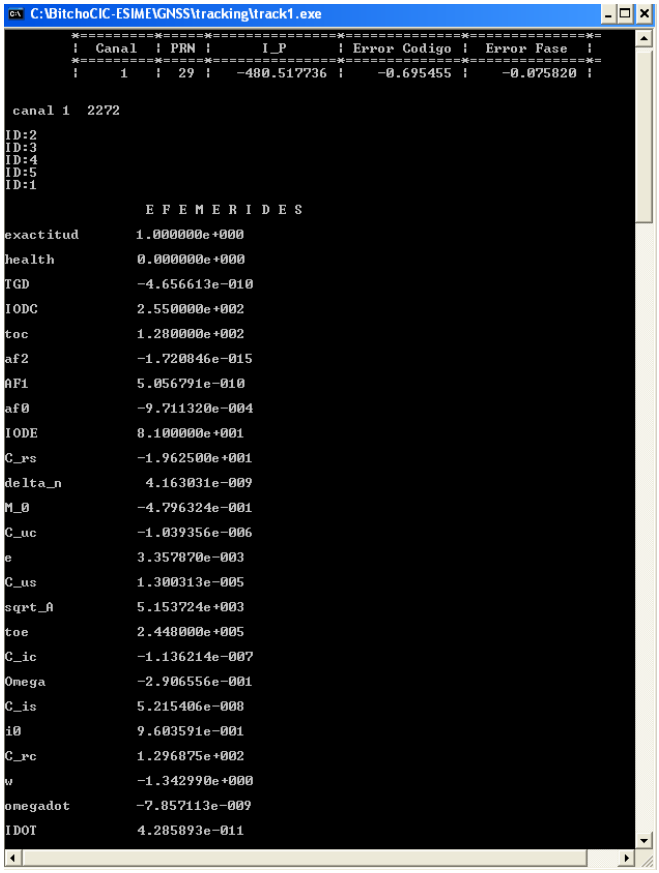


Figura 5.13: *Efémerides del canal 21.*

El cómputo de posición hasta ahora nos ha arrojado la posición de cada uno de los satélites visibles dentro de la señal GPS capturada. También, el cálculo de pseudodistancias y las correciones de reloj de cada satélite se han logrado hasta este momento. Siguiendo la figura 5.15, el método de mínimos cuadrados para el cálculo de posición de usuario está presto a ser ejecutado. La posición inicial de usuario es *(0,0,0)*.

Siete iteraciones fueron las realizadas para lograr el propósito de computar la posición de usuario en coordenadas ECEF (Figura 5.16). Para fines prácticos y de orientación, es inconsecuente dejar expresada de esta manera dicha posición. La última etapa que realiza el receptor, es la conversión de coordenadas ECEF a coordenadas angulares que utiliza la *Latitud* (Norte o Sur) y *Longitud* (Este u Oeste). Éste último proceso se observa en la figura 5.16.

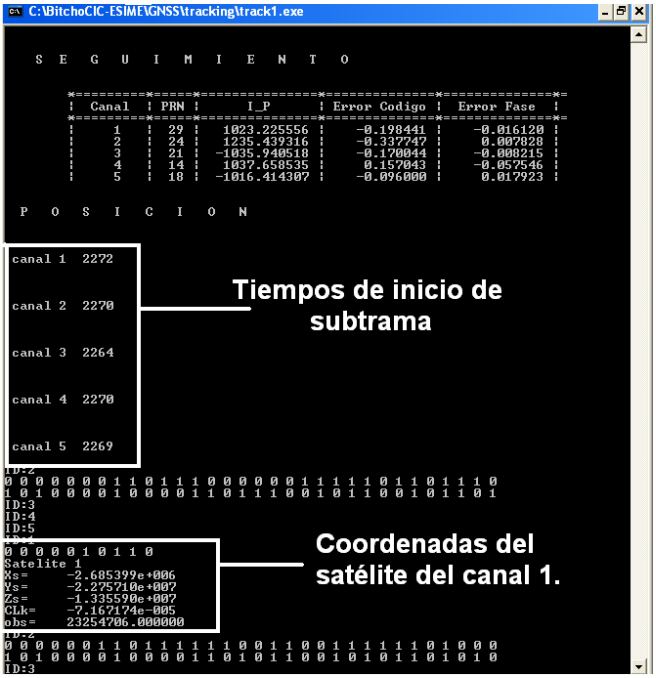


Figura 5.14: *Coordenadas para el satélite con PRN 29.*

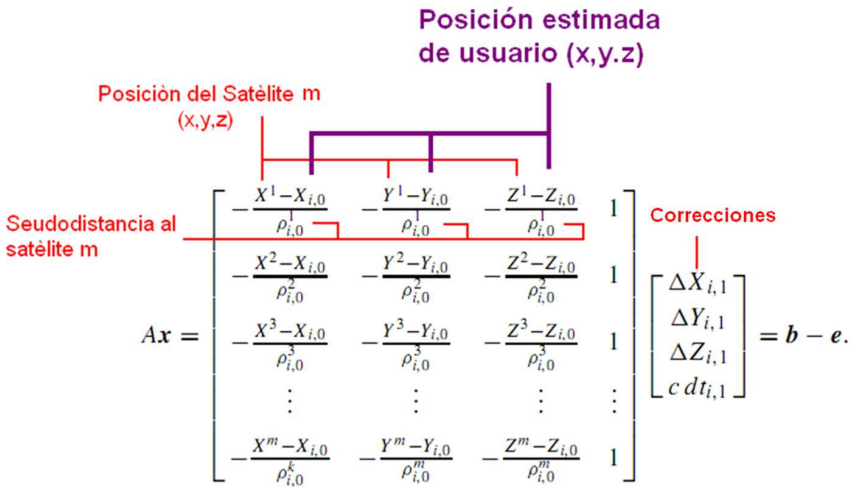


Figura 5.15: *Método de mínimos con las coordenadas de satélite y pseudodistancias como entradas requeridas.*

La figura 5.17 muestra la manera de como se recibe la señal GPS por medio del *front-end*. El *front-end* se conecta a la computadora vía USB y empieza a digitalizar la señal proveniente

