Universitat Oberta de Catalunya

uoc.edu

Estudio de la viabilidad de sensores GNSS en cloud para aplicaciones IoT

Fernando Rueda Teruel
Master Universitario en Ingeniería de Telecomunicación UOC-URL
Tecnologías de Radiocomunicación

Consultor/a: José Antonio López Salcedo
Profesor/a responsable de la asignatura: Germán Cobo Rodríguez

18 de Junio de 2018

Esta obra está sujeta a una licencia de
Reconocimiento-NoComercial-CompartirIgual
3.0 España de Creative Commons

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Estudio de la viabilidad de sensores GNSS en aplicaciones IoT*
Nombre del autor:	*Fernando Rueda Teruel*
Nombre del consultor/a:	*José Antonio López Salcedo*
Nombre del PRA:	*Germán Cobo Rodríguez*
Fecha de entrega (mm/aaaa):	06/2018
Titulación:	*Máster Universitario en Ingeniería de Telecomunicación UOC-URL*
Área del Trabajo Final:	*Tecnologías de Radiocomunicación*
Idioma del trabajo:	*Castellano*
Palabras clave	*GNSS, IoT, Cloud, GPS, Galileo*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

El TFM tiene por finalidad determinar si es viable el uso de receptores GNSS (Global Navigation Satellite System) en los que el procesamiento de la señal se llevará haciendo uso de la nube, es decir, el receptor GNSS actúa como elemento de recepción de las señales y mensajes de navegación procedentes de satélites tanto GPS como Galileo y dichas señales se envían a la nube para su procesamiento.

Se ha decidido dividir el TFM en dos fases, de manera que la primera fase sirva como base sobre la que se sustenta la segunda.

En una primera fase se procede al estudio de las prestaciones que debe brindar un receptor GNSS para adquirir las señales y mensajes de la constelación GPS en las bandas L1 y L5, así como las señales y mensajes de la constelación Galileo en las bandas E1, E5 y E6. Esta fase incluye además el desarrollo de una interfaz gráfica de usuario (GUI) que permite modelar el escenario y parámetros de recepción de la señal GNSS y permite obtener mediante MATLAB información sobre el rendimiento del receptor GNSS.

En una segunda fase se estudia la viabilidad del uso de sensores GNSS en aplicaciones IoT (Internet of Things), determinando los efectos limitantes de dicha solución (consumos, capacidades, etc) así como la viabilidad de externalizar el cálculo de la posición (recordemos que las señales y mensajes recibidos de los satélites son enviados a la nube y por tanto el procesamiento de dichas señales puede ser realizado por terceros).

Abstract (in English, 250 words or less):

The purpose of the TFM is to determine if it is feasible to use GNSS (Global Navigation Satellite System) receivers in which the signal processing will take place using the cloud, that is, the GNSS receiver acts as a signal reception element and navigation messages from both GPS and Galileo satellites and the signals are sent to the cloud for processing.
It has been decided to divide the TFM into two phases, so that the first phase serves as the basis on which the second phase is based.
In a first phase we proceed to the study of the benefits that a GNSS receiver must provide to acquire the signals and messages of the GPS constellation in the L1 and L5 bands, as well as the signals and messages of the Galileo constellation in the E1, E5 bands and E6. This phase also includes the development of a graphical user interface (GUI) that allows modeling the scenario and reception parameters of the GNSS signal and allows obtaining information on the performance of the GNSS receiver through MATLAB.
In a second phase, the feasibility of using GNSS sensors in IoT (Internet of Things) applications is studied, determining the limiting effects of this solution (consumption, capacities, etc.) as well as the viability of outsourcing the position calculation (remember that the signals and messages received from the satellites are sent to the cloud and therefore, the processing of the signals can be done by third parties)

Índice

Índice iii
Lista de figuras v
1. Introducción 1
1.1 Contexto y justificación del Trabajo 1
1.2 Objetivos del Trabajo 2
1.3 Enfoque y método seguido 3
1.4 Planificación del Trabajo 4
1.5 Breve sumario de productos obtenidos 4
1.6 Breve descripción de los otros capítulos de la memoria 5
2. Estado del Arte 7
2.1 HS-GNSS y A-GPS [4] [5]: 7
2.2. DINGPOS [6] 8
2.3 GCCS (Geodetic Cloud Computing Service) 9
2.4 XYBRID Cloud [8] 10
2.5 Ground Based Scintillation Climatology (GBSC) [9] 11
2.6 Soluciones comerciales en IoT tracking 13
3. Principio de funcionamiento GNSS 14
3.1. Estructura de la señal GPS 14
3.2. Estructura del receptor GPS 18
4. Parámetros relevantes en la detección señal GNSS 29
4.1 Potencia de transmisión y Pathloss 29
4.2 Ganancia de la antena 29
4.3 C/No, temperatura de ruido, fórmula de Friis 31
4.4 Sensibilidad del receptor 33
4.5 Detección de la señal en ruido 33
4.5.1 Criterio Neyman Pearson 34
4.5.2 Adquisición de la señal como un problema de detección [14][15] 35
5. Probabilidad detección de la señal GNSS 38
5.1 GUI MATLAB. Cálculo de Probabilidades de Detección 38
5.1.1. Probabilidad de Detección en función del Número de Integraciones no coherentes (Ni): 39
5.1.2. Probabilidad de Detección en función del Tiempo de Integración coherente (Tcoh): 41
5.1.3. Probabilidad de Detección en función de la relación portadora a densidad espectral de ruido de la señal (C/No): 42
5.1.4. Precisión en los pseudorangos 43
6. Internet of Things (IoT) y GNSS 45
6.1. Unmanned Aerial Vehicle, UAV (Drone Delivery) 46
6.2. Seguimiento Activo (Asset Tracking): 48
6.3. Cuidado de la Salud (healtcare), GNSS e IoT [33] 50
6.4. Agricultura de Precisión, AP 51
6.5. Ubicuidad 53
6.6. Escenarios de Trabajo 54
6.6.1 Escenarios Outdoor. 54
6.6.2 Escenarios Outdoor entorno urbano 56

7. Procesamiento de la señal GNSS en la nube 59
7.1. Motivación 59
7.2. Procesamiento en la nube 59
Infraestructura como servicio (IaaS) 60
Plataforma como servicio (PaaS) 60
Software como servicio (SaaS) 60
Servicios de Computación AWS: Elastic Compute Cloud (EC2) 60
Servicios de Almacenamiento EC2 62
Amazon Elastic Block Storage (EBS) 62
Almacén de instancias Amazon EC2 63
Sistema de archivos de Amazon Elastic File System (EFS) 63
Amazon Simple Storage Service (S3) 63
AWS Simple Queue Service (SQS) 64
7.3. Arquitectura de un receptor GNSS basado en la nube 64
8. Soluciones comerciales para cloud GNSS 67
8.1 ADALM-Pluto [64] 67
8.2 Airspy Mini [66] 68
8.3 Blade RF Nuand [68] 69
8.4 FreeSRP [69] 70
8.5 HackRF One [71] 70
8.6 LimeSDR [72] 71
8.7 RTL-SDR [73] 71
8.8 SDRplay (RSP1A) [74] 72
8.9 Sideqik [75] 72
8.10 SiGe GN3S Sampler v3 [76] 73
8.11 Ettus USRP N210 [77] 73
8.12 Ettus USRP X310 [78] 74
8.13 XTRX [79] 74
9. Consumos de Energía 75
10. Conclusiones 80
11. Glosario 83
12. Bibliografía 85
13. Anexos 90
Menú Principal GUI 90
Pd en función de Ni 92
Pd en función de Tcoh 98
Pd en función de C/No 103
Precisión en los pseudorangos 107

Lista de figuras

Figura 1. Planificación TFM
Figura 2. Diagrama de bloques sistema DINGPOS
Figura 3. Diagrama de bloques sistema XYBRID
Figura 4. Diagrama de bloques de GBSC
Figura 5. Tiempo viaje señal GNSS
Figura 6. Etapas uso GNSS
Figura 7. Generación señal GPS (L1 y L2)
Figura 8. Proceso de modulación señal GPS (1)
Figura 9. Proceso de modulación señal GPS (2)
Figura 10. Espacio búsqueda de Adquisición GNSS
Figura 11. Diagrama bloques receptor GPS
Figura 12. Diagrama bloques señal salida satélite
Figura 13. Etapa FI en el receptor
Figura 14. Ramas en fase y cuadratura receptor GPS
Figura 15. Correlación receptor GPS
Figura 16. Componente sinusoidal del código
Figura 17. Pico de correlación señal Prompt
Figura 18. Diagrama bloques. Correlación en Integración
Figura 19. Correlación en Integración con presencia de ruido
Figura 20. Integración coherente ideal (3msg)
Figura 21. Integración coherente real (3msg)
Figura 22. Diagrama de bloques Integración no coherente.
Figura 23. Energías en ramas en fase y cuadratura
Figura 24. Post-Correlación I y Q.
Figura 25. Densidad potencia espacio libre
Figura 26. Área de Apuntado
Figura 27. Ganancia Transmisión Antena (1)
Figura 28. Ganancia Transmisión Antena (2)
Figura 29. Diagrama Bloques canal del receptor
Figura 30. Parte Analógica del FrontEnd
Figura 31. Espacio de Observaciones
Figura 32. Conjunto de celdas. Espacio de búsqueda de adquisición
Figura 33. Menu Principal.GUI
Figura 34. Representación de la Pd en función de Ni
Figura 35. Representación de la Pd en función de Ni para distintos Tcoh
Figura 36. Representación de la Pd en función del Tcoh
Figura 37. Representación de la Pd en función de Tcoh para distintos Ni
Figura 38. Representación de la Pd en función de C/No
Figura 39. Ejemplo Cálculo de Precisión en el Pseudorango
Figura 40. Precisión en la localización por tecnología
Figura 41. Payload del UAV VS Km recorridos
Figura 42. Consumo de Energía UAV VS Km recorridos
Figura 43. Características Tipo. Asset Tracking Module
Figura 44. Duración Batería/Reportes diarios
Figura 45. Módulo GPS SIRF V
Figura 46. Clasificación AP según propósito

Figura 47. Características receptor GNSS Harvesting
Figura 48. Escenarios trabajo indoor típicos
Figura 49. AP Precisión (≈2,5m) GPS L1, GPS L5.
Figura 50. AP Precisión (≈2,5m) Galileo E1, E5 y E6.
Figura 51. Valor de C/No en entorno urbano. GPS Test
Figura 52. UAV Precision (≈2m) GPS L1, GPS L5 Tcoh=40ms
Figura 53. UAV Precisión (≈2m) Galileo E1, E5 y E6 Tcoh=40ms
Figura 54. Regiones y Zonas AWS
Figura 55. Ciclo de vida de una AMI
Figura 56. Tipos de Almacenamiento AWS
Figura 57. AWS: Simple Queue Service
Figura 58. Arquitectura de un receptor cloud GNSS
Figura 59. Receptores GNSS-SDR
Figura 60. Transceptor ADALM-Pluto
Figura 61. Diagrama de bloques ADALM-Pluto
Figura 62. AirSpy Mini
Figura 63. Blade RF
Figura 64. FreeSRP
Figura 65. HackRF One
Figura 66. LimeSDR
Figura 67. RTL-SDR
Figura 68. RSP1A
Figura 69. Sideqik
Figura 70. SiGe GN3S Sampler V3
Figura 71. Ettus USRP N210
Figura 71. Ettus USRP X310
Figura 72. Ettus USRP X310 [78]
Figura 73. XTRX [79]
Figura 74. Chunk Size GPS/L1
Figura 75. Chunk Size Galileo/E5
Figura 76. Consumo IoT device USB 3.0 [80]
Figura 77. Consumo IoT con trafico IP [80]
Figura 78. Consumos dispositivos opensky. GPS/L1
Figura 79. Consumos dispositivos urbano.GPS/L1
Figura 80. Consumos dispositivos opensky. Galileo/E5
Figura 81. Consumos dispositivos urbano. Galileo/E5

1. Introducción

1.1 Contexto y justificación del Trabajo

Los sistemas de radionavegación por satélite empezaron sus primeros pasos en firme a finales de la década de los 70, con el programa DNSS del Departamento de Defensa de los Estados Unidos y que dio como resultado la aparición de NAVSTAR GPS.
El primer satélite de la constelación GPS se lanzó en 1978 y formaba parte de un grupo de satélites que tenían por objetivo tareas de experimentación de la tecnología de posicionamiento por Satélite.
El despliegue, por tanto, de la constelación de satélites que forman el sistema GPS, se hizo en fases, dónde cada fase tiene asociado un bloque de satélites. Los distintos bloques, por orden cronológico son:

- Bloque I: Formado por 11 satélites lanzados entre los años 1978 y 1985 con tareas de experimentación
- Bloque II: Formado por 9 satélites lanzados entre los años 1989 y 1990 y cuyo objetivo era poner a prueba la experiencia ganada durante los años precedentes, de manera que los satélites mejoraban en fiabilidad y robustez.
- Bloque IIA: Formado por 19 satélites lanzados entre 1990 y 1997 permitían un periodo de operación más largo.
- Bloque IIR: Es el bloque de reposición, formado por 13 satélites (aunque uno falló) lanzados entre 1997 y 2004.
- Bloque IIRM: También llamado modernizado. Se lanzaron 8 satélites entre los años 2005 y 2009, que ofrecían nuevas funcionalidades y ciertos rediseños que mejoraban las prestaciones
- Bloque IIF: Formado por 7 satélites del bloque IIF,el último de los cuales se lanzó en 2016, con los que se ha conseguido mejorar el posicionamiento a un error de tan sólo 42 centímetros.
- Bloque III: A partir de este año 2018 está previsto el inicio de lanzamiento de hasta 36 satélites de este grupo (divididos en subgrupos A, B y C).

Por su parte la Comisión Europea, en vista de que los sistemas de posicionamiento por satélite se habían convertido en un tema estratégico, decidió en la década de los 90 iniciar el desarrollo de su propio sistema de GNSS y en el año 2003 anunció el desarrollo del sistema GNSS Galileo que contará con 30 satélites y estará operativo al cien por cien en el 2020, aunque desde el 2016 se encuentra operativo con 18 satélites.

Hay que tener en cuenta que el mercado de los sistemas GNSS mueve unos 175.000 millones de euros al año [1] y por ejemplo, se espera que Galileo tanga un impacto económico en los próximos años de 90.000 millones de euros.

Desde sus inicios hasta la fecha actual, el uso de sistemas de posicionamiento GNSS ha crecido de manera exponencial y tal y como se recoge en el informe de la GSA, sobre el mercado de GNSS de 2017[2]**.** La previsión es pasar de los 5,8 billones de dispositivos GNSS en uso en el mundo en 2017 a los 8 billones en 2020, lo que supone más de un dispositivo por persona en el planeta.

En un reciente informe de la European Global Navigation Satellite Systems Agency (GSA) [3] se pone de manifiesto que el futuro de la tecnología GNSS está determinado por las necesidades de los usuarios, la infraestructura y la tecnología disponible.
Entre las tendencias futuras nos encontramos con que el mercado de IoT va a jugar un papel importante. El posicionamiento está entre los principales requerimientos para el diseño de dispositivos embebidos y más teniendo en cuenta la tendencia hacia la movilidad de los mismos. Como estos dispositivos IoT no están sujetos a un emplazamiento determinado y además requieren tiempos de vida largos, la solución a priori sería minimizar el consumo de dichos dispositivos. Sin embargo, esto representa un problema, ya que típicamente los receptores GNSS tienen un consumo de potencia relativamente alto. Es aquí donde el procesamiento en la nube (SaaS, Software As A Service) puede resolver este problema reduciendo los consumos de potencia a nivel de dispositivo.

1.2 Objetivos del Trabajo

Los objetivos que recoge el trabajo son múltiples:
En una primera etapa se da una visión del funcionamiento general de los sistemas GNSS, en la que nos centramos en las constelaciones GPS y Galileo. Se introduce una visión muy general de las frecuencias de operación y de la estructura de la señal y el mensaje de navegación. Posteriormente damos una pequeña visión sobre la arquitectura de los receptores GNSS, centrándonos en los procesos de acondicionamiento y adquisición de la señal de navegación. De igual forma vemos cómo se pueden determinar las probabilidades de detección (Pd), de falsa alarma (Pfa), de detección fallida (Pdf) en función de distintos parámetros como son el tiempo de integración, la relación portadora a ruido C/No, bandwidth, etc. Estos parámetros, nos definirán la sensibilidad del receptor y la precisión de las medidas que podremos tener con él. Posteriormente se diseña una GUI en MATLAB que permite simular distintos entornos, modificando los parámetros antes expuestos, lo que nos permite ver de una forma gráfica las prestaciones que obtendríamos de nuestro receptor en cuanto a las medidas del pseudorango.

Por último, lo que se pretende es determinar si dado que los receptores GNSS típicamente tienen consumos de energía no despreciables, es posible utilizar aplicaciones de posicionamiento en dispositivos IoT, donde en contraposición a los receptores típicos de GNSS, el consumo de energía es un factor limitante. Se plantea como solución a este problema, utilizar los dispositivos IoT como elementos cuya única función sería captar la señal y enviarla a un servidor remoto donde se realizaría su procesado.
Aunque pudiera parecer un problema de segundo orden, no lo es. Debemos tener en cuenta que para el 2020 se espera tener más de 50 billones de dispositivos interconectados y vivimos pendientes de que se produzca una crisis energética mundial (los expertos coinciden en que dicha crisis se producirá con total seguridad, falta por determinar el cuándo), por lo que sólo aquellos dispositivos con muy bajos consumos de energía van a poder sobrevivir en este escenario.
Se realizará un sondeo del mercado actual de los dispositivos IoT y una visión del estado del arte de dichos dispositivos con capacidad de posicionamiento GNSS.

En resumen, todos los cálculos realizados en la primera etapa, respecto a Pd, Pfa, Pdf etc, nos ayudarán a entender y determinarán las características que debe tener nuestro receptor para alcanzar una determinada precisión en la medición de los pseudorangos. Esto nos permitirá determinar si dentro de los dispositivos IoT, podemos integrar una solución GNSS que cumpla los requerimientos determinados en el apartado anterior, lo que abre la puerta para el uso de estos dispositivos haciendo uso de la nube para el tratamiento de las señales y mensajes de radionavegación, lo que diluye la principal barrera de este enfoque, que no es otra que las limitaciones en cuanto al consumo de energía de los dispositivos IoT

1.3 Enfoque y método seguido

Como se ha señalado anteriormente, los receptores GNSS típicos, tienen un consumo de energía no despreciable, lo que en principio podría parecer incompatible con dispositivos IoT donde, precisamente, los consumos de energía deber ser reducidos. La vía clásica para enfrentarse a este problema ha sido intentar reducir al máximo los consumos de los dispositivos IoT, lo que se está consiguiendo gracias al avance de la técnica, llegándose en la actualidad dispositivos IoT de muy bajo consumo, típicamente del orden de los miliamperios.
Básicamente para ‘atacar’ desde este enfoque de reducir el consumo de los dispositivos IoT se ha hecho desde tres frente:

- Optimizando el consumo de energía del hardware:
- Optimizando la eficiencia del Software.
- Recolectando energía del entorno (solar, cinética, etc.)

La solución que analizamos aquí es ligeramente diferente y creemos que muy interesante, puesto que no sigue un camino que suponga una oposición frontal a las soluciones clásicas, más bien un complemento o una vía alternativa que apoya (o se ve apoyada) por dichas soluciones.

Lo que se pretende en esta solución es eliminar toda aquella parte de procesado de la señal que habitualmente se realiza en el propio receptor GNSS, para que se haga en un entorno externo y no sujeto a las restricciones en cuanto a consumo de energía que tienen los dispositivos IoT. Eliminar esta parte de procesado de la señal de navegación de la parte del dispositivo, veremos que tiene unas implicaciones importantes en la simplificación a nivel hardware y software del mismo y una consecuencia inmediata será la reducción del consumo de energía de nuestro dispositivo. Otra consecuencia inmediata que se obtiene con este planteamiento es el abaratamiento del receptor GNSS, al eliminar gran parte de la complejidad del mismo.

1.4 Planificación del Trabajo

Definición Alcance TFM	**8 días**	**jue 01/03/18**	**lun 12/03/18**
Inicio PEC1	1 día	jue 01/03/18	jue 01/03/18
Definición de tema, objetivos, contexto, etc	2 días	vie 02/03/18	lun 05/03/18
Planificación etapas TFM	5 días	lun 05/03/18	vie 09/03/18
Entrega PEC1	1 día	lun 12/03/18	lun 12/03/18
FASE I	**36 días**	**lun 19/03/18**	**lun 07/05/18**
Parte A	**19 días**	**lun 19/03/18**	**jue 12/04/18**
Cap1. Funcionamiento GNSS	3 días	mar 20/03/18	jue 22/03/18
Cap2.Estructura señal GNSS	3 días	lun 26/03/18	mié 28/03/18
Cap3.Estructura receptor GNSS	8 días	mar 27/03/18	jue 05/04/18
Entrega PEC2	1 día	lun 09/04/18	lun 09/04/18
Cap4. Parametros relevantes (Pd,Pfa, C/N0, interferencias..etc)	3 días	mar 10/04/18	jue 12/04/18
Parte B	**18 días**	**jue 12/04/18**	**lun 07/05/18**
Programación funciones Matlab	8 días	jue 12/04/18	lun 23/04/18
Desarrollo GUI Matlab	7 días	lun 23/04/18	mar 01/05/18
Simulaciones y Depuración	6 días	lun 30/04/18	lun 07/05/18
FASE II	**35 días**	**lun 07/05/18**	**vie 22/06/18**
Parte A	**15 días**	**lun 07/05/18**	**vie 25/05/18**
Requisitos Posicionamiento tipico IoT aplicaciones GNSS	15 días	lun 07/05/18	vie 25/05/18
Escenarios posibles (urbano, cielo abierto, interferencias...etc)	4 días	mar 22/05/18	vie 25/05/18
PARTE B	**21 días**	**vie 25/05/18**	**vie 22/06/18**
Estudio Consumos de Energía, Time of Life,coste	20 días	vie 25/05/18	jue 21/06/18
Preparación Defensa	2 días	jue 21/06/18	vie 22/06/18

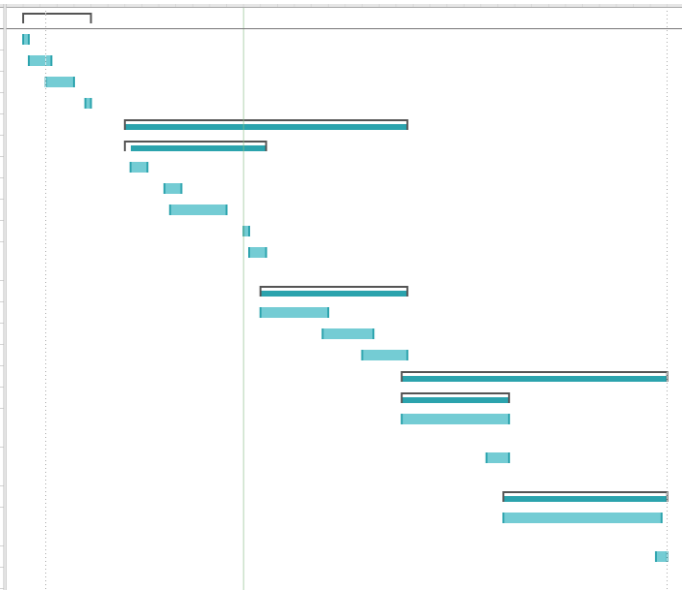

Figura 1. Planificación TFM

1.5 Breve sumario de productos obtenidos

A continuación, se van a describir brevemente los productos que se han obtenido en la elavoración de este Trabajo Fin de Máster:

- Memoria descriptiva donde se da una visión general del receptor GNSS y se analiza la viabilidad de realizar el procesamiento de la señal GNSS en la nube para aplicaciones IoT.
- GUI en Matlab que permite calcular la probabilidad de detección de la señal GPS/L1/L5 y Galileo E1/E5/E6 en función de distintos parámetros

de configuración como son el tiempo de integración coherente (Tcoh), el número de integraciones no coherentes (Ni), la probabilidad de falsa alarma global (PFA) y la relación portadora-densida espectral de ruido (C/No). Está GUI también permite ontener la máxima precisión esperada en los pseudorangos, en función del tipo de señal, del tiempo de integración coherente (Tcoh), del número de integraciones no coherentes (Ni), de la relación portadora-densida espectral de ruido (C/No) de la señal y del ancho de banda del receptor (BW).

- Breve video de presentación del TFM

1.6 Breve descripción de los otros capítulos de la memoria

- Estado del arte: En esta sección y sus respectivas subsecciones se ha dado un repaso muy general a algunas soluciones en las que las señales GNSS son envíadas a la nube para utilizarlas en una determinada aplicación. Aunque no se trata de señales raw GNSS, si parece relevante dar un breve repaso a aplicaciones donde GNSS y cloud funcionan en simbiosis.
- Principio funcionamiento GNSS: Esta sección se ha dividido en dos subsecciones. En la primera se da un vistazo a la estructura de la señal GPS por ser el sistema GNSS más extendido, no se entra en mucho detalle ya que se considera que hay literatura suficiente sobre el tema. En una segunda subsección se analiza la estructura y funcionamiento del receptor GNSS, con el objetivo de aclarar que elementos del receptor son necesarios en este acercamiento en el que la señal se procesa en la nube.
- Parámetros relevantes en la detección de la señal: Se hace un breve repaso a los distintos parámetros que juegan un papel en la detección de la señal como la potencia d etransmisió, el pathloss, la ganancia de la antena, la C/No, sensibilidad del receptor…etc. Además, se realiza un breve análisis de la adquisición de la señal como un problema de detección.
- Probabilidad de detección de la señal GNSS: En este apartado se analiza la GUI elaborada en MATLAB y se muestran ejemplos de las distintas probabilidades de detección de la señal en función de varios parámetros (Tcoh,Ni,C/No,PFA).
- Internet of Things (IoT) y GNSS: En este apartado se da un pequeño repaso no exahustivo sobre aplicaciones IoT que utilizan posicionamiento satélite, así como las precisiones requeridas. Además, se han definido dos escenarios tipo de trabajo, opensky y urbano que se utilizaran en los siguientes capítulos.
- Procesamiento de la señal GNSS: En este apartado se exponen los principales motivos para hacer uso del procesamiento de la señal GNSS en cloud para aplicaciones IoT, también se hace un pequeño análisis de uno de estos servicios en la nube, concretamente Amazon Web Services (AWS)
- Soluciones comerciales para cloud GNSS: En este se analizan de manera breve once dispositivos SDR que podrían utilizarse como

cabezales RF para capturar la señal raw GNSS y enviarla através de sus respectivos interfaces a la nube

- Consumos de Energía: En este apartado se hace un análisis aproximado del consumo de energía que podrían tener estos dispositivos para el envío de los samples raw GNSS a la nube en dos escenarios outdoor posibles, opensky y urbano.
- Conclusiones: En este punto se desglosan las lecciones aprendidas y los distintos puntos de mejora que quedan pendientes de analizar
- Bibliografía: Se muestra un listado de toda la documentación consultada para la realización de este TFM
- Anexos: Se incluye el código en MATLAB de la GUI desarrollada en este TFM

2. Estado del Arte

Antes de adentrarnos en la solución objetivo de este Trabajo Final, queremos dar una visión de conjunto de las soluciones punteras que se están dando en lo relativo al posicionamiento al uso de GNSS, empezando con el posicionamiento con receptores de alta sensibilidad, así como con experiencias que se están llevando a cabo en el uso de la nube y grid para el procesamiento de las señales GPS

2.1 HS-GNSS y A-GPS [4] [5]:

Las señales GPS y Galileo usan un esquema de modulación llamado CDMA (Code División Multiple Access).
La señal es tan débil que un receptor GNSS normal no puede determinar la posición del receptor en áreas populadas por edificios, etc. Para estas situaciones, se requiere un receptor de alta sensibilidad.
El documento de control de interfaces GPS (ARINC,2000) define la potencia mínima con la que un usuario en la superficie terrestre debe poder recibir la señal de navegación. En el caso de la señal C/A, esta potencia es de -160 dBW. Como sabemos la unidad dB indica un ratio logarítmico de potencias R_{DB} donde se comparan las potencias P_1 y P_2:

$$R_{DB} = 10 \cdot \log\left(\frac{P_1}{P_2}\right)$$

La unidad definida más arriba como dBW describe un ratio RDBW respecto a 1 Watt:

$$R_{DBW} = 10 \cdot \log\left(\frac{P}{1W}\right)$$

Este valor en la superficie de la tierra de -160dBW, obliga a transmitir el código C/A en el satélite con una potencia de 14,3 dBW lo que es igual a 27 Watt (W). Con esa potencia el valor de -160 dBW solo puede alcanzarse si no ocurren atenuaciones en la señal. Evidentemente en el mundo real, la señal se ve atenuada por la atmósfera, edificios, árboles, etc.
Este valor de atenuación puede alcanzar valores de 20 dB en el interior de edificios o 25 dB en garajes subterráneos. Este valor de atenuación hace que los receptores GPS “normales” no puedan adquirir la señal.

En lo que refiere a receptores de bajo coste y frecuencia única, los fabricantes ofrecen dispositivos capaces de adquirir la señal por debajo de los -180 dBW, este tipo de receptores son conocidos receptores de alta sensibilidad (HS, High Sensitive Receivers)

Estos receptores de alta sensibilidad consiguen una mejor relación señal a ruido (). Dado que la potencia de la señal no se puede cambiar, la potencia de ruido que es función del ancho de banda y de la densidad de ruido tiene que ser reducida para mejorar la SNR. Para llevar a cabo esta tarea se requiere un mayor tiempo de integración de la señal para reducir el efecto del ruido en el proceso de correlación de la señal, lo cual no es posible debido al efecto

Doppler, así como a la longitud de los bits de datos del mensaje de navegación. Estos inconvenientes pueden ser superados usando adicionalmente integración no coherente. De cualquier manera, la adquisición de señales con tan bajos dBw, sólo puede ser llevada a cabo en un proceso de horas. Por esta razón los receptores HS usan multitud de correladores reduciendo el espacio de búsqueda de cada correlador. Valores típicos rondan los 200.000 correladores de un receptor HS frente a los 36 correladores de un receptor GPS 'normal'. Algunas veces y aun usando tal cantidad de correladores el tiempo de adquisición puede ser muy alto, el problema puede resolverse usando información externa para reducir la ventana de búsqueda de cada correlador, son los llamados Assited GPS (A-GPS).

2.2. DINGPOS [6]

Como hemos visto el posicionamiento GNSS es una solución óptima cuando nos encontramos en el entorno adecuado, es decir este sistema de posicionamiento se pensó para condiciones de cielo abierto, con línea de visión directa entre los satélites y los receptores. Los problemas surgen cuando nos encontramos en ambientes urbanos, interior de edificios, bosques, cañones etc. Donde se producen efectos de multicamnino, atenuaciones, interferencias near-far etc. Para atacar este problema han ido surgiendo diferentes enfoques, como el proyecto DINGPOS cuyo objetivo es proporcionar un sistema de navegación robusto y fiable en ambientes exteriores e interiores. DINGPOS propone una plataforma de navegación basada en la unión de tres tecnologías distinta: GNSS, Sistemas de Navegación Pedestre (PNS, Pedestrian Navegation System) basados en sensores inerciales MEMS y WiFi.
En lo referente a la señal GNSS, DINGPOS es multi-satélite y multi-señal y puede trabajar con las señales en L1 y L5 en GPS, así como con E1 y E5a de Galileo. En el módulo de adquisición GNSS se consigue adquirir señales con una relación C/N0 por debajo de los 5dBHz haciendo uso de las ayudas que un receptor exterior proporciona el receptor interior (se entiende exterior como open-sky e interior como dentro de un recinto, edificio, etc.), Este módulo GNSS se ha desarrollado haciendo uso del lenguaje CUDA (Compute Unified Device Architecture) sacando ventaja de la computación en paralelo. Además, hace uso de GPUs muticore lo que le permite también correr miles de hilos de manera simultánea, lo cual encaja perfectamente con los receptores GNSS que por una parte deben adquirir de manera simultánea la señal de varios satélites y además se debe hacer un uso extenso de procesos FFT para la búsqueda del tiempo y la frecuencia de la señal.
Por otra parte, el PNS desarrollado para el sistema DINGPOS, es un sistema de navegación autónomo que proporciona información sobre la dirección y velocidad del usuario haciendo uso de tecnología inercial basada en sensores Xsens MEMS.

El módulo WiFi por su parte, es capaz de basar su posición absoluta en función de la potencia recibida de la señal.

Se muestra a continuación el diagrama funcional de la plataforma DINGPOS [6]:

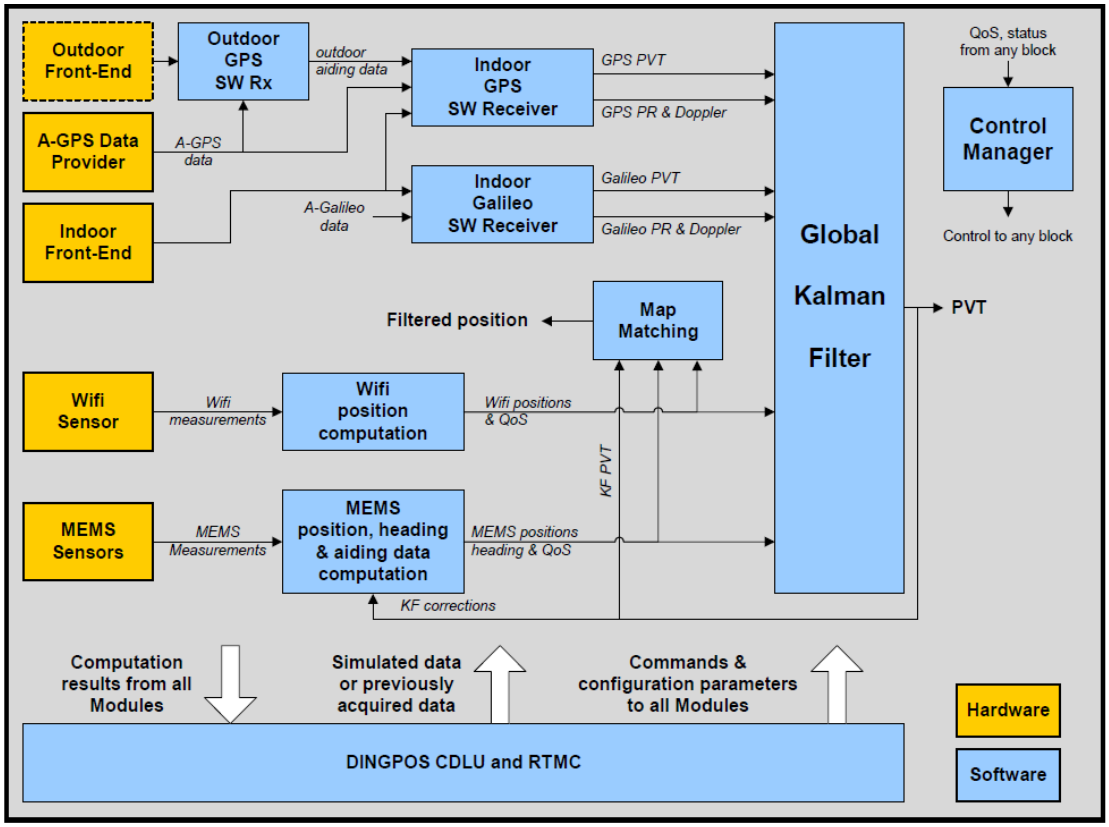

Figura 2. Diagrama de bloques sistema DINGPOS

La hibridación de las señales se produce en el módulo RTMC (real Time Monitoring and Control Manager), que además es el encargado de manejar todos los flujos de información entre los diferentes módulos del sistema DINGPOS

2.3 GCCS (Geodetic Cloud Computing Service)

GCCS es un servicio de geolocalización operacional ofrecido por el proyecto GPSdancer [7]. Básicamente GPS Dancer es un proceso peer-to-peer en internet para el análisis geodético preciso de señales GPS.
El sistema GPS Dancer es un proyecto voluntario desarrollado por la Asociación Internacional de Geodesia. Se define como un Dancer un programa informático para analizar los datos recibidos de un receptor GPS. Por si mismo, un peer Dancer no es capaz de computar información completa, tal y como podría ser las orbitas de los satélites, relojes de satélites, o parámetros de rotación de la tierra, sin embargo, los peers se conecta a través de internet con otros peers formando una red global de computadoras conectadas formando una red de dimensiones mundiales, lo cual brinda una capacidad de procesamiento muchísimo mayor.

2.4 XYBRID Cloud [8]

XYBRID Cloud ofrece nuevas opciones a dispositivos que requieren un posicionamiento riguroso y fiable, pero no necesariamente en tiempo real o bien donde los requerimientos de potencia o coste pueden exigir el uso de un chip GNSS de alta sensibilidad impracticable. El servicio XYBRID Cloud ofrece una interfaz simple a través de la cual los dispositivos de los clientes pueden subir a la nube instantáneas de las señales satélite crudas y además incorporarle alguna información adicional como medidas barométricas, listas de puntos de acceso WiFi, ID de las celdas, etc para hacer un request de su posición.
El diagrama básico de funcionamiento puede verse en la siguiente figura (Nima Sadrieh, Behnam Aminian and Muhammad Haris Afzal):

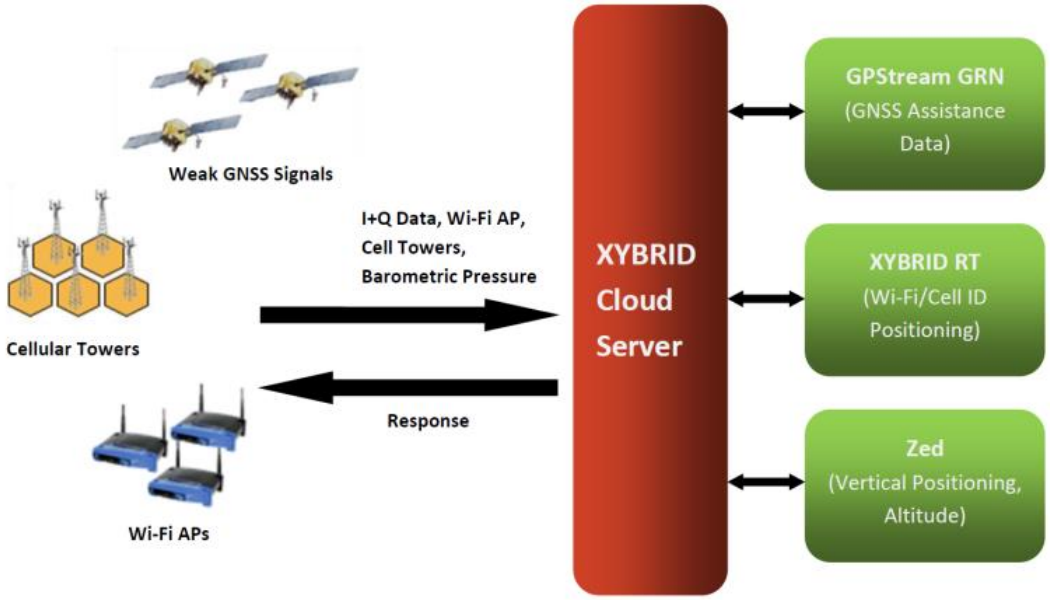

Figura 3. Diagrama de bloques sistema XYBRID

En el lado del servidor XYBRID Cloud, primero se recupera de la señal GPS la señal y mensaje de navegación. Después si el cliente que envía dicha señal detecta la ID de alguna celda o punto de acceso WiFi XYBRID RT utiliza esa información para proveer un posicionamiento 'aproximado'. En el caso de que el cliente envíe información barométrica, el receptor ZED la utiliza para estimar la altitud y proveer ciertas limitaciones verticales a XYBRID Cloud. Después de recibir la información requerida a otros servicios, el motor de posicionamiento procesa la señal GNSS recibida.
La unidad de procesamiento GNSS utiliza algoritmos de posicionamiento state-of-the-art desarrollados por Rx Networks para hacer correcciones de posición precisas.
Básicamente, estos algoritmos usan un tiempo de integración coherente largo, es decir receptores de alta sensibilidad con tiempos de integración de entre 10 ms y 20 ms.
Como sabemos hay varios inconvenientes en el incremento de los tiempos de integración, el de mayor importancia es el relacionado con la duración del periodo de la señal de navegación C/A en la frecuencia L1 (como veremos más

adelante al explicar el funcionamiento de los receptores GPS). Otro gran desafío para aumentar el tiempo de integración coherente es la carga computacional.