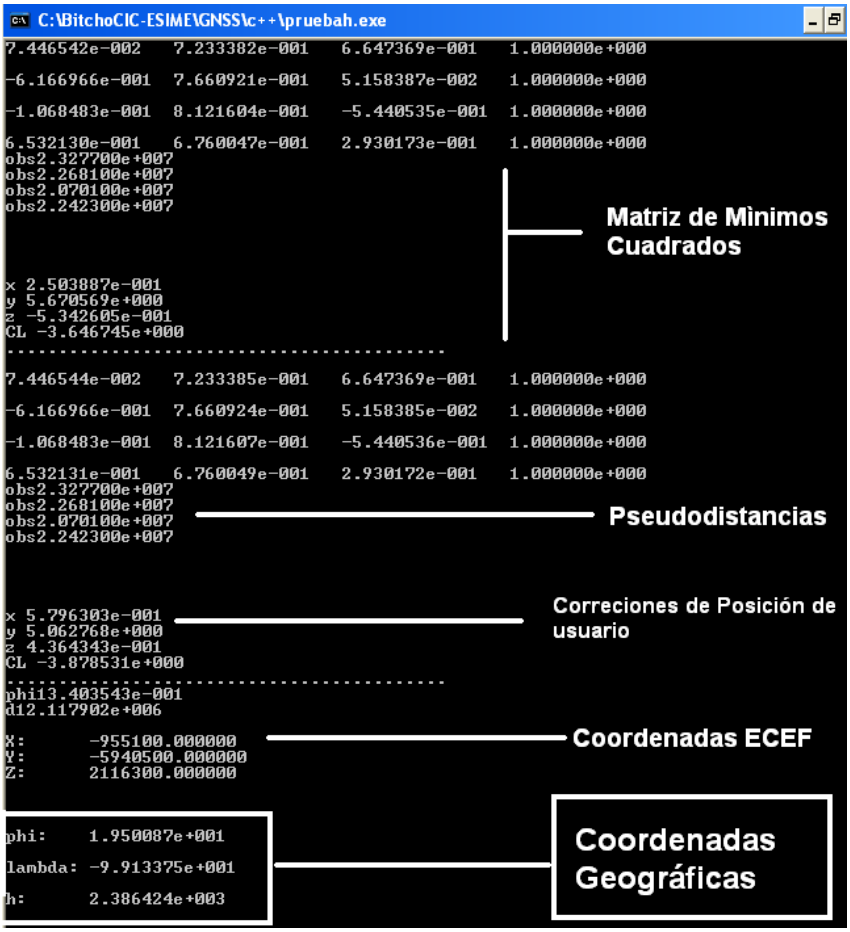


Figura 5.16: *Salida final del bloque de Cálculo de posición. Coordenas ECEF de usuario y conversión a coordenadas angulares.*

de la antena. Luego, este archivo que se genera se guarda en la *CompactFlash* con un lector de memorias comercial. Cuando la ML605 empieza la lectura del archivo y posteriormente la tarea de adquisición, en la *Hyperterminal* se despliegan los números PRN contenidos dentro de la señal. La figura 5.18 muestra la salida de los mensajes de navegación hacia una PC por vía serial.

La lectura de estos mensajes pueden leerse a su vez en la pantalla LCD empotrada en la ML605. La figura 5.19 hace referencia a ésto.

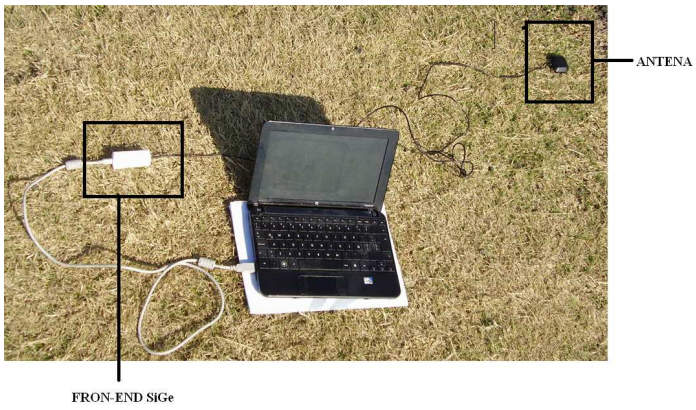


Figura 5.17: *Adquisición de la señal GPS con el front-end SiGe 4120.*

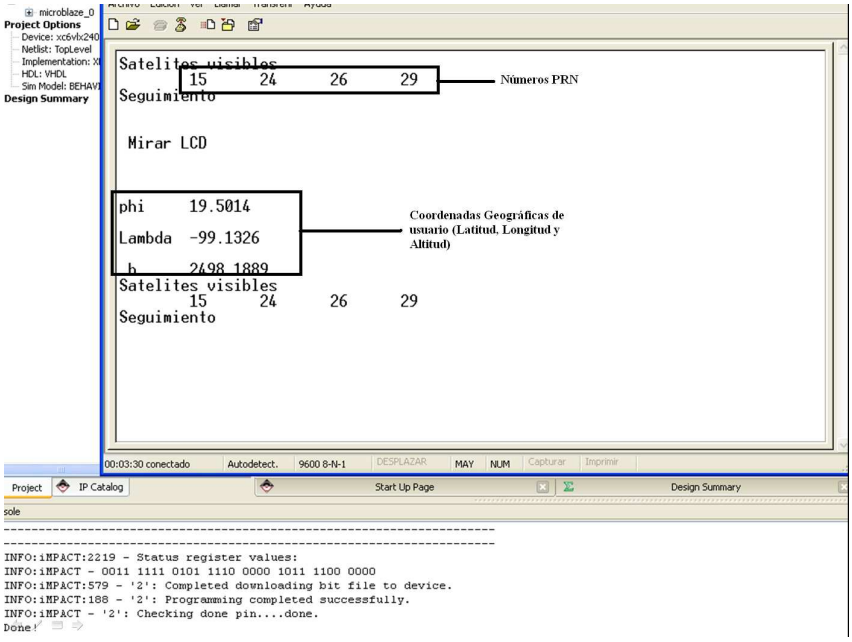


Figura 5.18: *Bloques de adquisición, seguimiento y cálculo de posición en la Hyperterminal.*

Dos señales más fueron capturadas dentro del Instituto Politécnico Nacional para corroborar el funcionamiento del receptor. La figura 5.20 muestra los mensajes de navegación a partir de la señal recibida en el estacionamiento de ESCOM (Escuela Superior de Cómputo) y el CIC (Centro de Investigación en Computación). Para reafirmar estos mensajes, se recurrió a la herramienta *Google-Earth* cuyo mapa se muestra en la parte inferior de la figura 5.20.

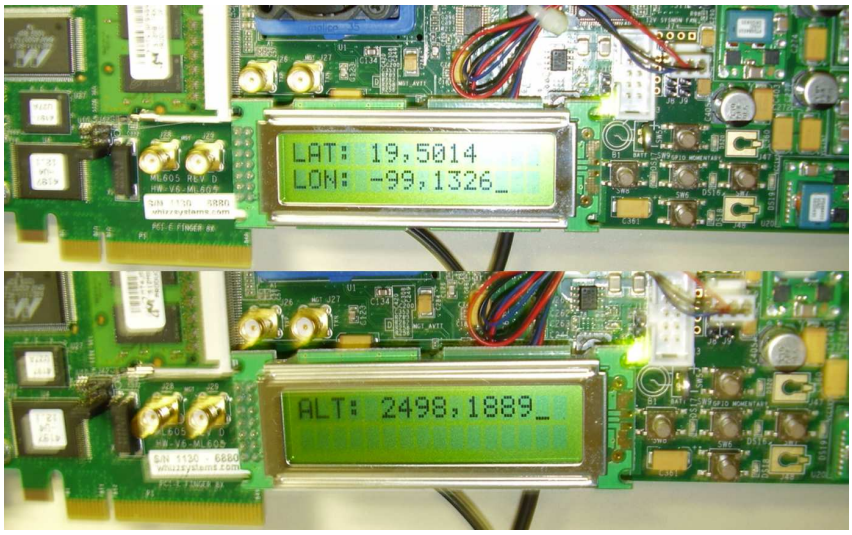


Figura 5.19: *Lectura de los mensajes de navegación en la pantalla LCD*

Una señal adicional fué capturada en otro punto del Instituto Politécnico Nacional. Los mensajes de Latitud, Altitud y Longitud ahora se mandan vía serial hacia una PC y se muestran en la figura 5.21. También, la figura 5.21 muestra la localización de este punto utilizando *Google Earth* y así ratificar la posición obtenida por el receptor GPS desarrollado. En esta figura, la posición se indica por las coordenadas obtenidas desde el receptor. Este sitio corresponde al laboratorio donde este proyecto ha sido desarrollado y la señal GPS ha sido capturada.

Como prueba adicional, se tomó una señal en otro sitio fuera del Instituto Politécnico Nacional, más concretamente en la localidad de Irapuato Guanajuato. La figura 5.22 muestra los resultados en la pantalla LCD de la tarjeta y a su vez, la posición geográfica utilizando *Google Earth*.