Por otra parte, si los receptores permanecen estacionarios por largos periodos de tiempo, XBRID Cloud tiene la oportunidad de recoger los datos GNSS durante un tiempo elevado, lo que no sólo proporciona más medidas para el motor de posicionamiento si no que mejora la Dilución en la Precisión (DOP)

Señalar también que el uso de largos tiempos de observación y el uso de servidores a nivel de operador, permiten el uso de técnicas de mitigación del multi-camino punteras.
En XYBRID Cloud existen dos puntos clave que afectan a la precisión y a la carga computacional de cada cálculo de la posición:

Posición aproximada (Coarse Position): Aunque teóricamente XYBRID Cloud no tiene dependencia con la posición gruesa, sin embargo, esta posición aproximada puede reducir de manera drástica la carga computacional de XYBRID Cloud. Así pues, para reducir los costes XYBRID Cloud acepta cualquier información relacionada con la posición aproximada que permita optimizar el proceso. La posición gruesa puede ser algo tan aproximado como una ciudad o un estado.
Al tener acceso a XYBRID RT, es posible para los dispositivos de los clientes enviar un listado de puntos de acceso WiFi o IDs de las celdas detectadas en la cercanía para estimar la posición aproximada.

Tiempo aproximado (Coarse Time): XYBRID Cloud requiere a los dispositivos de los clientes marcar el tiempo de captura de la señal GNSS cruda antes de enviarla al servidor. La precisión de esa marca de tiempo es un factor determinante que afecta tanto a la precisión en la posición como a la carga computacional. Existen cuatro niveles de precisión en el timming de la señal:

- Nivel A: Dispositivos que usan Network Timing Protocol (NPT) con una precisión mejor de 10ms
- Nivel B: Dispositivos que pueden suministrar una información sobre el timming con una precisión mejor que 100 ms
- Nivel C: Dispositivos que no pueden garantizar información sobre el timming de la señal con precisión mejor a 100 ms, aunque su tiempo relativo entre capturas es mejor que 100 ms
- Nivel D: Dispositivos que pueden proporcionar una precisión en el timming mejor de 60 s

2.5 Ground Based Scintillation Climatology (GBSC) [9]

El centelleo ionosférico (Ionospheric scintillation) es la variación rápida de las ondas de radio causada por estructuras de pequeña escala en la ionosfera (variaciones de decenas de metros a decenas de km en la densidad de electrones en la ionosfera). Cuando se dan severas condiciones de centelleo el receptor GNSS puede tener serios problemas para determinar la posición.
Es sabido que en función de determinados efectos solares (perturbaciones, fluctuaciones en las radiaciones, tormentas solares, etc.) se produce una

variabilidad en las partículas (electrones entre otros) y condiciones electromagnéticas en el llamado espacio cercano a la tierra, lo que puede dañar tanto las tecnologías en tierra como en el propio espacio. Una de las ramas de la ciencia que se ocupa de este fenómeno se conoce como "estudio del clima espacial" y lo que pretende es prever el estas variaciones en el clima espacial, así como su potencial impacto en sistemas biológicos y tecnológicos. Las tecnologías satélites están especialmente amenazadas por estos fenómenos relacionados con el clima espacial por dos motivos, primero están totalmente expuestos a ráfagas de partículas de alta energía que pueden deteriorar el hardware y segundo porque la ionosfera introduce un retardo en la propagación en la señal GNSS lo que representa la principal incertidumbre en términos de posicionamiento. Las señales trans-ionosféricas pueden encontrarse con variaciones repentinas en su fase y amplitud debida a irregularidades ionosféricas. La IS representa una amenaza real y en caso de una super tormenta solar, los sistemas GNSS podrían quedar parcialmente o incluso totalmente inoperativos de uno a tres días.
Una solución novedosa en el estudio del IS es la Ground Based Scintillation Climatology (GBCS), que consigue identificar en que sectores de la ionosfera es más factible que se produzca el centelleo, mostrándolo en mapas bidimensionales. Además, GBCS es capaz de mapear la calidad y características de las señales GNSS, como por ejemplo la relación C/N0. Debido a la gran carga computacional que se requiere, GBCS está en uso en la actualidad sólo para estudios científicos offline y no en tiempo real. Sin embargo, para solventar este problema y dar una solución en tiempo real, GBCS se ha implementado en entornos cloud computing.
Cloud computing ha sido considerado como mejor alternativa frente a Grid Computing para aplicaciones científicas que tiene un uso intensivo de datos. El acercamiento mediante Cloud proporciona varias ventajas, incluida la posibilidad de customizar los entornos en los que corren las aplicaciones, modificarlos y reorganizarlos sin modificaciones significativas. Además, la tecnología de virtualización que es la base del Cloud Computing permite correr diferentes sistemas operativos dentro de una misma máquina física, lo que tiene varias ventajas, entre las más importantes la reducción en cuanto al consumo de energía, mejor tolerancia al fallo, mejoras en cuanto al tiempo de instalación de un dispositivo.
GBSC es actualmente capaz de digerir datos de receptores GNSS con alto ratio de sampleo (GISTM, TEC Monitor, PolaRx y SCINTMON) y elaborar 2 clases de mapas:

- Occumap: mapas donde se muestran las ocurrencias que estén por encima de un límite de distintos parámetros (índices de centelleo, TEC, ROT, C/N0, desviación estándar de la portadora de código)
- Meanmap: Mapas donde se muestra el valor medio y la desviación estándar de los parámetros mencionados arriba.

La infraestructura de la nube se basa en IaaS (Infraestructure As A Service) y se trata de una nube privada basada en OpenNebula3.
OpenNebula3 es un estándar industrial basado en código abierto para la virtualización de data centers.

La virtualización de las máquinas se basa en KVM (Kernel-based Virtual Machine) en Linux, donde cada VM interactúa con su propio hardware virtual permitiendo correr múltiples máquinas virtuales corriendo imágenes Linux
Para el almacenamiento se usa SSHFS (Secure Shell File System) que permite montar de manera remota un sistema de archivos

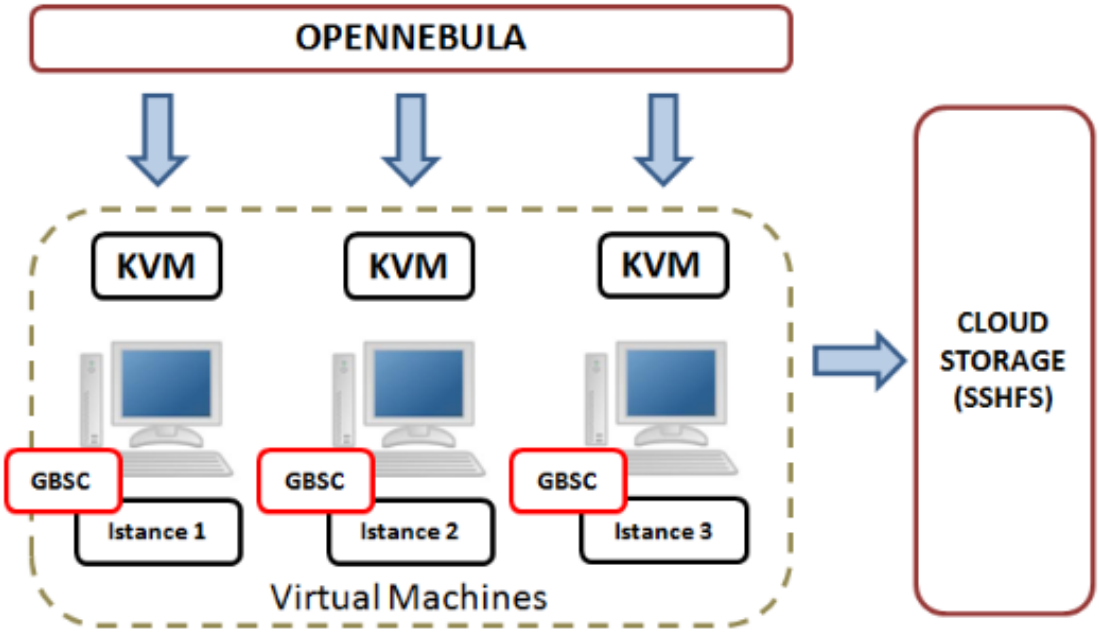

Figura 4. Diagrama de bloques de GBSC

2.6 Soluciones comerciales en IoT tracking

En la actualidad existen varias compañías que ofrecen soluciones de seguimiento y rastreo para múltiples aplicaciones, por ejemplo, para el rastreo de dispositivos IoT. En este contexto la empresa Sequans [11] ofrece la solución CLOE que ofrece una plataforma para el seguimiento de dispositivos IoT en el cual el procesamiento de la señal GPS se combina con las medidas obtenidas del chip LTE y los resultados se envían a la nube
Otra solución en el mismo sentido es el de la compañía Accent System [12] que ofrece NB-IoT Tracket, que ofrece un interfaz NB-IoT (*Narrowband IoT)* y Bluetooth Low Energy para el seguimiento de los dispositivos IoT, igual que con CLOE los resultados se envían a la nube.
Tenemos también la empresa Tracktio [13] que ofrece una solución de posicionamiento basada en GPS y LORA (Low-Power Wide-Area)

3. Principio de funcionamiento GNSS

3.1. Estructura de la señal GPS

El principio de funcionamiento de los sistemas de posicionamiento por satélite se basa en medir el retardo en la propagación de una señal emitida desde un satélite visible hasta que esta llega al receptor. Los satélites emiten señales de navegación, en dichas señales el satélite marca el tiempo en el que la señal se transmite (ts), así el receptor conociendo el tiempo en el que la señal es recibida (tu) es capaz de determinar el retardo en la propagación de la señal $\tau = tu - ts$. Calcular la distancia entre el satélite y el receptor es directo $d = c \cdot \tau$.

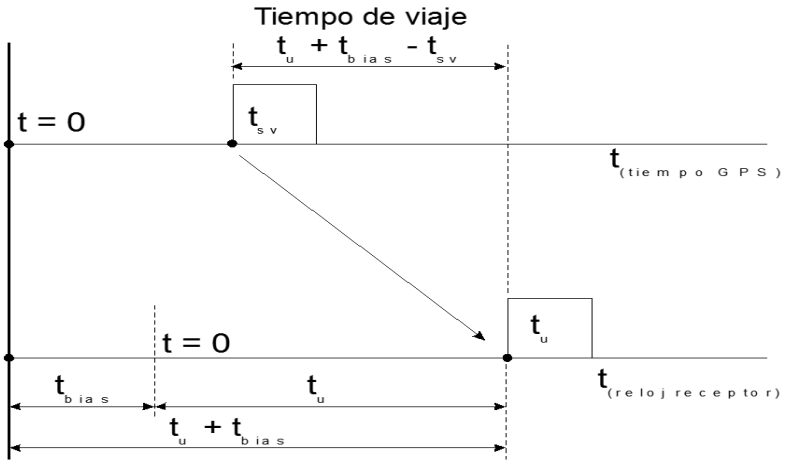

Figura 5. Tiempo viaje señal GNSS

El principio de funcionamiento GNSS es muy conocido desde hace tiempo y por tanto sólo vamos a dar una visión general, igualmente vamos a definir unos conceptos básicos que ayuden a comprender el problema al que nos enfrentamos.
Podemos ver en la siguiente imagen las etapas involucrados en el uso de GNSS para determinar el tiempo y la posición de la aplicación (receptor)[10]:

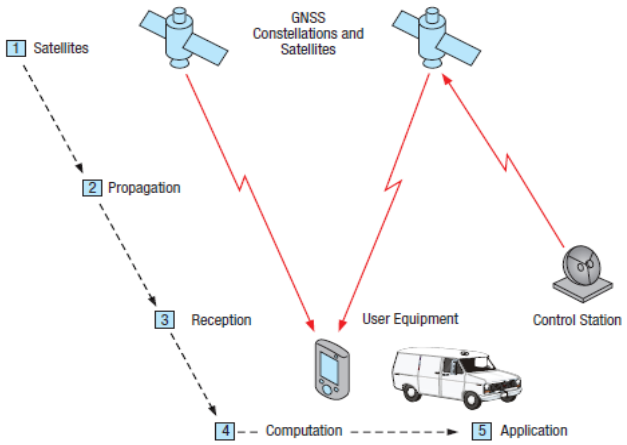

Figura 6. Etapas uso GNSS [10]

En una primera etapa tenemos a la constelación de satélites que orbitan la tierra (los satélites de la constelación GPS orbitan a una altitud de 20.000 km,

mientras que los satélites de la constelación Galileo lo hace un poco más alto, a 23.000 km de altitud). Los satélites conocen los distintos parámetros (efemérides) que definen su órbita, así como el tiempo de manera muy precisa (la última generación de satélites GPS usa relojes de rubidio con una precisión inferior a ±5 partes en 10^{11}.

Las señales satélite son complejas. En el caso de GPS (es ahora mismo el sistema más extendido en dispositivos de posicionamiento por satélite) las señales están basadas en los códigos llamados C/A (Coarse Acquisition code) y el código militar P (Precise). Estas señales se emiten en tres bandas diferentes:

- L1: Banda centrada en FL1=1.575,42 MHz (λ= 19 cm)
- L2: Banda centrada en FL2=1.227,60 MHz (λ= 24 cm)
- L5: Banda centrada en FL5=1.176,45 MHz (λ= 25 cm)

GPS trabaja con esquema de transmisión llamado CDMA (Code Division Multiple Access), es decir cada señal satélite está modulada por una única pseudosecuencia (pseudo código) digital o código y los receptores conocen el código de cada satélite. Las señales CDMA tiene muy baja potencia, pero mediante la correlación el receptor es capaz de saber a qué satélite pertenece la señal, y recuperar la información que contiene.

La señal transmitida por el satélite k-ésimo se puede expresar como:

$$S_k(t) = \sqrt{2P_c}C_k(t)D(t)\cos(2\pi F_{L1}t) + \sqrt{2P_{PL1}}P_k(t)D_k(t)\sin(2\pi f_{L1}t) + \sqrt{2P_{PL2}}P_k(t)D_k(t)\sin(2\pi f_{l2}t)$$

Donde Ck(t) es la señal transmitido por el satélite k-esimo asociada al código C/A (Coarse/Adquisition) y se trata de una señal binaria con amplitud -1 (cero lógico) y 1 (uno lógico), Pk(t) es la señal transmitida por el satélite k-esimo asociada al código P (Precise) y también es una señal de amplitud -1 y 1. Estos códigos no transportan información y son códigos de ensanchamiento (spreading), son secuencias de bits, denominados chips, con un periodo mucho menor que el del mensaje de navegación D(t), Ck(t) 1,023 Mega chips por segundo y Pk(t) 10,23 Mchips segundo
D(t) por su parte es el mensaje de navegación, es una señal binaria contiene la información relativa a los satélites, las llamadas efemérides, con información relativa a la posición, órbitas, etc. y que se transmite a 50 bps
L1, L2 son las portadoras, y es una señal de tipo sinusoidal, con las frecuencias anteriormente mencionadas, fL1=1.575,42 MHz y fL2= 1.227,60 MHz
Podemos ver un esquema de la generación de la señal satélite GPS para las bandas L1 y L2 en la siguiente figura

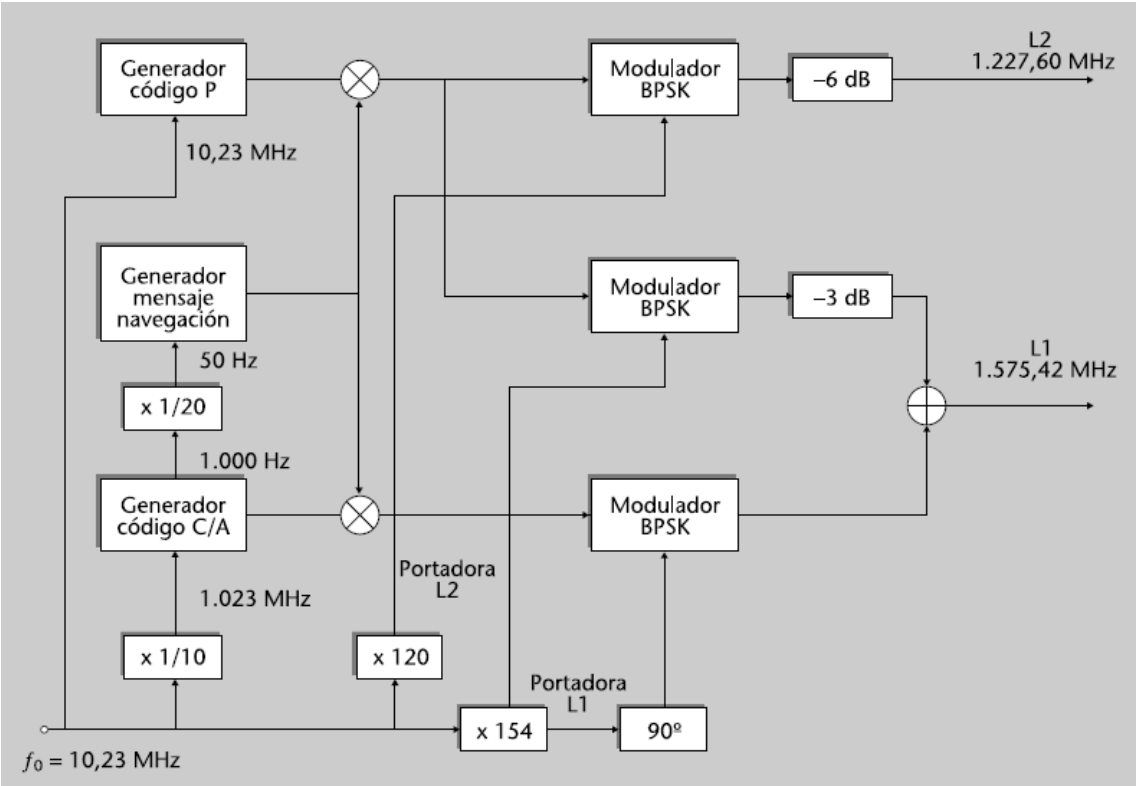

Figura 7. Generación señal GPS (L1 y L2)

Básicamente lo que se hace es, utilizando mezcladores en la banda L1 se mezla el código C/A más el mensaje de navegación D(t) con la portadora L1. En L2 además del código C/A también se transporta el código militar P.

El proceso de modulación puede verse en la siguiente figura:

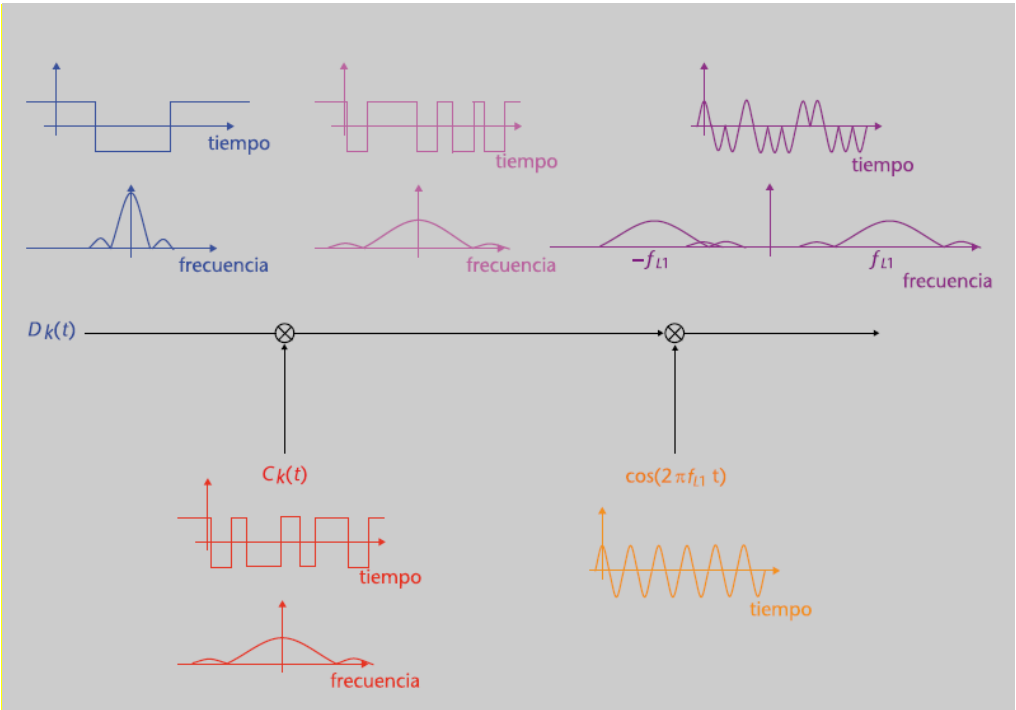

Figura 8. Proceso de modulación señal GPS (1)

Como vemos, se utiliza modulación BPSK, es decir, en función si hay un “-1” o un “1” la señal cambia de fase, de 0º a 180º

De manera muy simplificada, podemos ver en la siguiente figura, como se genera la señal C/A en L1:

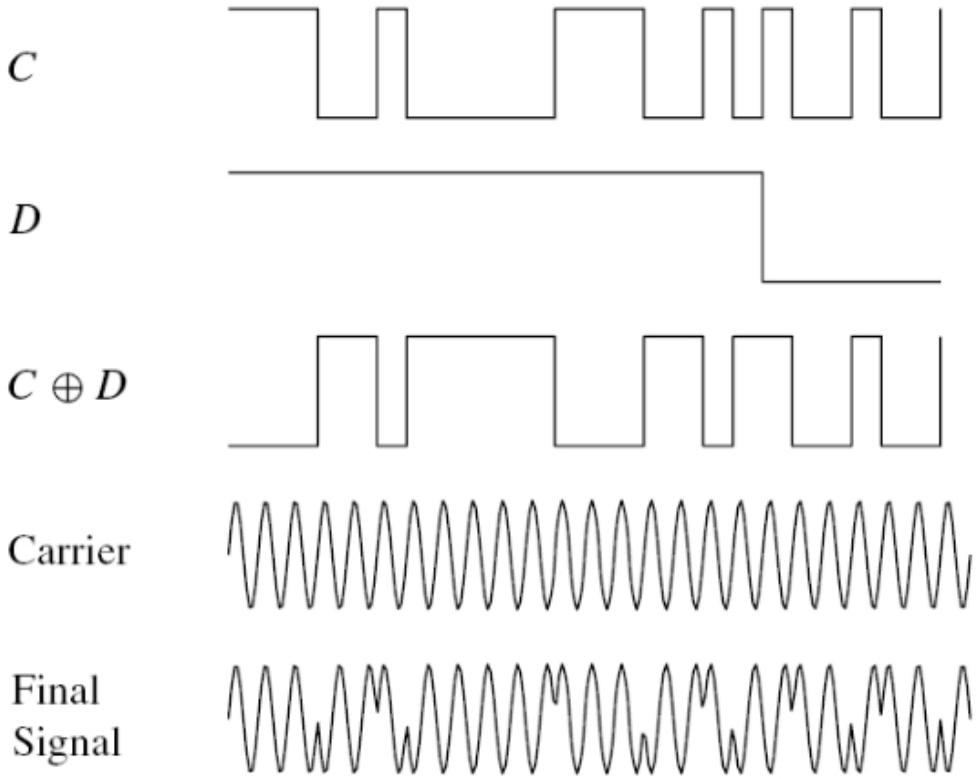

Figura 9. Proceso de modulación señal GPS (2)

En el receptor GPS hay una copia de cada uno de los códigos de los distintos satélites, así que cuando llega una señal lo que hace es comparar la señal recibida con la copia local. En función del retraso entre ambas señales es capaz de determinar su propia posición.
Veremos que ademas, debido a que los satélites y el propio recptor (en ciertas aplicaciones) están en movimiento, la señal procedente del satelite se encuentra afectada al llegar al receptor por el efecto doppler. Así pues en el receptor, se trata de estimar el retraso en el PRN recibido comparandolo con una copia local y la frecuencia Doppler.

Básicamente lo que tenemos son dos parámetros de busqueda que nos definen una función de coste en dos dimensiones. Estos dos parámetros de busqueda, retardo y doppler definen lo que se llama una celda de busqueda. El conjunto de celdas de busqueda definen una matriz de correlaciones para los distintos retrasos y frecuencias doppler que definen el llamado espacio de busqueda de adquisición:

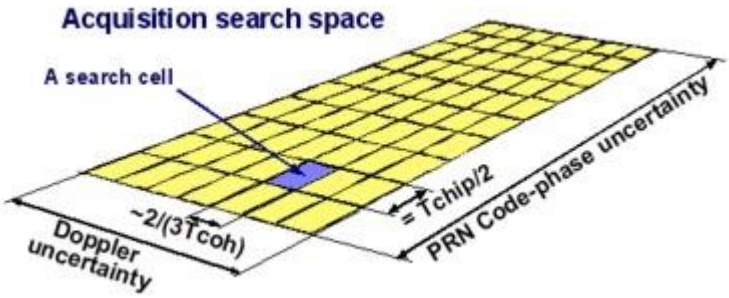

Figura 10. Espacio búsqueda de Adquisición GNSS

3.2. Estructura del receptor GPS

De una manera muy básica vamos a ver como funciona la etapa de adquisición de la señal en un receptor GPS (y en general cualquier receptor GNSS). Podemos ver el siguiente diagrama de bloques básico:

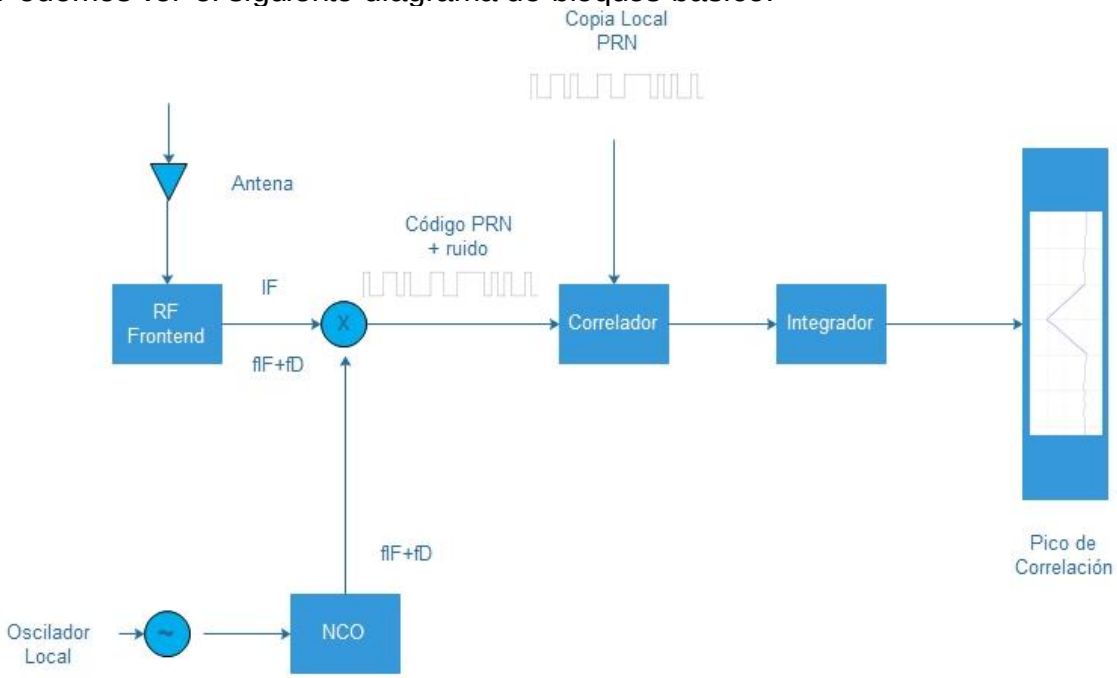

Figura 11. Diagrama bloques receptor GPS

Antes de entrar a ver con más detalle el receptor GPS veamos lo que tenemos a la salida del satélite. Vamos a ver una versión muy simplificada en la que sólo se emite el código C/A en la banda L1(obviamos el código militar P y la banda L2) y veremos que como se procesa dicha señal por el receptor:

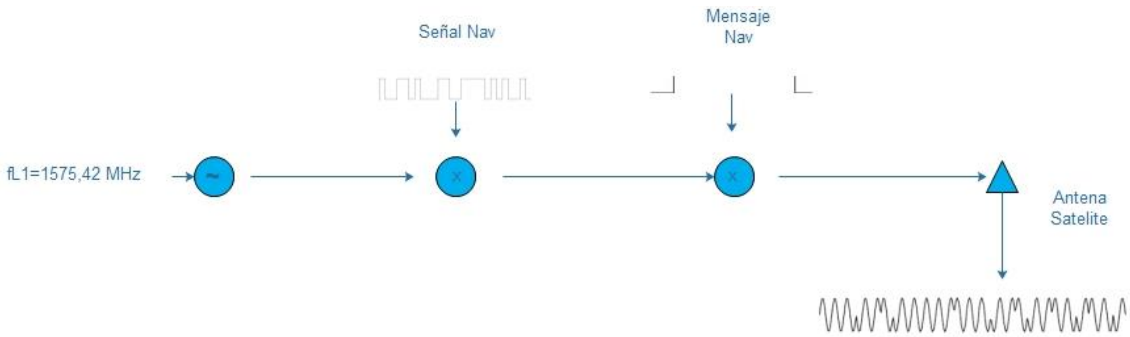

Figura 12. Diagrama bloques señal salida satélite

La señal de salida la podemos expresar como:

$$S_k(t) = \sqrt{2P_c}C_k(t)D(t)\cos(2\pi f_{L1}t)$$

Sin embargo, tanto el satélite como el receptor están en movimiento, por lo que la señal que llega al receptor será:

$$S_k(t) = \sqrt{2P_c}C_k(t-\tau)D(t-\tau)\cos(2\pi(f_{L1}+fD)t+\phi)$$

Como vemos la señal llega al receptor con un determinado retraso $\tau = tiempo\ de\ propagación$ además la frecuencia de la señal se le añade el efecto Doppler debido a que el satélite y el receptor están en movimiento y también un cierto desfase.

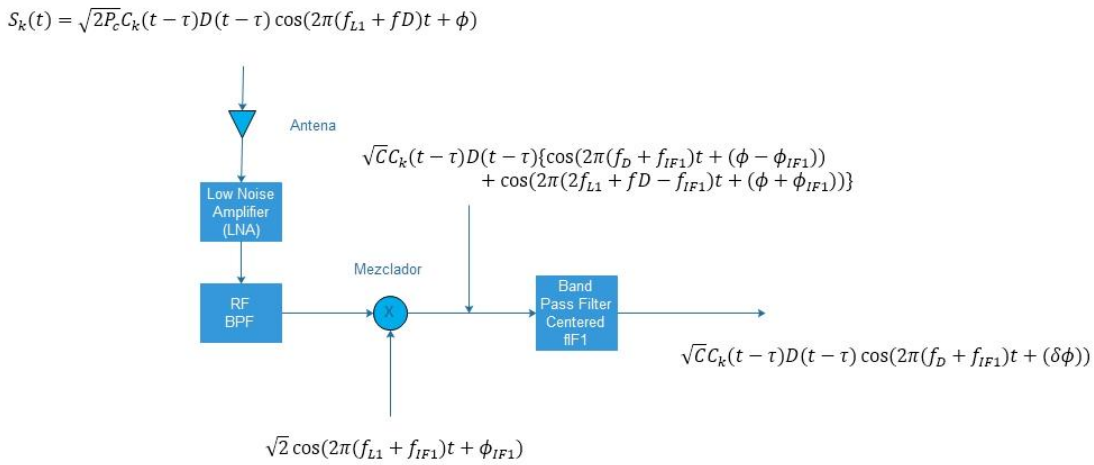

Figura 13. Etapa FI en el receptor

La señal, tras ser amplificada y filtrada se mezcla con la señal: $\sqrt{2}\cos(2\pi(f_{L1}+f_{IF1})t+\phi_{IF1})$

Como podemos ver, aplicando razones trigonométricas, de esta forma,a la salida del mezclador tenemos una señal cuya amplitud C dependerá de la ganancia de la antena y a su vez tendremos el coseno de la suma de frecuencias de las señales mezcladas y el coseno de la diferencia de dichas frecuencias

$$\sqrt{C}C_k(t-\tau)D(t-\tau)\{\cos(2\pi(f_D+f_{IF1})t+(\phi-\phi_{IF1}))$$
$$+\cos(2\pi(2f_{L1}+f_D-f_{IF1})t+(\phi+\phi_{IF1}))\}$$

Si ahora hacemos pasar la señal por un filtro paso banda centrado en IF1 tendremos:

$$\sqrt{C}C_k(t-\tau)D(t-\tau)\cos(2\pi(f_D+f_{IF1})t+(\delta\phi))$$

Vemos que la señal, está aún afectada por el efecto doppler, así que hacemos un sampleado de la misma en fase y cuadratura para eliminar dicho efecto:

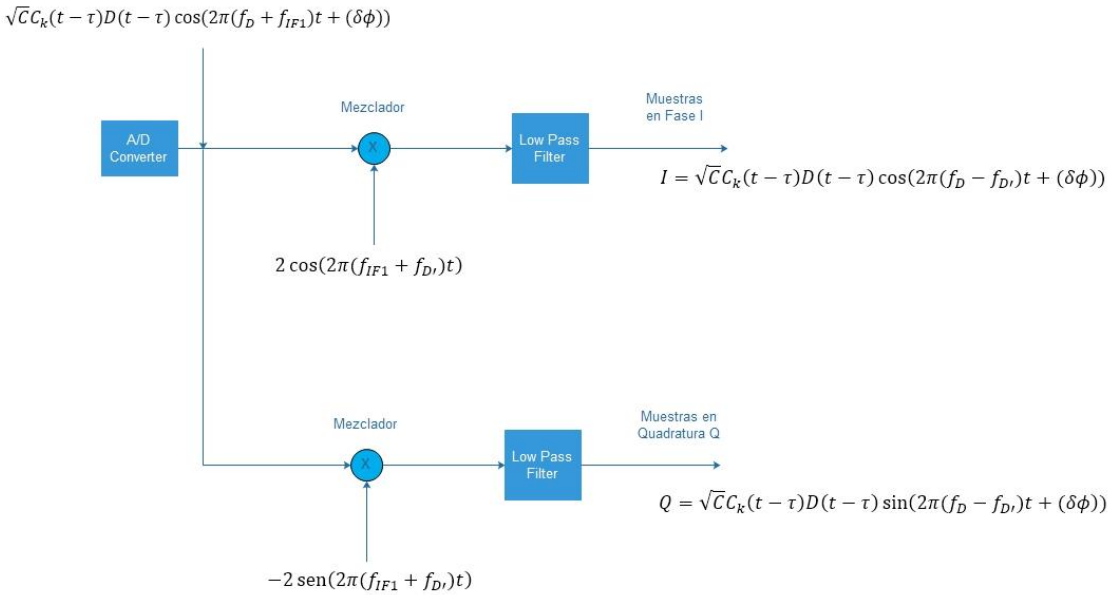

Figura 14. Ramas en fase y cuadratura receptor GPS

Por una parte tenemos la rama en fase, donde la señal se mezcla con la señal $2\cos(2\pi(f_{IF1}+f_{D'})t)$, donde la frecuencia intermedia es conocida, pero **la frecuencia dopppler $f_{D'}$ es estimada**. Tras pasar por un filtro paso bajo la señal a la salida de la rama en fase será:

$$I=\sqrt{C}C_k(t-\tau)D(t-\tau)\cos(2\pi(f_D-f_{D'})t+(\delta\phi))$$

Por otra parte tenemos la rama en cuadratura, donde ahora la señal se mezcla con $-2\,\mathrm{sen}(2\pi(f_{IF1}+f_{D'})t)$, donde la frecuencia intermedia es conocida, pero **la frecuencia dopppler $f_{D'}$ de nuevo es estimada**, de manera que tras pasar por un filtro paso bajo, la salida de la rama en cuadratura será:

$$Q=\sqrt{C}C_k(t-\tau)D(t-\tau)\sin(2\pi(f_D-f_{D'})t+(\delta\phi))$$

Veamos ahora que pasa en la etapa de correlación en el receptor:

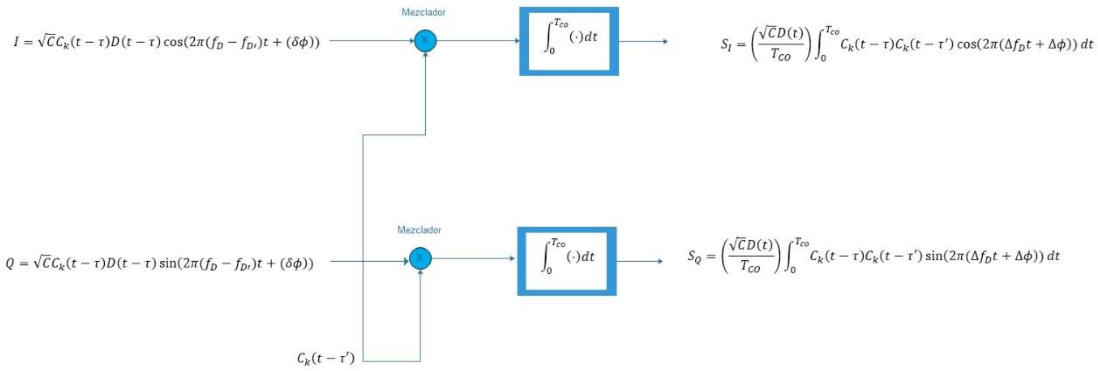

Figura 15. Correlación receptor GPS

Las muestras en fase y cuadratura se mezclan con la copia local, donde **el tiempo de propagación τ' es una estimación**: $C_k(t-\tau')$

A la salida de los correladores en fase y cuadratura tendremos:

$$S_I = \left(\frac{\sqrt{C}D(t)}{T_{CO}}\right)\int_0^{T_{co}} C_k(t-\tau)C_k(t-\tau')\cos(2\pi(\Delta f_D t + \Delta\phi))\,dt$$

$$S_Q = \left(\frac{\sqrt{C}D(t)}{T_{CO}}\right)\int_0^{T_{co}} C_k(t-\tau)C_k(t-\tau')\sin(2\pi(\Delta f_D t + \Delta\phi))\,dt$$

Como Δf_D no es exactamente cero, ya que la frecuencia Doppler es estimada, el PRN que obtenemos tiene una cierta componente sinusoidal:

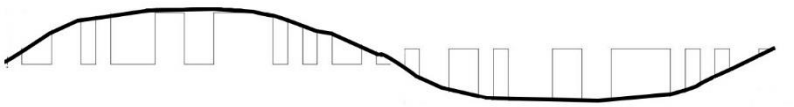

Figura 16. Componente sinusoidal del código

Un poco más adelante vamos a ver como el uso de los canales en fase y cuadratura en el receptor nos ayuda a eliminar esa componente sinusoidal introducida por el efecto Doppler.