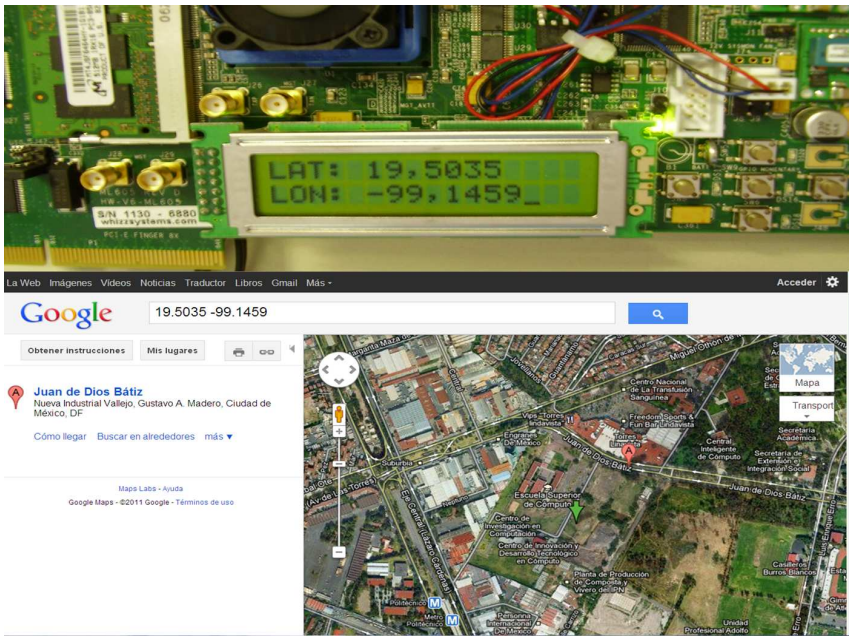


Figura 5.20: *Posición de usuario en el CIC del Instituto Politécnico Nacional.*

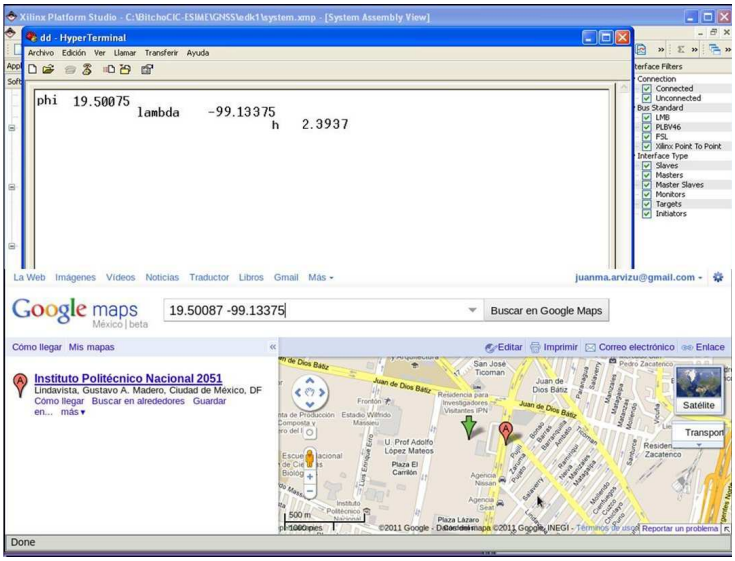


Figura 5.21: *Posición de usuario utilizando Google Earth para la señal GPS adquirida.*

Los recursos que utiliza el sistema se muestran en la figura 5.23. En esta figura se puede notar que el sistema todavía tiene la capacidad de que se le puedan agregar otros núcleos e incluso habilitar otros periféricos de entrada y salida para fines particulares de otro usuario.

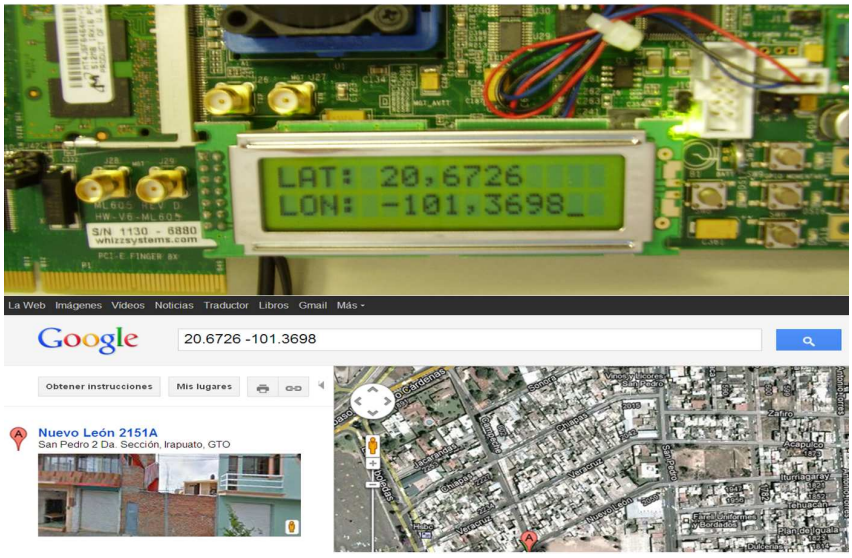


Figura 5.22: *Posición de usuario utilizando Google Earth para la segunda señal GPS adquirida.*

Device Utilization Summary (actual values)			
Slice Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	8,625	301,440	2%
Number of Slice LUTs	7,262	150,720	4%
Number used as logic	6,297	150,720	4%
Number used as Latch-thrus	0		
Number used as Memory	610	58,400	1%
Number of occupied Slices	3,812	37,680	10%
Number with an unused Flip Flop	2,522	10,292	24%
Number with an unused LUT	3,030	10,292	29%
Number of fully used LUT-FF pairs	4,740	10,292	46%
Number of slice register sites lost to control set restrictions	1,865	301,440	1%
Number of bonded IOBs	186	600	31%
Number of LOCed IOBs	164	186	88%
Number of RAMB36E1/FIFO36E1s	87	416	20%
Number of RAMB18E1/FIFO18E1s	2	832	1%
Number of BUFG/BUFGCTRLs	5	32	15%
Number of ILOGICE1/ISERDESE1s	58	720	8%
Number of OLOGICE1/OSERDESE1s	175	720	24%
Number of BSCANs	1	4	25%
Number of BUFHCEs	0	144	0%
Number of BUFIODQSs	4	72	5%
Number of BUFRs	2	36	5%
Number of LOCed BUFRs	2	2	100%
Number of CAPTUREs	0	1	0%
Number of DSP48E1s	3	768	1%
Number of EFUSE_USRs	0	1	0%
Number of FRAME_ECCs	0	1	0%
Number of GTXE1s	0	20	0%
Number of IBUFDS_GTXE1s	0	12	0%
Number of ICAPs	0	2	0%
Number of IDELAYCTRLs	2	18	11%
Number of IODELAYE1s	46	720	6%
Number of LOCed IODELAYE1s	6	46	13%
Number of MMCM_ADVs	1	12	8%
Number of PCIE_2_0s	0	2	0%
Number of STARTUPs	1	1	100%
Number of SYSMONs	0	1	0%
Number of TEMAC_SINGLEs	0	4	0%

Figura 5.23: *Recursos utilizados por el sistema desarrollado.*

Capítulo 6

Conclusiones y Trabajo a Futuro

Tal como se indicó al inicio de esta tesis, el objetivo principal de esta tesis era "Desarrollar un prototipo de un receptor GPS capaz de decodificar los mensajes de altitud, longitud, latitud y tiempo, basado en el uso de dispositivos FPGA".