Como hemos dicho, en el proceso de correlación, básicamente lo que se hace es comparar la señal recibida con la copia local. En el receptor, la copia local se va variando el valor de τ de manera que el valor de correlación va variando. Cuando este valor supera un valor límite de detección fijado, tendremos la copia local alineada con la recibida, lo que con el valor de τ nos permite definir la posición del receptor:

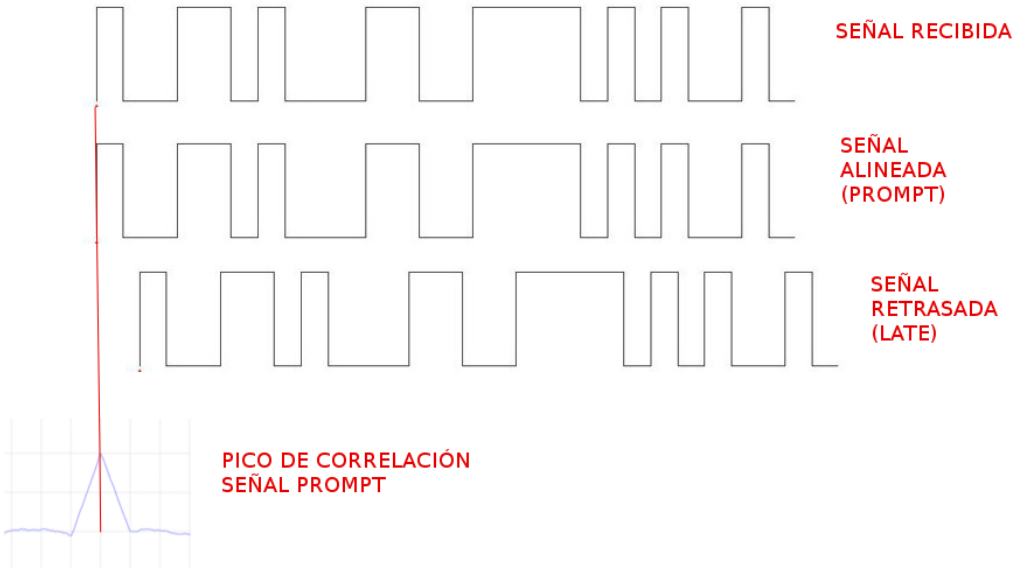

Figura 17. Pico de correlación señal Prompt

Como vimos los códigos PRN tienen propiedades de autocorrelación muy buenas (picos bien definidos y afilados) mientras que la correlación cruzada entre ellas es muy mala. De esta forma podemos identificar el satélite desde el que se emite la señal recibida.

Para poder obtener el pico de correlación, veíamos que era necesario un proceso de integración:

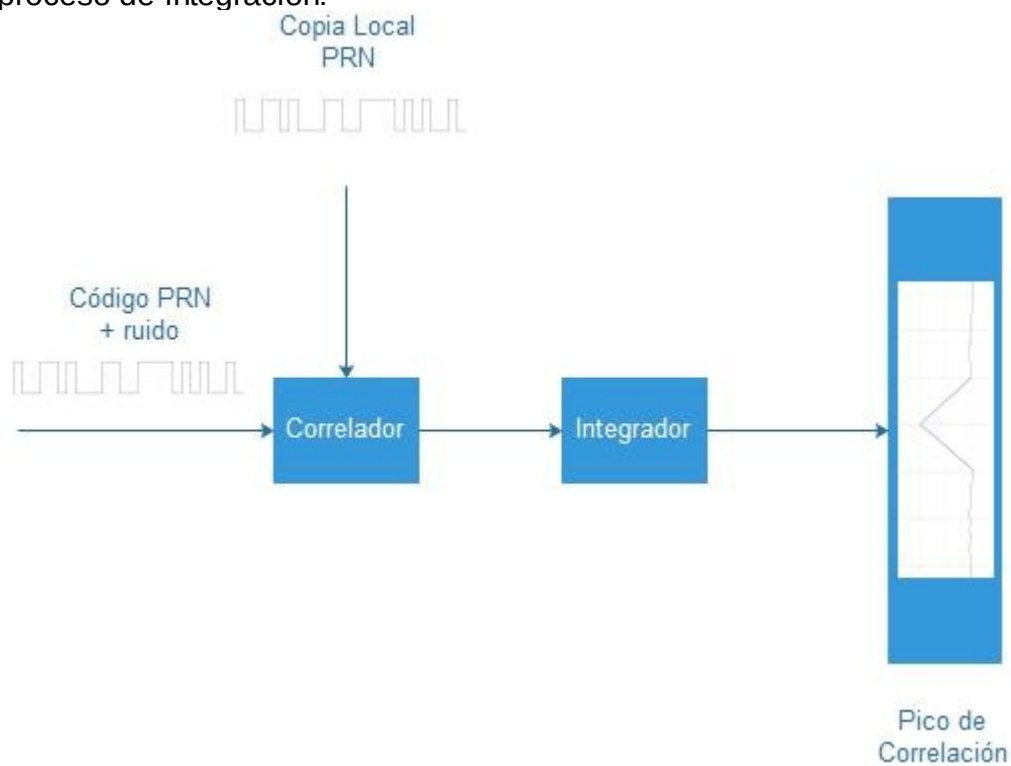

Figura 18. Diagrama bloques. Correlación en Integración

Es decir, como vimos, la señal procedente del satélite es muy débil y viene afectada por el ruido (teóricamente incorrelado), por lo que es necesario hacer un proceso de integración del proceso de correlación ente las muestras de la señal recibida y las de la copia local. De esta manera podemos detectar la señal GPS sobre el ruido.
Podría parecer que a mayor tiempo de integración es deseable, ya que de esta manera podemos destacar la señal sobre el ruido. En la práctica eso no es así y veremos que hay límites en el tiempo de integración de la señal.
Tenemos dos tipos de integración, la integración coherente y la integración no coherente.
Se denomina integración coherente porque si hay un cambio en la fase de la señal, este cambio se propaga de manera coherente en el proceso de integración, es decir el pico de correlación cambia de signo.
Podemos ver que la señal procedente del satélite contiene además del código un cierto nivel de ruido.
Cuando realizamos la integración de dicha señal vemos los diferentes ritmos de crecimiento de la magnitud de correlación y de la desviación estándar del ruido. A modo de simplificación vamos a tratarlas como señales independientes, por una parte, tendremos el código y por otra el ruido:

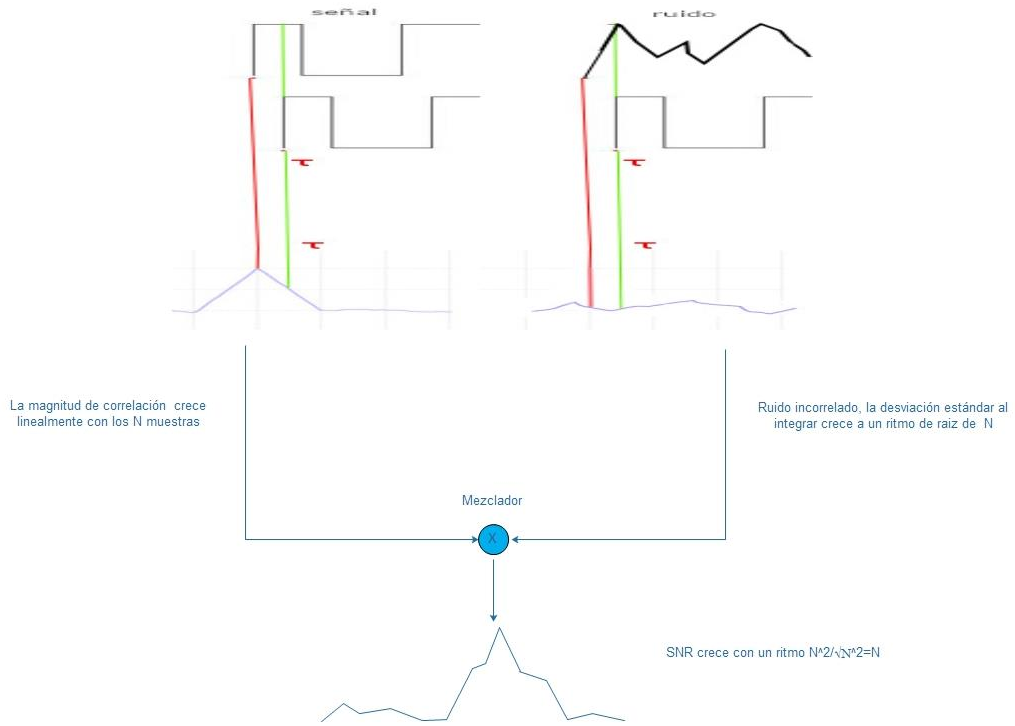

Figura 19. Correlación en Integración con presencia de ruido

Vemos que la magnitud de correlación de la señal crece de manera lineal con el número de muestras N, sin embargo, en el caso del ruido, la desviación estándar crece a un ritmo $\sigma = \sqrt{N}$. Teniendo en cuenta que la relación SNR es una relación de potencias, por lo que, de manera ideal, cuando integramos

crece a un ritmo de $SNR = \frac{N^2}{\sqrt{N^2}} = N$. En la vida real tal ganancia en la relación SNR no es exactamente así ya que el ruido en la vida real no está puramente incorrelado.

Como estamos viendo, parece que lo ideal sería correlar el mayor tiempo posible, para tener un mayor número de muestras a integrar ya que la relación señal a ruido SNR, crece en dB según el número de muestras:

$$SNR_{dB} = 10 \log_{10} N$$

Esto no es exactamente así, hasta ahora lo que hemos supuesto es que lo que nos llegaba hasta el receptor es directamente el pseudocódigo PRN, sin embargo, hemos de recordar que el código PRN viene modulado por el mensaje de navegación Dk(t) a 50bps.
Es decir, que en el receptor sabemos cómo es la señal PRN, pero no sabemos cómo es la señal Dk(t), así un cambio de fase de Dk(t) puede conllevar un cambio de fase en la PRN transmitida pero no de la copia local.
A modo de ejemplo vamos a suponer que integramos por 3 msg:

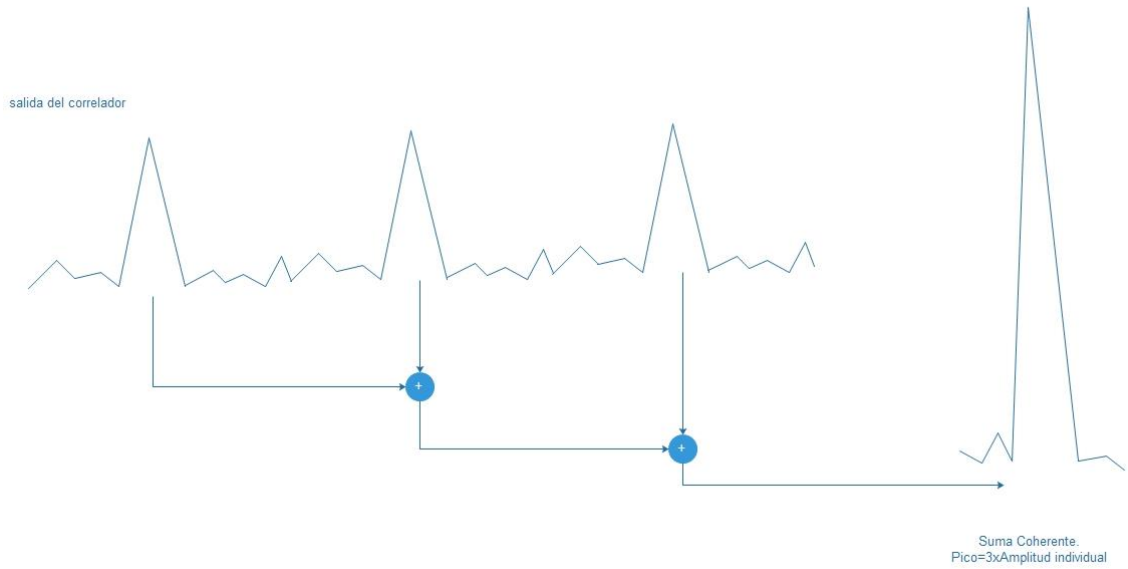

Figura 20. Integración coherente ideal (3msg)

En un caso ideal, no habría cambio de fase en la señal PRN, con lo que con la integración coherente durante 3 msg, conseguiríamos que la amplitud del pico de correlación fuese el triple que en el caso de que el tiempo de integración coherente fuera de 1msg.

Pero veamos que sucede si hay un cambio en la fase de la señal:

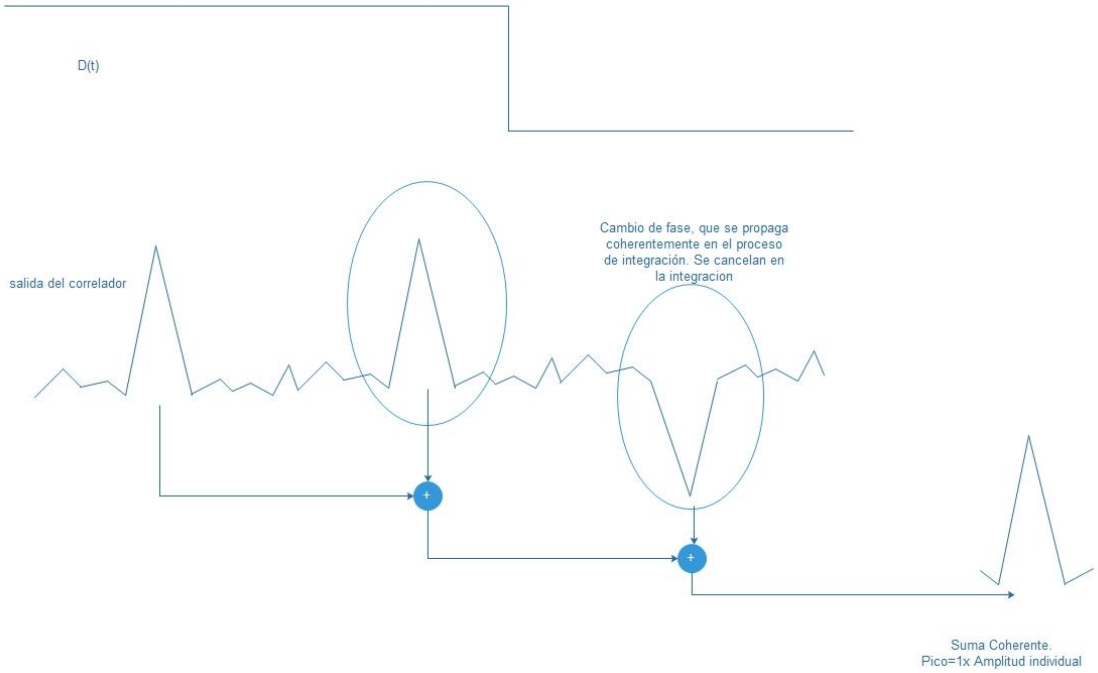

Figura 21. Integración coherente real (cambio de signo)

Como vemos ahora se ha producido un cambio de fase en el mensaje de navegación lo que ha provocado un cambio de fase en el pico de correlación. A la salida de la etapa de integración, dos de los picos de correlación se han cancelado mutuamente.
En la práctica se observa como el pico de correlación va creciendo durante un determinado tiempo, a partir del cual, el pico empieza a desmoronarse y acaba por desaparecer, es decir los picos de correlación positivos se cancelan con los negativos. Podemos decir de manera no rigurosa que el límite en la integración coherente es de 20 ms, que es el periodo de bit de Dk(t), de esta manera evitamos el posible cambio de fase en el código. Por otra parte, el mínimo número de muestras que pueden integrarse de manera coherente será la longitud del código, que para el caso de GPS-L1 será de Tcode=1ms.
Así pues, de manera coherente podemos integrar Ni·Tcode segundos <20ms, siendo Ni el número de integraciones cortas de Tcode=1ms.

A veces, es necesario extender los tiempos de integración más allá, debido a que la energía de la señal llega muy reducida y es necesario aumentar dichos tiempos con el fin de hacer el pico de correlación visible, es la llamada **integración no coherente.** En este caso el cambio de fase en la señal, no se propaga coherentemente desde la etapa de correlación a la de integración. Veamos el diagrama de bloques de la integración no coherente:

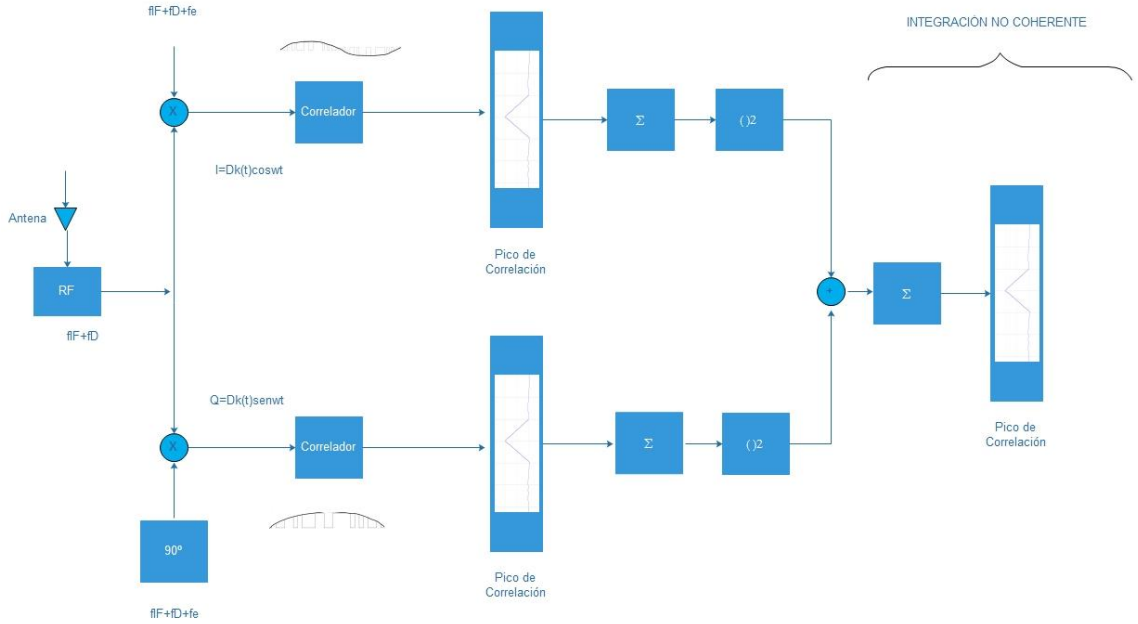

Figura 22. Diagrama de bloques Integración no coherente.

Para corregir el posible cambio de signo que se pueda producir entre dos bits de navegación consecutivos, hacemos el módulo del resultado de la correlación coherente de cada bit de navegación, y lo elevamos al cuadrado, de esta manera conseguimos que aquellos valores pequeños provocados por el ruido se hagan aún más pequeños y no tengan tanto peso en el resultado de la integración no coherente.
Básicamente lo que hacemos es dividir una correlación muy larga en correlaciones coherentes de fragmentos más cortos.

Señalar además que, tanto en la rama en fase como en la rama en cuadratura, se ven afectadas por una componente sinusoidal, es importante señalar que cuando en la rama en fase I tenemos mínima energía, en la rama en cuadratura Q tenemos la máxima y viceversa:

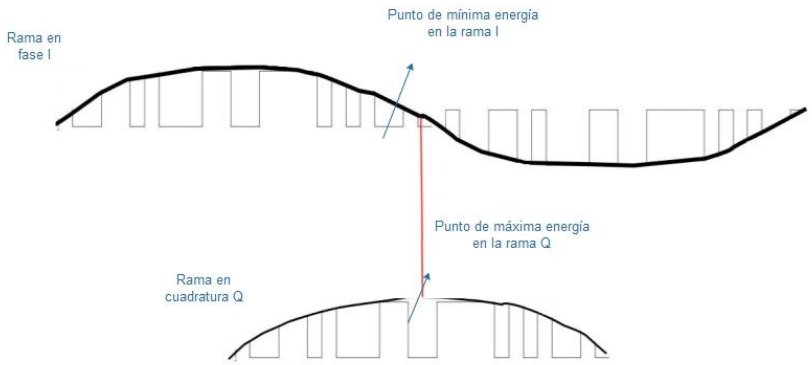

Figura 23. Energías en ramas en fase y cuadratura

El objetivo de utilizar canales I y Q antes de la correlación vendría justificado por la necesidad de eliminar la componente sinusoidal en el PRN, evitando que el error Doppler se vaya acumulando y pueda provocar un plegamiento de fase.

Vemos que, en cada rama, tenemos:

$$I = C_k(t) cos\, wt$$

$$Q = C_k(t) sen\, wt$$

Así pues:

$$I^2 + Q^2 = C_k(t)(\cos^2 wt + sen^2 wt) = C_k(t)$$

Como vemos, de manera teórica, al usar las ramas en fase y cuadratura conseguimos deshacernos de la componente senoidal y nos quedamos con la señal.
Hay que tener en cuenta que correlar es una operación lineal, es decir todo lo que pasa antes de la correlación también pasa después:

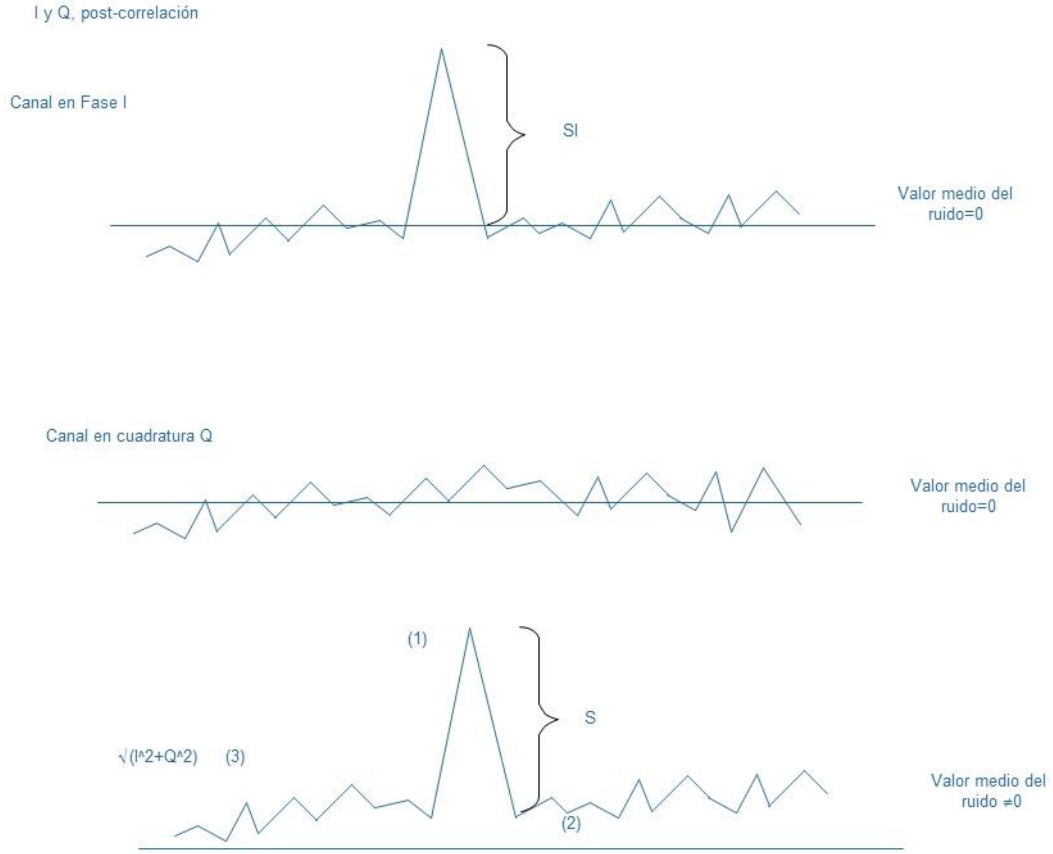

Figura 24. Post-Correlación I y Q.

Vamos a imaginar que toda la energía de la señal está en el canal en fase I, aunque en realidad, la energía se reparte de manera dinámica entre los canales en fase y cuadratura.
Podemos ver en (1) que el valor de amplitud del pico de correlación se reduce debido al ruido, que deja de tener media 0 (2) desde un valor SI a S. Es el denominado efecto *squaring loss*, que supone una degradación en la relación SNR de la señal, no obstante, tenemos la ventaja de que podemos integrar durante más tiempo con lo que será más fácil encontrar el pico de correlación

Es decir, por una parte, con la integración coherente la relación SNR aumenta a un ritmo $SNR = N$ siendo N el numero de términos integrados, pero si los tiempos de integración son largos (del orden de 20msg), se puede producir un cambio de fase en la señal que invierta el signo del pico de correlación, haciendo incluso que este se desmorone.

Por otra parte, tenemos la integración no coherente, donde los tiempos de correlación pueden ser mucho mayores y así el valor del pico de correlación aumenta, pero hemos de tener en cuenta que la relación SNR se degrada (squaring loss). Lo que hacemos es, dividir la correlación de un fragmento muy largo en n correlaciones coherentes consecutivas de Ni muestras que después se acumulan de forma no coherente.

4. Parámetros relevantes en la detección señal GNSS

4.1 Potencia de transmisión y Pathloss

En el caso de GPS los satélites emiten la señal con una potencia muy baja, de unos 27 W o lo que es lo mismo $P_T = 14.3\ dB$. Si consideramos que la señal se emite de manera isotrópica, tendremos que la densidad de potencia disminuye conforme aumenta la distancia de transmisión según:

$$P_r = \frac{P_T}{4\pi R^2}\ donde\ P_T = 27W \rightarrow 14{,}3\ dB$$

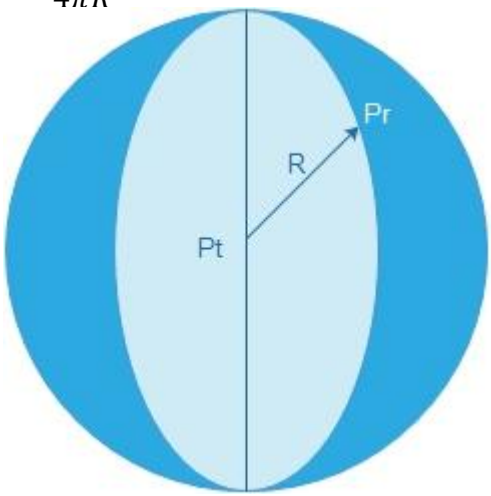

Figura 25. Densidad potencia espacio libre

Si lo expresamos en forma logarítmica, la potencia con la que se recibe la señal será:

$$P_{r_{dB}} = 10\log_{10}\left(\frac{P_T}{4\pi R^2}\right) = 10\log_{10} P_T - 10log_{10}(4\pi R^2)$$

4.2 Ganancia de la antena

En el ejemplo anterior, se ha supuesto una radiación isotrópica. En el caso de la señal satélite eso no es así, si no que es directiva y tiene un área de apuntado:

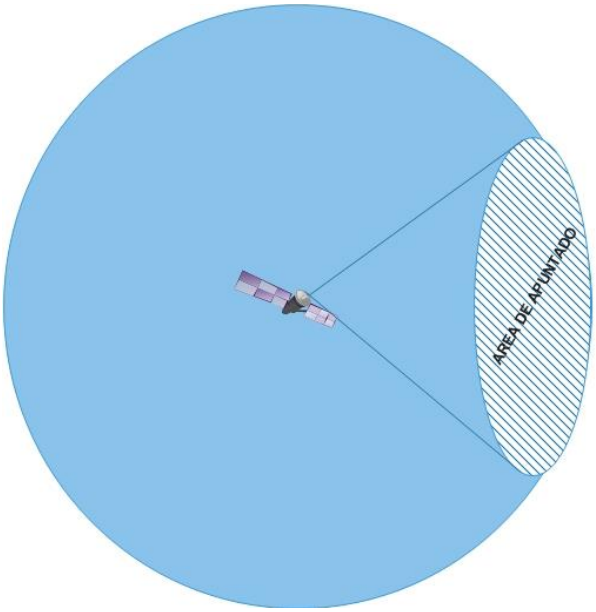

Figura 26. Área de Apuntado

Aparece un nuevo concepto llamado Ganancia de la Antena:

$$Ganancia\ Antena = \frac{area\ de\ la\ esfera}{area\ de\ apuntado}$$

Así pues:

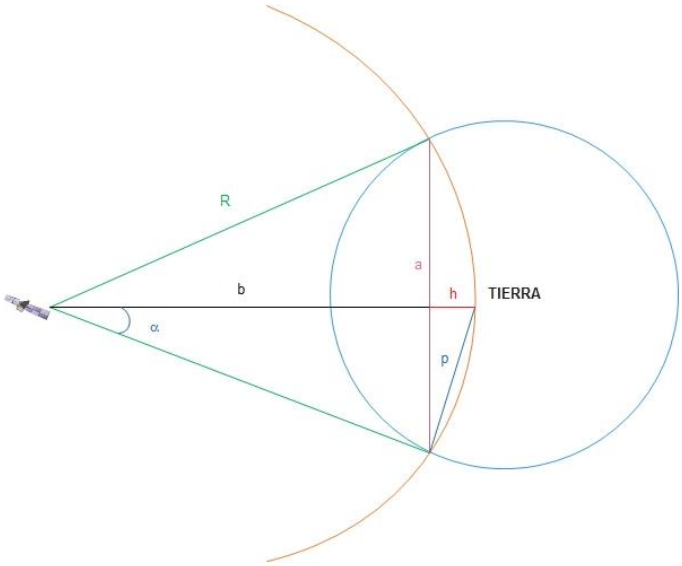

Figura 27. Ganancia Transmisión Antena (1)

$$Area\ de\ la\ esfera = 4\pi R$$
$$Area\ de\ Apuntado = \pi p^2 = 2\pi Rh$$
$$a = Rsin\alpha$$
$$b = Rcos\alpha$$
$$p = \sqrt{h^2 + a^2} = R\sqrt{2 - 2cos\alpha}$$

$$Ganancia\ de\ Transmisión\ Antena = G_T(\alpha) = \frac{4\pi R^2}{\pi p^2} = \frac{2}{1 - cos\alpha}$$

Para el caso de GPS $\alpha = 21{,}3º\ \ P_T = 27\ W$, y la $G_T\ entre\ 10\ y\ 17$
Así pues, la Potencia Efectiva va de 270W a 460W

Las antenas se construyen para que la potencia con la que llega al receptor sea siempre la misma en de manera que cambia la G en función de la dirección (distancia al receptor):

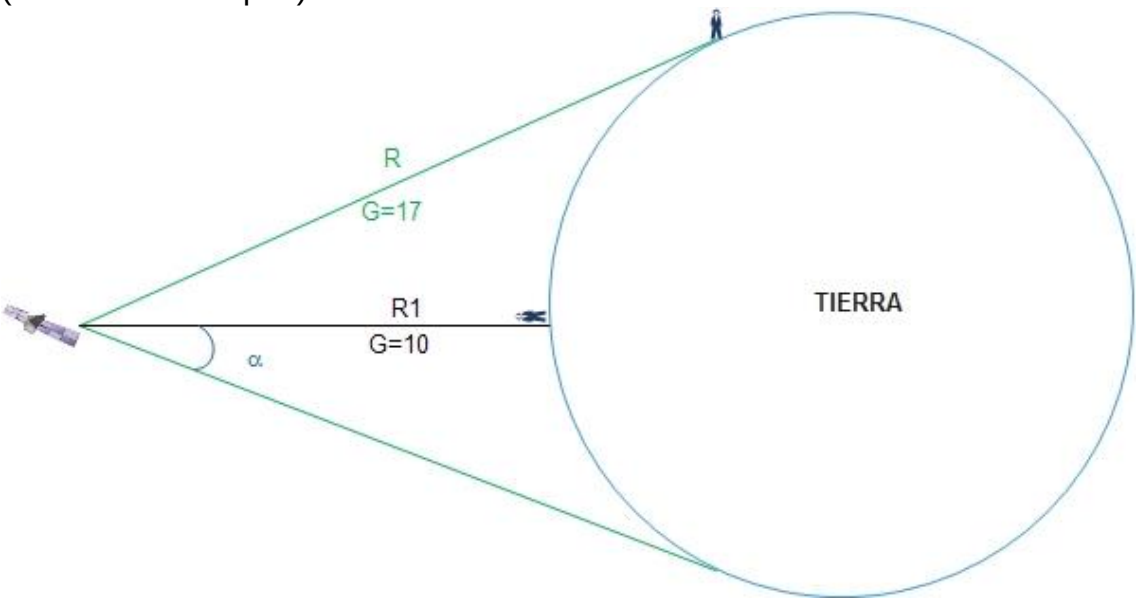

Figura 28. Ganancia Transmisión Antena (2)

La Potencia Recibida puede expresarse como:

$$Potencia\ Recibida = \frac{(P_T G_T G_A)}{4\pi R^2}$$

Donde G_A es la ganancia de la antena del receptor.

De manera general, para GPS podemos decir que:

$$Potencia\ Recibida = 1{,}41x10^{-16} \rightarrow\ -128{,}5\ dBm$$

4.3 C/No, temperatura de ruido, fórmula de Friis

Se denomina C/No como la relación de potencia entre la portadora de la señal y la densidad de ruido:

$$\frac{C}{No}\frac{(watts)}{(\frac{watss}{Hz})} \rightarrow \frac{C}{No}\ (Hz)$$

Se expresa en dB.Hz:

$$10\log_{10}\left(\frac{C}{No}\right) = \left(\frac{C}{No}\right)_{dB.Hz}$$

Expresando esta ratio de potencia en función del ancho de banda BN tenemos:

$$\frac{C}{P_N} = \frac{C}{NoB_N}$$

donde P_N es la Potencia de ruido y No la densidad de ruido, es decir la potencia de ruido por cada herzio.

Por otra parte, también se define la densidad de ruido como:

$$No = K \cdot Teq$$

$$K = \ constante\ de\ Boltzman = 1{,}38x10^{-23}\ J/K$$

$$Teq = Temperatura\ Equivalente$$

Así pues, podemos calcular la potencia de ruido como:

$$P_N = K \cdot Teq \cdot B_N$$

Recordando el diagrama de bloques del receptor GPS:

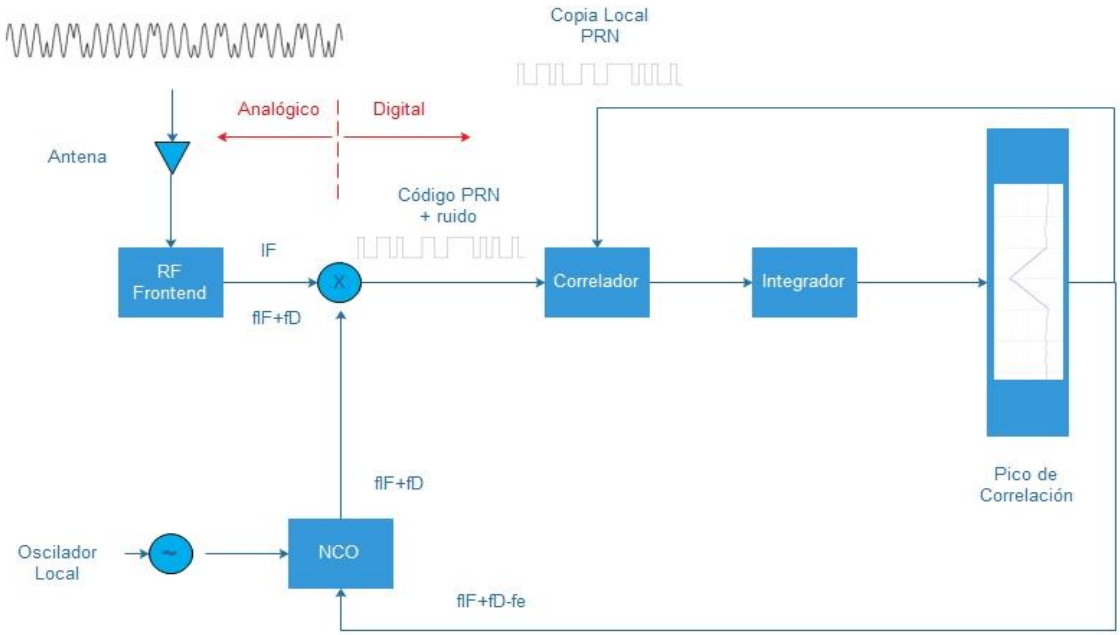

Figura 29. Diagrama Bloques canal del receptor

Si nos fijamos en la parte analógica tenemos:

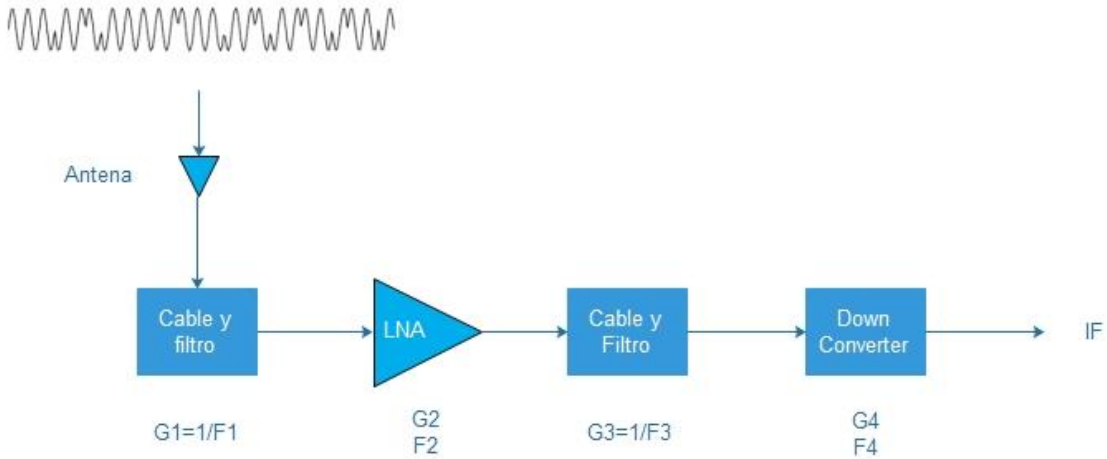

Figura 30. Parte Analógica del FrontEnd

Mientras la señal viaja por el espacio, no se añade ruido, la principal contribución de ruido es debida al propio receptor, especialmente la cadena de elementos de la anterior figura. Vamos a ver como calcular, mediante la fórmula de Friss, la temperatura equivalente del circuito completo y de ahí ¡podremos obtener tanto la densidad de potencia de ruido como la potencia de ruido:

$$T_{eff} = T_A\ (F_1 - 1)T + \frac{(F_2 - 1)T}{G} + \frac{(F_3 - 1)T}{G_1 G_2} + \frac{(F_4 - 1)T}{G_1 G_2 G_3}$$

Donde:

$G_x\ Es\ la\ ganancia\ del\ elemento$
$F_x\ Es\ la\ figura\ de\ ruido\ del\ elemento$
$T\ Es\ la\ temperatura\ ambiente$
$T_A\ Es\ la\ temperatura\ de\ la\ antena$

La temperatura de la antena no representa el valor de temperatura real al a que se encuentra la antena. La antena introduce un ruido en la señal, la TA sería la temperatura que hace falta para que por agitación térmica se genere ese ruido. Con este valor de Teff podemos calcular tanto la densidad de ruido y potencia ruido:

$$Densidad\ de\ Potencia\ de\ ruido = K * T_{eff}\ (\frac{W}{Hz})$$

$$Potencia\ de\ Ruido = K * T_{eff} * Bw\ (W)$$

4.4 Sensibilidad del receptor

La sensibilidad de un receptor puede definirse como el mínimo nivel de potencia que debe tener la señal recibida para la cual el receptor es capaz de devolver la posición. La sensibilidad de un receptor puede expresarse en función de la ratio C/N0 y la figura de ruido F del receptor:

$$Sensibilidad\ = -174dBm + \frac{C}{N_0} + F_{receptor}$$

4.5 Detección de la señal en ruido

Hemos visto que la señal emitida desde el satélite se hace con una potencia de emisión muy baja y que además el propio receptor introduce ruido, con lo cual el proceso de 'distinguir' la señal entre el ruido necesita de un proceso de integración, de manera que el pico de correlación con la copia local sea detectable. Con esta envolvente, la probabilidad de detección de la señal por parte del detector no es del 100% y por lo tanto la probabilidad de falsa alarma, es decir de que el detector asuma la presencia de señal en ausencia de la misma, es distinta de cero.

Básicamente lo que se hace uso de la teoría de detección en la que lo que se hace es discernir si se ha producido un evento (hipótesis H1) o no se ha producido dicho evento (hipótesis H0). Es decir, tenemos una probabilidad P1 de ocurrencia de la hipótesis H1 y una probabilidad P0 de ocurrencia de la hipótesis P0. Si se considera desde el punto de vista del receptor la señal como un conjunto de N muestras r[n] que se agrupan formando el vector r, dicho vector

queda caracterizado por una función de densidad de probabilidad multidimensional $f(r;H_1)$ cuando dichas muestras se generan bajo la hipótesis H1 y $f(r;H_0)$ cuando las muestras se generan bajo la hipótesis H0

Como hemos dicho la señal se ve afectada por el ruido, así pues, será decisión del receptor determinar si la señal ruidosa que detecta se ha generado bajo la hipótesis H1 o H0, en cuyo caso tendremos una detección o una falsa alarma. Como hemos dicho, los problemas de detección se modelan en términos de probabilidades, así pues, lo que se hace es dividir el espacio de observaciones en dos regiones distintas Ω_1 y Ω_0 asociadas a las hipótesis H1 y H0, el receptor establece una separación entre ambas regiones, un valor límite por encima del cual el evento o señal recibida se encuentra por encima o por debajo de esa frontera o límite (threshold, γ) y por tanto en la región Ω_1 o Ω_0.

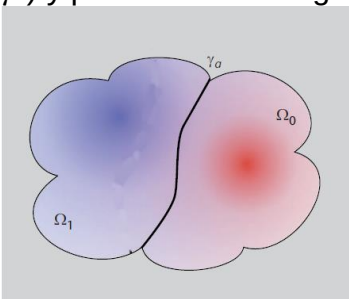

Figura 31. Espacio de Observaciones

Si nos encontramos en la región $\Omega_1$bajo la hipótesis H1 es que se ha producido una detección:

$$P_d = probabilidad\ (decidir\ H_1|H_1) = \int_{\Omega_1} f(r;H_1)dr$$

Si nos encontramos en la región Ω_0 bajo la hipótesis H1 es que hay una falsa alarma:

$$P_{fa} = probabilidad\ (decidir\ H_1|H_0) = \int_{\Omega_1} f(r;H_0)dr$$

Por último, si nos encontramos en la región $\Omega_1$bajo la hipótesis H0 nos encontramos ante una perdida (hay señal, pero el receptor no la detecta):

$$P_p = probabilidad\ (decidir\ H_0|H_1) = \int_{\Omega_0} f(r;H_1)dr$$

Para el caso de GPS el criterio que se usa para decidir la presencia o ausencia del evento es el de Neyman Pearson.

4.5.1 Criterio Neyman Pearson

Lo que se trata es de maximizar la probabilidad de detección Pd, manteniendo las probabilidades de falsa alarma Pfa y de pérdida Pp lo más bajas posibles [*], el problema como veremos es que no es posible minimizar simultáneamente Pfa y Pp, así que lo que se hace es maximizar la probabilidad de detección y minimicen uno de los otros dos errores dejando fijo el otro. En este caso el algoritmo de detección se diseña de modo de maximizar la probabilidad de
detección bajo la restricción de que la probabilidad de falsa alarma no exceda un determinado valor constante. La solución a este problema de optimización

da como resultado la regla de decisión que se conoce como test de relación de verosimilitud:

$$L_{NP}(r) = \frac{f(r;H_1)}{f(r;H_0)} \gtrless_{H0}^{H1} \gamma$$

Donde γ es el umbral y viene determinado por la probabilidad de falsa alarma que tiene un valor constante α

$$P_{fa} = \int_{L(r)>\gamma} f(r;H_0)dr = \alpha$$

$$f(r;H1)$$

4.5.2 Adquisición de la señal como un problema de detección [14][15]

Podemos plantear el proceso de adquisición de la señal GNSS como un problema de detección, donde se trata de comprobar si la señal del satélite está presente en el receptor.

Volvemos sobre el análisis bidimensional de la función de coste que se genera durante los procesos de correlación e integración de la señal.
Vamos a trabajar en un contexto de detección donde la señal llega con muy poca potencia y por tanto se hace necesaria la acumulación de varias integraciones coherentes consecutivas en orden de conseguir que la señal quede presente de manera inequívoca frente al ruido durante el proceso de adquisición.

Vimos que la señal de entrada en cada celda (salida del front-end) la podemos expresar según:

$$\sqrt{C}C_k(t-\tau)D(t-\tau)\cos(2\pi(f_D+f_{IF1})t+(\delta\phi))$$

Si la expresamos en forma polar:

$$r(n) = \sqrt{C}s(n-\tau)\mathrm{e}^{-\mathrm{j}2\pi\mathrm{f_d}n} + w(n)$$

A la salida del correlador coherente para una celda específica tendremos:

$$y(m;\hat{\tau},\hat{f}_d) = \sum_{n=0}^{N_{coh}-1} r(n)c*(n-\hat{\tau})\ e\hat{}(-j2\pi\hat{f}_d n) = A'(\tau_e,f_e)e\hat{}(j\phi(m)) + w'(m)$$

La señal de entrada se divide en m correlaciones coherentes, donde cada correlación coherente contiene Ncoh muestras.τ_e, f_e son los errores residuales en el retardo y la frecuencia Doppler, resultado de restar los valores reales de la señal de entrada con los valores estimados por el receptor.

$$y(m;\hat{\tau}_i,\hat{f}_{d,j}) \sim \begin{cases} \mathcal{N}(A'(\tau_{e,i},f_{e,j}),\sigma_n^2 : H1 \\ \mathcal{N}\ (0,\sigma_n^2) : H0 \end{cases}$$

A la salida del correlador no coherente para una celda específica, lo que tendremos serán las sumas de correlaciones coherentes consecutivas acumuladas de manera no coherente (utilizando el módulo al cuadrado):

$$z(k;\hat{\tau},\hat{f}_d) = \sum_{m=kN_i}^{(k+1)N_i-1} |y(m;\tau,f_d)|^2$$

$$= \sigma_n^2 \sum_{m=kN_i}^{(k+1)N_i-1} \left|\frac{y(m;\tau,f_d)}{\sigma_n}\right|^2 \sim \begin{cases} \mathcal{N}\left(\left(\frac{A'(\tau_{e,i},f_{e,j})}{\sigma_n}\right),1\right) : H1 \\ \mathcal{N}\ (0,1) : H0 \end{cases}$$

Donde Ni es el número de integraciones coherentes.
Si definimos ahora:

$$\xi(k;\hat{\tau},\hat{f}_d) \doteq \left(\frac{z(k;\hat{\tau},\hat{f}_d)}{\sigma_n^2}\right)$$

$$= \sum_{m=kN_i}^{(k+1)N_i-1} \left|\frac{y(m;\tau,f_d)}{\sigma_n}\right|^2 \sim \begin{cases} X_2^2 Ni\left(\left(\frac{A'^2(\tau_{e,i},f_{e,j})}{\sigma_n^2}\right),1\right) : H1 \\ X_2^2 Ni : H0 \end{cases}$$

Donde $\frac{A'^2(\tau_{e,i},f_{e,j})}{\sigma_n^2} = SNR_{coh}$ es la salida del correlador coherente.

Así pues:

$$\xi(k;\hat{\tau},\hat{f}_d) \doteq \left(\frac{z(k;\hat{\tau},\hat{f}_d)}{\sigma_n^2}\right) = \sim \begin{cases} X_2^2 Ni(N_i SNR_{coh}) : H1 \\ X_2^2 Ni : H0 \end{cases}$$

<u>Considerando una sola celda</u> podemos definir la Pfa y la Pd como:

$$P_{fa} \doteq prob\left(\xi(\hat{\tau},\hat{f}_d) > \gamma \,\middle|\, H0\ es\ cierta\right.$$

$$P_d \doteq prob\left(\xi(\hat{\tau},\hat{f}_d) > \gamma \,\middle|\, H1\ es\ cierta\right.$$

Como:
$P_{fa} = prob\left(\xi(\hat{\tau},\hat{f}_d) > \gamma \,\middle|\, Ho\right) = \int_\gamma^\infty f(\xi;H0)d\xi = 1 - \int_{-\infty}^\gamma f(\xi;H0)d\xi = 1 - CDF_{x_{2N_i}^2}(\gamma)$

Para un valor determinado de Pfa podemos obtener el límite $\gamma = Q_{X_{2N_i}^2}^{-1}(P_{fa})$

<u>En el caso de que tengamos múltiples celdas de búsqueda</u>, la probabilidad global de falsa alarma PFA, viene dada de manera aproximada por:

$$PFA\ (\gamma) = N_\tau * N_f * Pfa(\gamma)$$

En el caso de la PD vamos a suponer que nos encontramos con un receptor GNSS cuya estrategia de adquisición es tomar el máximo de todas las celdas, así pues, su probabilidad de detección global vendrá dada por:

$$P_D(\gamma) = prob(\max_{\tau,fd}\{\ (\xi(k;\hat{\tau},\hat{f}_d\ )\} > \gamma\ |H1)$$

Vamos a definir al conjunto de valores de las distintas celdas como:

$$S_n\ = \{\xi(k;\theta_1), \xi(k;\theta_2), \dots\ , \xi(k;\theta_n)\}$$

Donde θ_i es uno de los $n = N_\tau N_f$ posibles pares $(\hat{\tau}, \hat{f}_d)$ dentro del espacio de búsqueda:

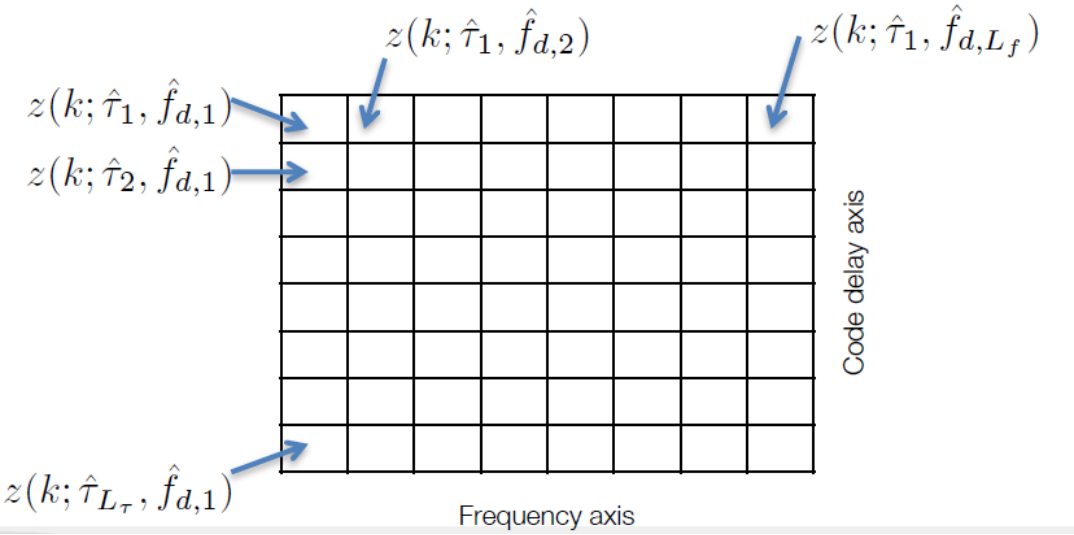

Figura 32. Conjunto de celdas. Espacio de búsqueda de adquisición

Suponiendo que una de las celdas contiene la señal a detectar, vamos a calcular la probabilidad de que el valor dichas celdas, se encuentre por encima del umbral γ :

$$P_D(\gamma) = prob\ \left(ma\,x\left\{S_{N\tau N_{f-1}}|H0, \xi(k;\hat{\tau}_{ok}, \hat{f}_{d_{ok}}\ )|H1\right\} > \gamma\right) =$$
$$prob\ \left(ma\,x\left\{\max\{S_{N\tau N_{f-1}}|H0\}, \xi(k;\hat{\tau}_{ok}, \hat{f}_{d_{ok}}\ )|H1\right\} > \gamma\right)$$

Se presenta pues dos opciones, o bien que el valor máximo se de en la celda bajo la hipótesis H1 o que se de en alguna de las celdas bajo la hipótesis H0:

$$if\ \xi(k;\hat{\tau}_{ok}, \hat{f}_{d_{ok}}\ )|H1 > \max\left\{S_{N\tau N_{f-1}}\middle|H0\right\} \rightarrow P_D(\gamma) = prob\ (\xi(k;\hat{\tau}_{ok}, \hat{f}_{d_{ok}}\ )|H1 > \gamma)$$
$$if\ \xi(k;\hat{\tau}_{ok}, \hat{f}_{d_{ok}}\ )|H1 > \max\left\{S_{N\tau N_{f-1}}\middle|H0\right\} \rightarrow P_D(\gamma) = prob\ (\max\left\{S_{N\tau N_{f-1}}\middle|H0\right\} > \gamma)$$

El primer supuesto es el más razonable, por lo tanto:

$$P_D(\gamma) \approx prob\ (\xi(k;\hat{\tau}_{ok}, \hat{f}_{d_{ok}}\ )|H1 > \gamma) = Q_{x^2_{2N_i}}\ (N_i SNR_{coh})\ (\gamma)$$

5. Probabilidad de detección de la señal GNSS

Como hemos visto en el apartado anterior, la probabilidad de detección de la señal GNSS, pude calcularse con la expresión:

$$P_D(\gamma) \approx prob\left(\xi\left(k;\hat{\tau}_{ok},\hat{f}_{d_{ok}}\right)\middle|H1 > \gamma\right) = Q_{x^2_{2N_i}}(N_i SNR_{coh})\,(\gamma)$$

Haciendo uso de MATLAB, el cálculo de P_D vendría expresado por

$$P_D = 1 - ncx2cdf(\gamma, 2 * Ni, SNR * Ni)$$

A su vez el umbral se puede calcular como:

$$\gamma = Q^{-1}_{X^2_{2NI}}(Pfa)$$

Haciendo uso de MATLAB se podría calcular como:

$$gamma = chi2inv(1 - Pfa, 2 * Ni)$$

Como vimos el valor de gamma se calcula para una Pfa dada, es decir se asume que la señal a la entrada del detector es solo ruido (hipótesis H0). Por el contrario, la PD se define asumiendo que a la entrada tenemos señal más ruido (hipótesis H1). Suponiendo el ruido sigue una distribución gaussiana con media nula, la acumulación con coherente bajo la hipótesis HO da lugar a una distribución Chi-square centrada. Por el contrario, en el caso de que tengamos señal, a la salida de la acumulación coherente tenemos una distribución Chi-square no centrada donde como vemos el parámetro de no centralidad viene dado por Ni*SNR y el número de grados de libertad por 2*Ni.
Así pues, como vemos, el threshold (gamma) depende tanto de la probabilidad de falsa alarma (Pfa) como del número de integraciones no coherentes (Ni).
Por su parte la Probabilidad de Detección depende de gamma como de Ni y de la ratio SNR.