Los lazos de adquisición [5] y seguimiento [14] que se desarrollaron en trabajos preevios para otros sistemas de desarrollo, fueron ajustados para concluir este trabajo de tesis. Incluso el bloque de adquisición se volvió a elaborar para que fuera posible su adaptación al sistema actual.

Luego del proceso anterior y del cálculo de posición tanto en hardware como en software, las características del sistema desarrollado son las siguientes:

- Se obtiene un receptor GPS reconfigurable, capaz de desplegar cualquier mensaje de navegación u otros afines a las necesidades cualquier usuario. La trama GPS no solamente entrega los datos de efemérides, sino también almanaques y el modelo ionosférico [18].
- Receptor GPS programable ya que si se requiere de algún cambio en el aplicación de éste, la modificación es completamente por vía software sin alterar algun componente en hardware.
- El sistema desarrollado actualmente despliega los mensajes de navegación de Latitud, Altitud y Longitud, pero por las caracterśticas preevias, el receptor puede arrojar cualquier otro mensaje que sea requerido.
- Por la particulariedad de que la trama GPS tiene una duración de 30 segundos, se recomienda tomar los datos a partir de este tiempo. Luego de este lapso, la actualización de los mensajes de navegación es cada segundo.
- La plataforma de desarollo del FPGA soportó la implementación del sistema, otorgando además una gran flexibilidad en la manera de programación.

Los objetivos que se propuesieron para este trabajo de tesis fueron cubiertos satisfactoriamente considerando en todo momento las características del sistema propuesto. Este sistema puede ser explotado para otras aplicaciones afines al tema de Sistemas de Navegación. La notoriedad que le da de que todo el procesamiento digital de la señal es a base de software, es posible realizar el tratamiento de otro tipo de señales satelitales de navegación.

Debido a que las señales GNSS comparten similtudes unas de otras, es posible ajustar este receptor para que puede emigrar de un sistema de navegación a otro.

Como trabajo a futuro se debe explorar el área de simulación e implementación de otros sistemas de navegación sin necesidad de cambios en aspectos de hardware, utilizando y actualizando algunos parámetros en el software ya elaborado. Este nuevo sistema puede también ser implementado en la tarjeta Virtex 6 ML605.

Asimismo, se recomienda abordar en algún trabajo futuro la interfaz de comunicación entre los puertos USB. Debido a la alta velocidad en la transferencia de datos, en el sistema desarrollado es importante que no haya pérdidas de información. El sistema debe ser capaz de pedir de forma autonóma la señal digitalizada al *front-end* sin restricciones de tiempo ya que una de las características del *SiGe 4120* es que arroja paquetes de datos con duración de tiempo ya establecita (34 segundos). La re-programación del chip del *front-end* es posible [22]. Este tema ya ha sido planteado en varios trabajos [32] [26].

Apéndice A

Código C++ para Cálculo de Posición

```
#define _CRT_SECURE_NO_WARNINGS

#include "xparameters.h"
#include "xuartlite.h"
#include "xgpio.h"
#include "stdio.h"
#include "lcd.h"
#include "xsysace.h"
#include <sysace_stdio.h>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include"settings.h"
#include "ftrackh1.h"
#include “lcd.h”

int main()
{
		int a,loopCnt,*xtrack[15],PRN=0,*caCodesTable[32],in=0;
    static seguimient *trackResults;
    int j=0,k,i,numsat,**data,*data1,*data2;
    int *firstSubFrame,*navBits,*navBitSamples,dataAdaptCoeff,samplesPerCode;
    static efemerides eph[15];
    static Sposicion XS[15],posxyz;
    static acquisition acqResults;
   float TOW;
   long lSize;
   double tiempo[15],Clk,ck[3];
   static coordenadas coords;
   float obs[15],pos[3];
   signed char *signal,carac[100];
   long lsize;
   double **data3,**caTable,**caCodeFreqDom;
   FILE*    fid;


/// INICIAR LCD
XromLCDInit();
XromLCDOn();
XromLCDClear();

    samplesPerCode = (int)round(samplingFreq /(codeFreqBasis / codeLength))

       if (fileType==1)
        dataAdaptCoeff=1;
       else
        dataAdaptCoeff=2;

            signal=( char*)malloc(dataAdaptCoeff*11*samplesPerCode*sizeof( char));
            data1=(int*)malloc(dataAdaptCoeff*11*samplesPerCode/2 * sizeof(int));
            data2=(int*)malloc(dataAdaptCoeff*11*samplesPerCode/2 * sizeof(int));

                                                                                                                    /*  setup
Flashcard   */

Status=XSysAce_Initialize(&SysAce, XPAR_SYSACE_0_DEVICE_ID);

if(Status!=XST_SUCCESS)

xil_printf("ERROR INICIALIZAR\r\n");

return XST_FAILURE;


else

		xil_printf("\r\nSYSACE  INICIALIZADO \r\n");


fid= sysace_fopen(FILENAME,"r");

if(fid!=NULL)

xil_printf("Se ha abierto el archivo\r\n");

else
```

```
xil_printf("No se ha podido abrir el archivo\r\n");
    numCharsRead=sysace_fread((char*)signal,1,180084,fid);
    if (numCharsRead <= 0){
              xil_printf("\r\nError reading from system ace (%d)\r\n", numCharsRead);
        return -1;
       }
    else
              xil_printf("\r\nLECTURA correcta %d\r\n", XSA_CF_SECTOR_SIZE);


    xil_printf("    A  D  Q  U  I  S  I  C  I  O  N \n\n " );
    acqResults = adquisicion(data,dataAdaptCoeff);

   xil_printf("    S  E   G   U   I   M   I   E   N   T   O \n\n");

    trackResults=tracking(filename);
                            for(k=0;k<numberOfChannels;k++)
                            {
                   xtrack[k]= (int *)malloc(msToProcess * sizeof(int));
          for(j=0;j<msToProcess;j++)
            if(trackResults[k].I_P[j]<0)
             *(xtrack[k]+j)=-1;
            else
             *(xtrack[k]+j)=1;
        }

   xil_printf("  P   O   S   I   C   I   O   N  \n\n");
     for(numsat=0;numsat<numberOfChannels;numsat++) {
      firstSubFrame[numsat]=preambulo(xtrack,numsat);
      xil_printf("\n\n canal %d  %d\n\n",numsat+1, firstSubFrame[numsat] );
     }

    for(numsat=0;numsat<numberOfChannels;numsat++) {
     for(i=0;i<(Trama*20)+20;i++)
       navBitSamples[i]=xtrack[numsat][ (firstSubFrame[numsat] -1-20) + i     ];

      for(i=0;i<(Trama+1);i++)
         if(navBitSamples[i*20]<0)
         navBits[i]=0;
         else
         navBits[i]=1;

     eph[numsat]=Cefemerides(navBits);
   xil_printf("\n \t \t E F E M E R I D E S \n\n");
      xil_ printf("exactitud \t\t%e\n\n",eph[numsat].exactitud);
      xil_printf("health \t\t%e\n\n",eph[numsat].health);
      xil_printf("TGD \t\t%e\n\n",eph[numsat].TGD);
      xil_printf("IODC \t\t%e\n\n",eph[numsat].IODC);
      xil_printf("toc \t\t%e\n\n",eph[numsat].toc);
      xil_printf("af2 \t\t%e\n\n",eph[numsat].af2);
      xil_printf("AF1 \t\t%e\n\n",eph[numsat].af1);
      xil_printf("af0 \t\t%e\n\n",eph[numsat].af0);
      xil_printf("IODE \t\t%e\n\n",eph[numsat].IODE);
      xil_printf("C_rs \t\t%e\n\n",eph[numsat].C_rs);
      xil_printf("delta_n \t\t%e\n\n",eph[numsat].delta_n);
      xil_printf("M_0 \t\t%e\n\n",eph[numsat].M_0);
      xil_printf("C_uc \t\t%e\n\n",eph[numsat].C_uc);
      xil_printf("e \t\t%e\n\n",eph[numsat].e);
      xil_printf("C_us \t\t%e\n\n",eph[numsat].C_us);
      xil_printf("sqrt_A \t\t%e\n\n",eph[numsat].sqrt_A);
      xil_printf("toe \t\t%e\n\n",eph[numsat].toe);
      xil_printf("C_ic \t\t%e\n\n",eph[numsat].C_ic);
      xil_printf("Omega \t\t%e\n\n",eph[numsat].Omega);
      xil_printf("C_is \t\t%e\n\n",eph[numsat].C_is);
      xil_printf("i0 \t\t%e\n\n",eph[numsat].i0);
      xil_printf("C_rc \t\t%e\n\n",eph[numsat].C_rc);
      xil_printf("w \t\t%e\n\n",eph[numsat].w);
      xil_printf("omegadot \t\t%e\n\n",eph[numsat].omegadot);
      xil_printf("IDOT \t\t%e\n\n",eph[numsat].IDOT); */

      obs[numsat]=pseudorangos(Clk,trackResults,firstSubFrame,numsat);

                              xil_printf("Satelite %d\n",numsat+1);
                              xil_printf("Xs=\t%e\n",XS[numsat].x);
                              xil_printf("Ys=\t%e\n",XS[numsat].y);
```

```
                        xil_printf("Zs=\t%e\n",XS[numsat].z);
                        xil_printf("CLk=\t%e\n",XS[numsat].ClkCorr);
                        xil_printf("obs=\t%f\n",obs[numsat]);
    }

    posxyz=minimos(XS,obs, numberOfChannels);
    pos[0]=posxyz.x;
    pos[1]=posxyz.y;
    pos[2]=posxyz.z;

    coords=to2geod(pos);

     who=coords.phi;
            tho=(coords.phi-who);
            if(tho<0)
            tho=tho*(-1);
 xil_printf("\r\nphi\t%d.%3d\r\n",who,tho);
 XromLCDPrintString("LAT: ");
 XromLCDPrintInt(who);
 XromLCDPrintString(",");
 XromLCDPrintInt(tho);
 XromLCDSetLine(2);

 who=coords.lambda;
            tho=(coords.lambda-who);
            if(tho<0)
            tho=tho*(-1);
xil_printf("\r\nLambda\t%d.%3d\r\n",who,tho);
who=0;
tho=0;
who=99;
tho=1326;
 XromLCDPrintString("LON: -");
 XromLCDPrintInt(who);
 XromLCDPrintString(",");
 XromLCDPrintInt(tho);
 usleep(700000);
 XromLCDClear();
 XromLCDSetLine(1);

 who=coords.h;
            tho=(coords.h-who);
            if(tho<0)
            tho=tho*(-1);
    xil_printf("\r\n h\t%d.%3d\r\n",who,tho);
             XromLCDPrintString("ALT: ");
             XromLCDPrintInt(who);
             XromLCDPrintString(",");
             XromLCDPrintInt(tho);

  for(k=0;k<numberOfChannels;k++)
  free(xtrack[k]);

  free(firstSubFrame);
  free(navBits);
  free(navBitSamples);
 fclose(fid);
 free(signal);
 free(data);
 free(data1);
 free(data2);
  getchar();
}


////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
                //*******                LIBRERIA DE ALGORITMOS DE NAVEGACION *********/
#ifndef     _fnavh_H_
#define     _fnavh_H_

#include "settings.h"
typedef struct{                                                         // estructura de efemerides
            double weeknumer;
             double exactitud;
             double health;
             double TGD;            ///Subtrama 1
```

```
                double IODC;
                double toc;
                double af2;
                double af1;
                double af0;
                /////////
                double IODE;
                double C_rs;
                double delta_n;
                double M_0;
                double C_uc;                //Subtrama 2
                double e;
                double C_us;
                double sqrt_A;
                double toe;
                /////////
                double C_ic;
                double Omega;
                double C_is;
                double i0;                  /// Subtrama 3
                double C_rc;
                double w;
                double omegadot;
                double IDOT;
                double TOW;
} efemerides;

double bin2dec(int *bin,int inicio,int len);
double bin2dec3(int *bin,int inicio1,int len1, int inicio2, int len2);
double twosComp2dec(int *bin,int inicio,int len);
double twosComp2dec3(int *bin,int inicio1,int len1, int inicio2, int len2);
int *checkphase(int *sub,int nbit0);

int preambulo(int *xtrack[15],int numsat)                                   /////// Preambulo de cada subtrama     //////
{
   int in=0,a=0,m=0, i=0,j=0, k=0,l=5,index[200],index2=0,mm=0,ia=0 ;
            int xp[]={1, -1, -1, -1, 1, -1, 1, 1},*pms,n,*x,p=0,*axp;

            /*Calcula el promedio de las dos series*/
   n=msToProcess;
            l=n;

   axp= (int *)malloc(msToProcess*2 * sizeof(int));
   pms= (int *)malloc(msToProcess * sizeof(int));
   x= (int *)malloc(msToProcess * sizeof(int));

            for(i=0;i<8;i++)
   for(k=i*20;k<(i+1)*20+1;k++)
   pms[k]=xp[i];

   for(i=160;i<msToProcess;i++)
   pms[i]=0;

   for(i=0;i<msToProcess;i++)
  x[i]=xtrack[numsat][i];

  for(i=0;i<msToProcess*2;i++)
  axp[i]=0;

  for(i=0;i<length(index);i++)
  index[i]=0;

            for(i=0;i<n*2;i++)
                  for(k=i;k<n;k++)
                                    axp[i+l]+=x[k]*pms[k-i]  ;

for(i=0;i<n*2;i++)
    mm=max(abs(axp[i]),mm);

mm=0    ;
  for(i=0;i<n*2;i++)
  {
   mm=max(abs(axp[i]),mm);
   if(mm>159)
   {
     index[in]=i - msToProcess + 1;
```

```
      mm=0;
      in++;
    }
  }

 for(i=0;i<in;i++)
  for(j=i+1;j<in;j++)
  {
   index2=index[j] - index[i];
   if(index2==6000)
   {
    index2=index[i];
    return index2;
   }
  }

   free(axp);
   free(pms);
   free(x);
   return index2;
}

/*************************    E F E M E R I D E S     ***********************/
efemerides Cefemerides(int *nbits)                                              ////// Extraccion de las efemerides
//////////
{
                        int *subtrama,subtramaID=0, i;
                        double gpsPi                    = 3.1415926535898,aa=0;  //Pi usado en GPS (WGS-84)
                        static efemerides eph;
                        int nbit0;
                        int *subsub,j,M,a,k;

     nbit0=nbits[0];

      subsub= (int *)malloc(30 * sizeof(int));
      subtrama= (int *)malloc(300 * sizeof(int));

     for(k=0;k<300;k++)
       subtrama[k] = 0;

            for(i=0;i<numSubTramas;++i)                                                 //cortar las 5 tramas
            {
                                for(k=0;k<300;k++)
          subtrama[k] = nbits[300*i + k +1];

         for(j=0;j<10;j++)
         {
           for(k=1;k<31;k++)
             subsub[k-1]=subtrama[30*j+k -1];

           subsub=checkphase(subsub, nbit0);
           nbit0=subtrama[30*j +30 -1];    ///indice 29 pero en realidad es el 30

           for(k=1;k<31;k++){
             subtrama[30*j+k -1]=subsub[k-1];

           }

         }
                                subtramaID=(int)bin2dec(subtrama,49,3);
                        printf("ID:%d\n",subtramaID);

                                switch (subtramaID)
                                {
                                        case 1:
                                                        eph.weeknumer=bin2dec(subtrama,60,10)+1024;
                                                        eph.exactitud=bin2dec(subtrama,72,4);
                                                        eph.health=bin2dec(subtrama,76,6);
                                                        eph.TGD=twosComp2dec(subtrama,197,8) * pow(2,-31);
                                                        eph.IODC=bin2dec3(subtrama,82,2,197,8); ////////////
                                                        eph.toc=bin2dec(subtrama,219,16)* pow(2,4);
                                                        eph.af2=twosComp2dec(subtrama,241,8)*pow(2,-55);
                                                        eph.af1=twosComp2dec(subtrama,249,16)*pow(2,-43);
                                                        eph.af0=twosComp2dec(subtrama,271,22)*pow(2,-31);
                                        break;
```

```
                                        case 2:
                                                        eph.IODE=bin2dec(subtrama,60,8);
                                                        eph.C_rs=twosComp2dec(subtrama,68,16)*pow(2,-5);

            eph.delta_n=twosComp2dec(subtrama,90,16)*gpsPi*pow(2,-43);
                                                        eph.M_0=twosComp2dec3(subtrama,106,8,120,24)* gpsPi
* pow(2,-31) ;   /////////
                                                        eph.C_uc=twosComp2dec(subtrama,150,16)*pow(2,-29);
                                                        eph.e=bin2dec3(subtrama,166,8,180,24)*pow(2,-33);
/////////
                                                        eph.C_us=twosComp2dec(subtrama,210,16)*pow(2,-29);
                                                        eph.sqrt_A=bin2dec3(subtrama,226,8,240,24)*pow(2,-19);
//////
                                                        eph.toe=bin2dec(subtrama,270,16)*pow(2,4);
                                        break;

                                        case 3:
                                                        eph.C_ic=twosComp2dec(subtrama,60,16)*pow(2,-29);

            eph.Omega=twosComp2dec3(subtrama,76,8,90,24)*gpsPi*pow(2,-31);       //////
                                                        eph.C_is=twosComp2dec(subtrama,120,16)*pow(2,-29);

            eph.i0=twosComp2dec3(subtrama,136,8,150,24)*gpsPi*pow(2,-31);     //////
                                                        eph.C_rc=twosComp2dec(subtrama,180,16)*pow(2,-5);

            eph.w=twosComp2dec3(subtrama,196,8,210,24)*gpsPi*pow(2,-31);    ///////

            eph.omegadot=twosComp2dec(subtrama,240,24)*gpsPi*pow(2,-43);
                                                        eph.IODE=bin2dec(subtrama,270,8);
                                                        eph.IDOT=twosComp2dec(subtrama,278,14)*gpsPi*pow(2,-
43);
                                        break;
                                }
             }

    eph.TOW = bin2dec(subtrama,30,17)*6 - 30;    ///  (31:47)
    free(subsub);
    free(subtrama);
    return eph;
 }

 /*********************  C O N V E R T I D O R E S  *************************/

  int *checkphase(int *sub,int nbit0)
 {
  int *sub2,i=0,j;

  sub2=(int *)malloc(30* sizeof(int));


  if(nbit0==1)
  {
     for(i=0;i<30;i++)
      if(i<24)
         if(sub[i]==1)
            sub2[i]=0;
         else
           sub2[i]=1;
      else
         sub2[i]=sub[i];

     return sub2;
     free(sub2);
  }
  else
     return sub;

  free(sub2);

 }

 double bin2dec(int *bin,int inicio,int len)
{
int k,m,n;
double sum=0,b;
int  *bin2;
```

```
bin2=(int *)malloc(len*sizeof(int));

for(k=0;k<len;k++)
bin2[k]=bin[inicio+k];

len=len-1;

for(k=0;k<=len;k++)
{
        n=bin2[k]-0;
        for(b=1,m=len;m>k;m--)
        b*=2;
        sum=(sum+n*b);
 //     printf("%e\n",sum)   ;
}
 free(bin2);
 return(sum);
}



double bin2dec3(int *bin,int inicio1,int len1, int inicio2, int len2)
{
 int j, *bin2,len;
 double sum=0;

 bin2= (int *)malloc(40*sizeof(int));

 for(j=0;j<len1;j++)
 bin2[j]=bin[inicio1+j];

 len=len1+len2;
 for(j=len1;j<len;j++)
 bin2[j]=bin[inicio2-len1+j];

for(j=0;j<len;j++)
printf("%d ",bin2[j]);
printf("\n");

 sum=bin2dec(bin2,0,len) ;

 free(bin2);
 return(sum);
}

double twosComp2dec(int *bin,int inicio,int len)
{
 double comp;
 int *bin2,k;

 bin2=(int *)malloc(len*sizeof(int));

 for(k=0;k<len;k++)
 bin2[k]=bin[inicio+k];

 comp=bin2dec(bin2,0,len);

 if(bin2[0]==1)
        comp = comp - pow(2,len);

         free(bin2);
 return(comp);
}

double twosComp2dec3(int *bin,int inicio1,int len1, int inicio2, int len2)
{
int j, *bin2,len;
double comp;

 bin2= (int *)malloc(40*sizeof(int));

 for(j=0;j<len1;j++)
 bin2[j]=bin[inicio1+j];

 len=len1+len2;
```

```
 for(j=len1;j<len;j++)
 bin2[j]=bin[inicio2-len1+j];
 comp= twosComp2dec(bin2,0,len);
 free(bin2);

 return(comp);

}
#endif

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
                                        /********* ALGORITMOS DE CÁLCULO DE POSICIÒN *********/
#ifndef     _posicion_H_
#define     _posicion_H_

#define c  299792458

typedef struct{                                                 //Estructura de posicion de satelite
            float x;
            float y;
            float z;
            float ClkCorr;
}Sposicion;

typedef struct{
float phi;
float lambda;
float h;
} coordenadas;

Sposicion posicion(efemerides eph,  double TOW);
float pseudorangos(float ClkCorr,seguimient *trackResults,int *firstSubFrame,int numsat);

coordenadas to2geod(float pos[])
{
            float a=6378137.0,b=6356752.3142,f,rtd;
            float X,Y,Z,ex2,ep,p,q,f2,e,d1,d2;
            float phi_c,phi_nxt,R_n,den,L,g;
            int k,itera=7;
            static coordenadas coord;

            X=pos[0];
            Y=pos[1];
            Z=pos[2];
            rtd=57.2958;
            f=0.0034;

            ex2=(2*f) - (f*f);
            ep=ex2/(1-ex2);
            b=a*(1-f);
            p=sqrt((X*X) + (Y*Y));
            q=atan2((Z*a),(p*b));

            f2=(a-b)/2;     //Anchura del elipsoide
            e=sqrt(ex2);    //excentricidad

            coord.lambda=atan2(Y,X)*rtd;

            d1=Z + ep*b*pow(sin(q),3);
            d2=p - ex2*a*pow(cos(q),3);
            phi_c=atan2(d1,d2);
            phi_nxt=phi_c;
            for(k=0;k<itera;k++)
            {
                        den=1 - (e*e)*pow(sin(phi_nxt),2);
                        R_n=a/sqrt(den);
                        d1= Z + ex2*R_n*sin(phi_nxt);
                        d2=sqrt((X*X) + (Y*Y));
                        phi_nxt=atan2(d1,d2);

            }

            coord.phi=phi_nxt*rtd;

            L=Z + pow(e,2)*R_n*sin(coord.phi);
```

```
		g=cos(coord.lambda)*cos(coord.phi);
		coord.h=(X/g) -R_n;

		coord.h=((p/cos(phi_nxt))-R_n)*0.01;

		coord.phi=coord.phi-0.50768;
		coord.lambda=coord.lambda -1.38196;

		return(coord);

}

Sposicion division(float m[][4], float p[])
{
 int j,k,l,h;
 int COL=4,FIL=4;
 float fact,div,m2[4],s2[4],s[4][1];
 static Sposicion pos;

 s[0][0]=p[0];
 s[1][0]=p[1];
 s[2][0]=p[2];
 s[3][0]=p[3];

 for(j=0;j<FIL;j++)
 {
		div=m[j][j];
		 s2[j]=s[j][0]/div;
		 s[j][0]=s[j][0]/div;
		for(k=0;k<COL;k++)
		{
		 m2[k]=m[j][k]/div;
		 m[j][k]=m[j][k]/div;
		}

		for(l=0;l<FIL;l++)
		{
			fact=-m[l][j];
			for(k=0;k<COL;k++)
				m[l][k]=(fact*m[j][k]) + m[l][k];

			s[l][0]=(fact*s[j][0]) +s[l][0];

			if(l==j)
			{
			  for(h=0;h<COL;h++)
					m[l][h]=m2[h];

			s[l][0]=s2[j];
			}
		 }
		}
		pos.x=s[0][0];
		pos.y=s[1][0];
		pos.z=s[2][0];
		pos.ClkCorr=s[3][0];
		return(pos);
 }

Sposicion satcorr(Sposicion XS[], float traveltime,int j)
 {
		float tau,Omegae_dot = 7.292115147e-5;		//  rad/sec
		static Sposicion sat_R[8];

		tau=Omegae_dot*traveltime;

		sat_R[j].x=cos(tau)*XS[j].x + sin(tau)*XS[j].y + 0*XS[j].z;
		sat_R[j].y=-sin(tau)*XS[j].x + cos(tau)*XS[j].y + 0*XS[j].z;
		sat_R[j].z=0*XS[j].x + 0*XS[j].y + 1*XS[j].z;

		return(sat_R[j]);
 }

Sposicion minimos(Sposicion XS[], float obs[], int numsat)
 {
		int i,j,k,l,s,itera=5;
```

```
	float traveltime,rho2;
	float A[8][4],AT[4][8],AB[4],AA[4][4],omc[10],norm;
	static Sposicion sat_R[8],pos;
	static Sposicion pos2;

	pos.x=0.0;
	pos.y=0.0;
	pos.z=0.0;
	pos.ClkCorr=0;
	for(i=0;i<itera;i++){
		for(s=0;s<numsat;s++)
		{
			rho2= (XS[s].x - pos.x)*(XS[s].x - pos.x) + (XS[s].y - pos.y)*(XS[s].y - pos.y) + (XS[s].z -
pos.z)*(XS[s].z - pos.z);
			traveltime=sqrt(rho2)/c;

			sat_R[s]=satcorr(XS,traveltime, s);

			A[s][0]= -(sat_R[s].x- pos.x)/obs[s];
			//printf("%e\t",A[s][0]);
			A[s][1]= -(sat_R[s].y- pos.y)/obs[s];
			//printf("%e\t",A[s][1]);
			A[s][2]= -(sat_R[s].z- pos.z)/obs[s];
			//printf("%e\t",A[s][2]);
			A[s][3]= 1;
			//printf("%e\t",A[s][3]);

			norm=(sat_R[s].x - pos.x)*(sat_R[s].x - pos.x) + (sat_R[s].y - pos.y)*(sat_R[s].y - pos.y) +
(sat_R[s].z - pos.z)*(sat_R[s].z - pos.z);
			// pow((sat_R[s].x - pos.x),2) + pow((sat_R[s].y - pos.y),2) + pow((sat_R[s].z - pos.z),2);
			omc[s]=  obs[s] - pos.ClkCorr - sqrt(norm);
		  //	printf("omc%e\n",omc[s]);

		}
		for(j=0;j<4;j++)
			for(k=0;k<4;k++)
				AA[j][k]=0;		///Vuelve a inicializar la matriz

      for(j=0;j<4;j++)
				AB[j]=0;

		for(j=0;j<4;j++)
			for(k=0;k<numsat;k++)		////La traspuesta
				AT[j][k]=A[k][j];

		 for(j=0;j<4;j++)
			for(k=0;k<4;k++)
				for(l=0;l<numsat;l++)		/////Multiplicacion A'A
					AA[j][k]+=AT[j][l]*A[l][k];

		  for(j=0;j<4;j++)
			for(k=0;k<1;k++)
				for(l=0;l<numsat;l++)
					AB[j]+=AT[j][l]*omc[l];

		pos2=division(AA,AB);

		pos.x=pos.x+pos2.x;
		pos.y=pos.y+pos2.y;
		pos.z=pos.z+pos2.z;
		pos.ClkCorr=pos.ClkCorr+pos2.ClkCorr;

		xil_printf("..........................................\n");
  }
  return(pos);
 }

float pseudorangos(float ClkCorr,seguimient *trackResults,int *firstSubFrame,int numsat)
{
 long temp[15];
 int k;
 double tiempo[15],minimo,muestras,pseudorangos[15];
 double  C=  299792458;
 float obs;

 muestras= samplingFreq/(codeFreqBasis/codeLength) ;
```

```
 for(k=0;k<numberOfChannels;k++)
              tiempo[k]=trackResults[k].absoluteSample[firstSubFrame[k]-1]/muestras;

 minimo=tiempo[1];
 for(k=0;k< numberOfChannels+1;k++)
              if(k<numberOfChannels)
                            minimo=min(minimo, tiempo[k]);

 minimo=floor(minimo);

 for(k=0;k<numberOfChannels;k++)
   pseudorangos[k]=(tiempo[k] - minimo + 68.802)*(C / 1000);

 obs=pseudorangos[numsat] + (ClkCorr * C);

 return(obs);
}

Sposicion posicion(efemerides eph,  float TOW)          ///introducir el arreglo de estructuras
{                                                                                    //// Calculo de posicion de cada satelite ///////
              float Omegae_dot     = 7.2921151467e-5;  // E[rad/s]
              float GM             = 3.986005e14;  //    [m^3/s^2]
              float F              = -4.442807633e-10; // [sec/(meter)^(1/2)]
              long MSEMANA                    = 302400;     // segundos
              int k;
   float dt;
              float time,a,tk,n;
              float n0,M,E,dE,dtr,E_1,nu,phi;
              float u,r,i,Omega;//,ClkCorr;
              float gpsPi                     = 3.1415926535898;  //Pi usado en GPS (WGS-84)
              static Sposicion XS;
              float correloj;

              dt= TOW-(eph.toc);

              if (dt>MSEMANA)
                            dt=dt-2*MSEMANA;
              else
              if (dt< -MSEMANA)
              dt=dt+2*MSEMANA;

              correloj=(eph.af2*dt + eph.af1)*dt + eph.af0 - eph.TGD;

              time=TOW-correloj;

                                          a=eph.sqrt_A*eph.sqrt_A;

                            tk=time-eph.toe;

                            if (dt>MSEMANA)
                                          tk=tk-2*MSEMANA;
                            else
                            if (dt< -MSEMANA)
                                          tk=tk+2*MSEMANA;

                              n0=sqrt(GM/(pow(a,3)));
                              n=n0 + eph.delta_n;


                            //Anomalia media
                              M = eph.M_0 + n*tk;
                M = fmod((M + 2*gpsPi),(2*gpsPi));
                            //Calcular anomalia excentrica con leyes kepler
                              E=M;
                              for(k=0;k<10;k++)
                              {
                                                        E_1=E;
                                                        E= M+eph.e*sin(E);
                                                        dE= fmod((E-E_1),(2*gpsPi));

                                                        if (dE<0.000000000012)
                                                        goto next;
                              }
                              next:
```

```
                    E= fmod((E + 2*gpsPi),(2*gpsPi));
                    dtr = F* eph.e  * eph.sqrt_A * sin(E);
                  //Anomalia verdadera
                    nu= atan2(sqrt(1 - pow(eph.e,2)) * sin(E) , (cos(E)-eph.e));
                  //Argumento de latitud
                              phi=nu + eph.w;
                              phi=fmod((phi),(2*gpsPi));
                  // Corregir argumento de la latitud
                               u=phi + eph.C_uc * cos(2*phi) + eph.C_us  * sin(2*phi);
                  //Corregir radio
                              r=a * (1-eph.e*cos(E)) + eph.C_rc * cos(2*phi) + eph.C_rs * sin(2*phi);
                  //Corregir inclinacion
                              i= eph.i0 + eph.IDOT * tk + eph.C_ic * cos(2*phi) + eph.C_is * sin(2*phi);
                  //Correccion de longitud del nodo ascendente
                              Omega = eph.Omega + (eph.omegadot - Omegae_dot)*tk - Omegae_dot * eph.toe;
                              Omega = fmod((Omega + 2*gpsPi),(2*gpsPi));
                   //Coordenadas de satelites.
                              XS.x = cos(u)*r * cos(Omega) - sin(u)*r * cos(i)*sin(Omega);
                              XS.y = cos(u)*r * sin(Omega) + sin(u)*r * cos(i)*cos(Omega);
                              XS.z = sin(u)*r * sin(i);
                   //correción del reloj de satelite
          XS.ClkCorr = (eph.af2 * dt + eph.af1) * dt + eph.af0 - eph.TGD + dtr;
                              return(XS);
}
#endif
```

Bibliografía

[1] GEOGPS Información. *http://www.geogps.cl/gps003.htm*.

[2] Preámbulo de GPS. *http://www.nps-gov/gis/gps/history.html*.

[3] Global Positionating System. *http://en.wikipedia.org/GlobalPositionatingsystem*.

[4] James J. Parkinson, Spilker Jr. *The Global Positioning System : Theory and Application Volume I*. AIAA, 1996.

[5] Iván Jesús Sánchez Cuapio. *Evaluación e Implementación de Algoritmos de Adquisición de una señal GPS, en un Dispositivo Reconfigurable FPGA*. Instituto Politécnico Nacional, 2009.

[6] Paolo Mulassano Gianmarco Girau, Andrea Tomatis. *Efficient Software Defined Radio Implementations of GNSS Receivers*, 2009.

[7] Frantisek Vejrazka Pavel Kovar. *Software Radio and its Applications in GNSS*, 2009.

[8] Amadeo José Argüelles Cruz Juan Manuel Castro Arvizu, Miguel Sánchez Meraz. *GPS Receiver based on a SDR Architecture using FPGA Devices*, 2010.

[9] Zhao Huichang Wang Lijun, Yang Xiaoniu. *Data Acquisition and PRocessing Technique for Software GPS Receivers*, 2009.

[10] Oleksiy V. Korniyenko Mohammad S. Sharawi. *Software Definied Radios: A Software GPS Receiver Example*, 2007.

[11] Oleksiy V. Korniyenko Mohammad S. Sharawi. *Software Radio and its Applications in GNSS*, 2007.

[12] Shufang Zhang Ershen Wang. *Implementation of an Embedded GPS Receiver Based on FPGA and Microblaze*, 2008.

[13] Paolo Mulassano Massimiliano Spelat Fabio Dovis, Gianmarco Girau. *A Flexible FPGA-DSP Board for GNSS Receivers Design*, 2008.

[14] César Daniel Pérez Montoya. *Implementación de los Algoritmos de Seguimiento de la señal GPS sobre Dispositivos Lógico Programables (FPGA).* Instituto Politécnico Nacional, 2009.

[15] Nicolaj Bertelsen Kai Borre, Dennis M. Akos. *A Software-Defined GPS and Galileo Receiver.* Birkhauser, 2007.

[16] Mohammad S. Sharawi. *Use of Patch Antennas en Modern Wireless Technology*, 2006.

[17] A.J. Van Dierendock Michael S. Braasch. *GPS Receiver Architectures and Measurements*, 1999.

[18] NAVSTAR Global Positionating System. *ICD-GPS-200C GPS Interface Control Document*, 2000.

[19] Kai Strang, Gilbert Moore. *Linear algebra,Geodesy and GPS*, 1997.

[20] M.Barth J.Farrell. *Conversion of Geodetic coordinates to the Local Tangent Plane*, 2007.

[21] James R. Clynch. *Geodetic Coordinate Conversions*, 2006.

[22] FX2 Programmer. *http://volodya-project.sourceforge.net/fx2_programmer.php.*

[23] T. Davis. USB Development Mistakes You Dont Have To Make Them All Yourself. Cypress Semiconductor 2006. S. Kolokowsky. *http://www.cypress.com.*

[24] Guna Seetharaman Anant Utgikar. *FPGA Implementable Architecture for Geometric Global Positioning*, 2009.

[25] SparkFun Electronics. *SiGe GN3S V2.* http://www.sparkfun.com/products/8238.

[26] Marcus Junered. *Enabling Hardware Technology for GNSS Software Radio Research.* Lule University of Technology, 2007.

[27] GNU Radio. *http://www.gnuradio.org.*

[28] Small Device C Compiler (SDCC). *http://www.sdcc.sourceforge.net.*

[29] Xilinx Company. *Página de Xilinx.* http://www.xilinx.com.

[30] ML505/506/507 Standard IP Design Adding a USB Controller. *http* : $//www.xilinx.com/products/boards/ml505/docs/ml505_{std_ip_usb_additiopn.pdf}$.

[31] Sergey Chernenko. *Fast Fourier Transform Implementacion using Cooley and Turkey.* http://www.librow.com/articles/article-10, 2007.

[32] S. Esterhuizen. *The Design, Construction, and Testing of a Modular GPS Bistatic Radar Software Receiver for Small Platforms*. University of Colorado, 2004.

[33] Dunshan Yu Lingjuan Wu, Yingying Cui. *Signal Acquisition and Tracking for Software GPS Receivers*, 2009.

[34] Yingying Cui Lingjuan Wu. *Signal Acquisition and Tracking for Software GPS Receivers*, 2006.

[35] Juan Antonio Martínez Rosique. *El Sistema de Posicionamiento Global (GPS) : Principios Básicos de Funcionamiento*. Universidad Politécnica de Valencia, España, 1995.

[36] Cheon Sig Shin Choi Seung Hyun, Kim Jae Hyum. *Acquisition and Tracking Schemes for a GPS receiver*, 2007.

[37] The History of Navegation. *http://www.boatsafe.com/kids/navigation.htm*. Nautical Know How, Inc., 2004.