5.1 GUI MATLAB. Cálculo de Probabilidades de Detección

Se ha desarrollado una pequeña GUI en MATLAB en la que se calcula la Probabilidad de Detección en función de distintos parámetros

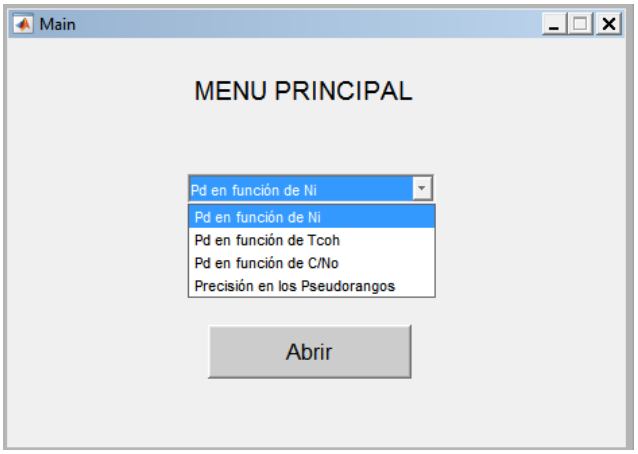

Figura 33. Menu Principal.GUI

5.1.1. Probabilidad de Detección en función del Número de Integraciones no coherentes (Ni):

Se trata por un lado de fijar el límite que separa el espacio de observaciones en las regiones Ω_1 y Ω_0 , es decir fijar gamma, para lo cual se pide al usuario que introduzca la Probabilidad de Falsa Alarma, así como el número de integraciones no coherentes. De esta forma ya podemos calcular el valor de gamma:

```
gamma=chi2inv(1-Pfa,2*Ni)
```

Una vez hallado el valor de gamma, sólo necesitamos fijar los tiempos de integración coherente y el valor de SNR
El usuario introduce el valor del ratio portadora-densidad espectral de ruido (C/No) y el tiempo de integración coherente (Tcoh) lo que nos permite calcular la SNR:

```
SNR_outcohcorr1=2*10^(CNo/10)*Tcoh
```

Una vez calculada la SNR, podemos ver como varía la Pd en función de Ni (representamos hasta un total de 1000 integraciones no coherentes por facilitar la visualización):

```
Ni=linspace (1,1000,1000);

Pd_Ni=1-ncx2cdf(gamma,2*Ni,SNR_outcohcorr1*Ni)
```

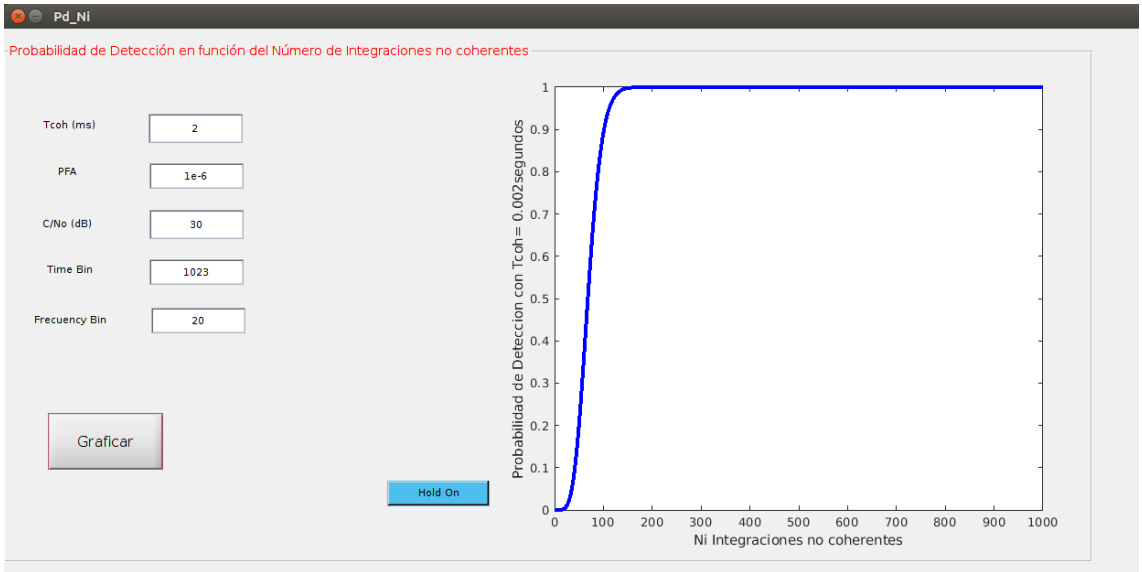

Figura 34. Representación de la Pd en función de Ni

En el ejemplo anterior, se han fijado el tiempo de integración a 2ms, la probabilidad de falsa alarma total1e-6 (nº de celdas $N_\tau x N_f = 1023x20$) y la relación C/No de la señal a 30 dB. Podemos ver como la probabilidad de detección de la señal va aumentando conforme sube el número de integraciones no coherentes.

De igual manera, podemos ir modificando los distintos parámetros que aparecen en el interfaz y comparar varias gráficas para ver cómo afectan los distintos parámetros a la probabilidad de detección (pulsando el botón Hold On):

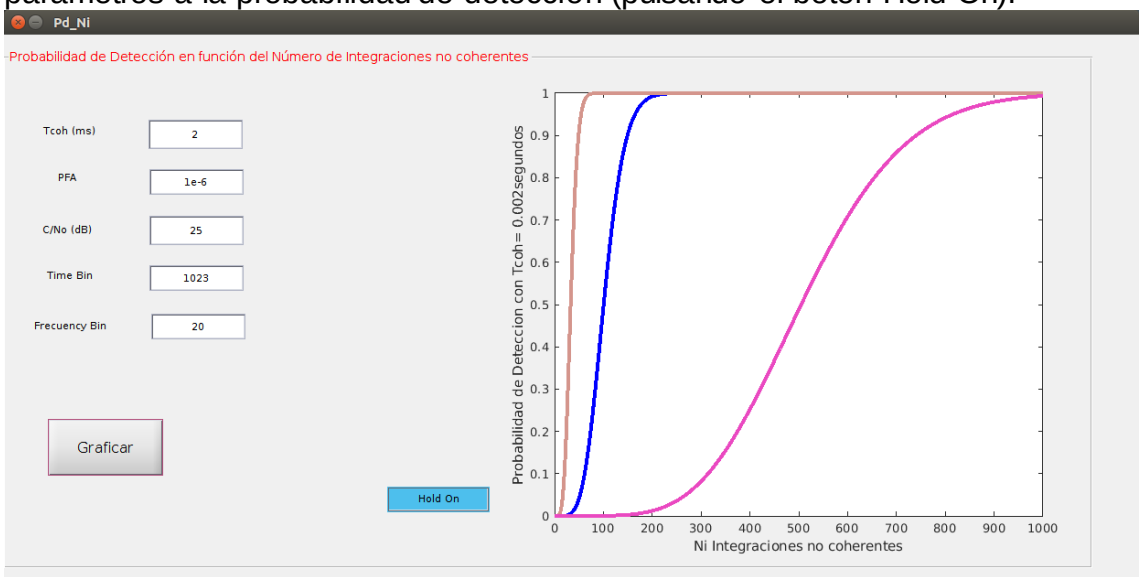

Figura 35. Representación de la Pd en función de Ni para distintos Tcoh

En la figura anterior se ha fijado la probabilidad de falsa alarma total1e-6 (nº de celdas $N_\tau x N_f = 1023x20$) y la relación C/No de la señal a 25 dB y se ha variado el tiempo de integración con Tcoh=2 ms, Tcoh=5 ms y Tcoh=10 ms.

5.1.2. Probabilidad de Detección en función del Tiempo de Integración coherente (Tcoh):

En este caso el proceso es idéntico al caso anterior salvo que vamos a ver cómo crece la Pd en función del incremento del Tcoh.
Se fija de nuevo el límite que determina el valor de señal a la entrada para que el decisor determine que tenemos señal, para lo cual es necesario conocer el valor de Pfa y los distintos valores de Ni:

```
gamma=chi2inv(1-Pfa,2*Ni)
```

Una vez calculada la SNR, en función del valor de C/No de la señal:

```
SNR_outcohcorr1=2*10^(CNo/10)*Tcoh
```

Obtenemos la Probabilidad de Deteccion en función del tiempo de integración coherente (desde 1ms a 100msg) :

```
Tcoh=linspace (10^-3,100*10^-3,100000);
Pd_Tcoh=1-ncx2cdf(gamma,2*Ni,SNR_outcohcorr1*Ni)
```

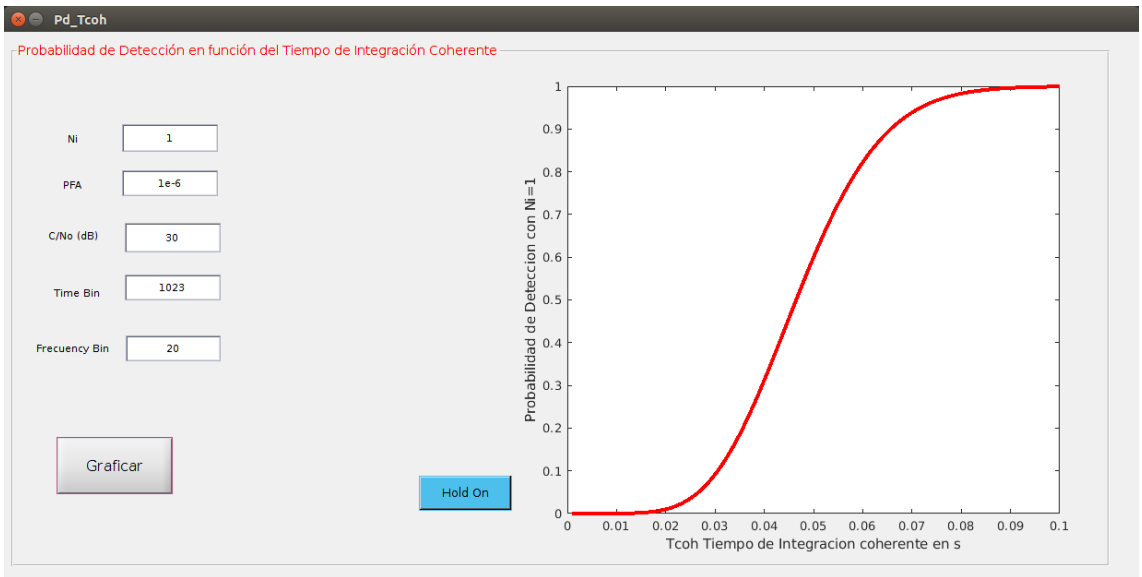

Figura 36. Representación de la Pd en función del Tcoh

Podemos ver en la gráfica anterior que para unos valores dados de Probabilidad de falsa Alarma (Pfa), número de integraciones no coherentes Ni y relación portadora densidad espectral de ruido C/No, la probabilidad de detección de la señal va aumentando conforme aumenta el tiempo de integración coherente Tcoh.

De igual manera, podemos ir modificando los distintos parámetros que aparecen en el interfaz y comparar varias gráficas para ver cómo afectan los distintos parámetros a la probabilidad de detección (pulsando el botón Hold On)::

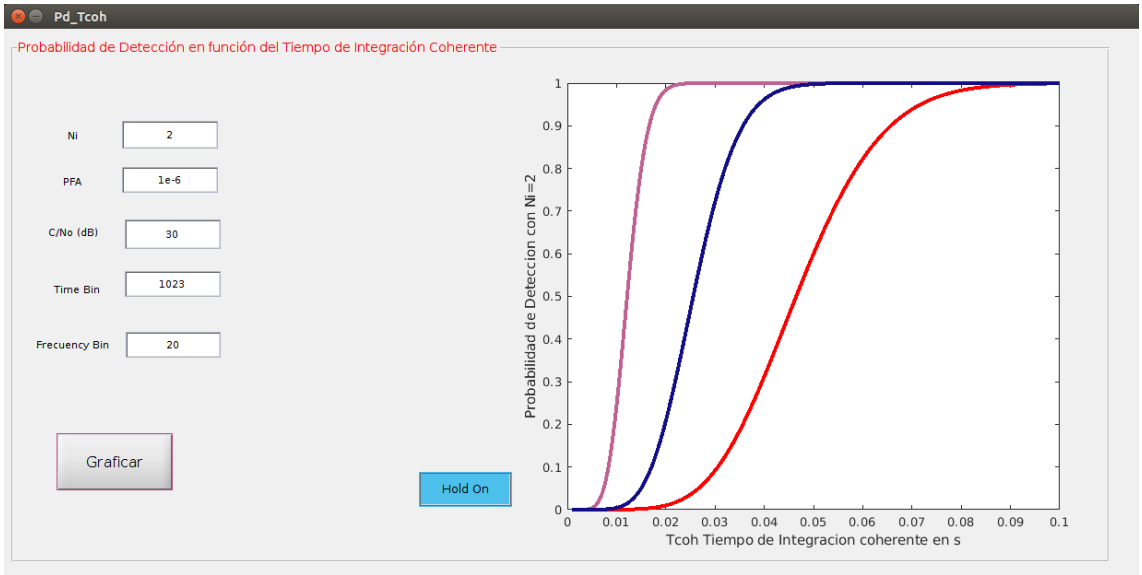

Figura 37. Representación de la Pd en función de Tcoh para distintos Ni

En la figura anterior se ha fijado la probabilidad de falsa alarma total1e-6 (nº de celdas $N_\tau x N_f = 1023x20$) y la relación C/No de la señal a 30 dB y se ha variado el número de integraciones no coherentes con Ni=1, Ni=2 y Ni=5.

5.1.3. Probabilidad de Detección en función de la relación portadora a densidad espectral de ruido de la señal (C/No):

De manera análoga también se puede representar la Probabilidad de Detección en Función de la relación C/No de la señal para unos valores determinados de tiempo de integración Tcoh, probabilidad de falsa alarma y número de integraciones no coherentes, podemos ver como varía la probabilidad de detección de la señal en función de la relación C/No de la misma.

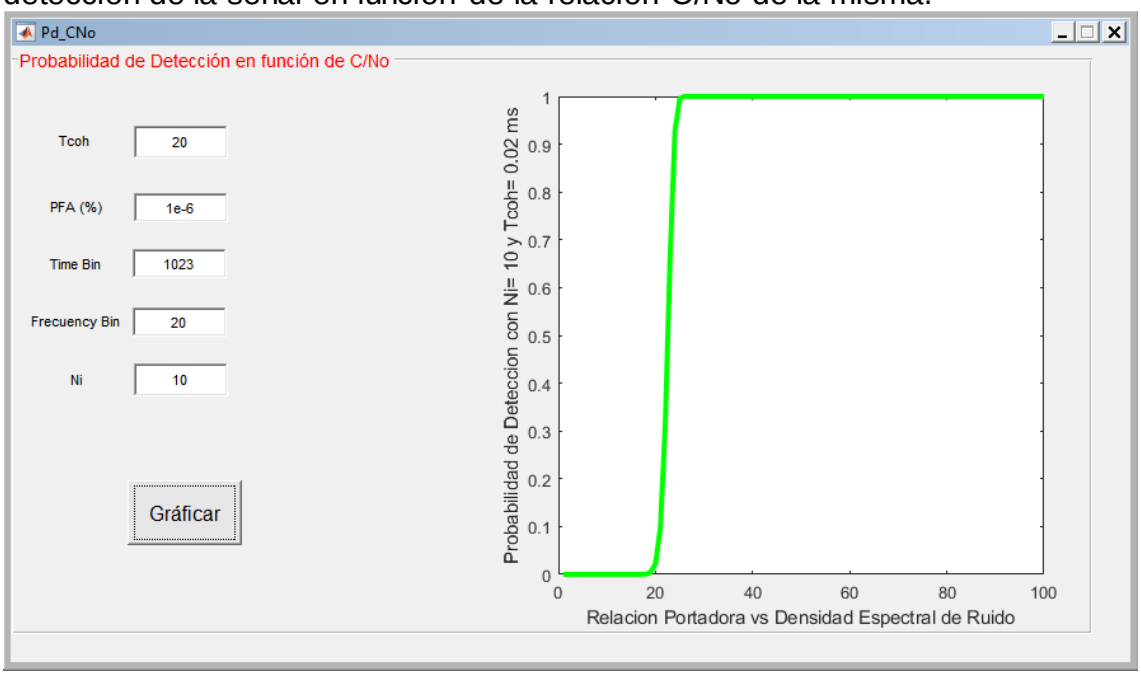

Figura 38. Representación de la Pd en función de C/No

Como puede observarse en la figura anterior, la probabilidad de detección de la señal, para unos valores dados de tiempo de integración coherente Tcoh, probabilidad de falsa alarma global PFA y número de integraciones coherentes Ni, crece con el ratio portadora-densidad espectral del ruido.

5.1.4. Precisión en los pseudorangos

Hemos visto que, en el proceso de adquisición de la señal, se hace necesario estimar la frecuencia Doppler y el retardo de la señal. Este retardo temporal de la señal es el que nos permite determinar la posición del receptor.
En el proceso de adquisición de la señal se usan estimadores cuyo objetivo es que la estimación de este retardo tienda al valor real (hemos de tener en cuenta que este retardo se multiplica por la velocidad de la luz para determinar la posición, por lo que un pequeño error en la estimación del retardo conduce a un error grande en la estimación de la posición). Así pues, se trata de definir un estimador que presente la varianza mínima para garantizar la precisión en la medida del pseudorango.
A tal fin se define la Cota Inferior de Cramer-Rao (CRB), que lo que hace es definirnos el límite en la varianza en la estimación en el retardo de la señal en función del cociente entre energía de la señal y la densidad espectral de ruido (Es/No) y del ancho de banda cuadrático medio de la señal B_{ms}^2:

$$var\ (\tau) \geq \frac{1}{2 * \left(\frac{E_s}{No}\right) B_{ms}^2}$$

Donde:

$$B_{ms}^2 = \frac{\int_{-\infty}^{\infty} (2\pi f)^2 |S(f)|^2 df}{\int_{-\infty}^{\infty} |S(f)|^2 df}$$

Siendo S(f) la transformada de Fourier de la señal s(t)

Si tenemos en cuenta que:

$$\frac{E_s}{No} = \frac{C * Tcoh * Ni}{No}$$

Podemos calcular la cota CRB como:

$$var\ (\tau) \geq \frac{1}{2 * \left(\frac{C}{No} * Tcoh * Ni\right) B_{ms}^2}$$

Una vez obtenida la varianza en el retardo, podemos calcular la precisión en la medida de la posición por:

$$precisión = c * \sqrt{CRB}$$

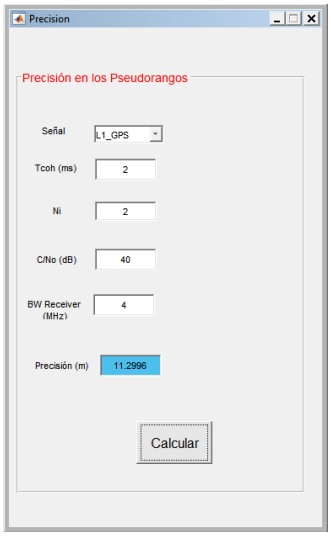

Figura 39. Ejemplo Cálculo de Precisión en el Pseudorango

En la figura anterior podemos ver un ejemplo del cálculo de precisión en el pseudorango para una señal GPS L1, con tiempo de integración Tcoh de 2 ms, 2 integraciones no coherentes Ni, y una señal con C/No de 40 dB, nos da una precisión aproximada de 11 m para un receptor con un ancho de banda de 4MHz (sin contar degradaciones debidas a efectos ionosféricos y troposféricos)

6. Internet of Things (IoT) y GNSS

De acuerdo con el informe de la Agencia Europea GNSS (GSA), en su informe sobre la evolución del mercado GNSS [15] el Internet de las Cosas (IoT) comprende la conexión en red de objetos físicos con una dirección IP única. Un importante avance en el papel de Internet es que el IoT permite que dispositivos físicos, vehículos, edificios y otros objetos se interconecten y/o controlen de forma remota a través de las infraestructuras de red existentes.
Las estimaciones sobre el potencial de crecimiento futuro del mercado mundial de la IoT continúan siendo muy diferentes.

Abrazando una definición más restringida de la IoT y usando estimaciones conservadoras, se espera que el mercado global de IoT sea capaz de generar € 1 trillón en ingresos en 2020. El uso de una definición más amplia produce un pronóstico de hasta € 7 trillones en ingresos para el mismo año.

Independientemente de la definición utilizada, la mayoría de los pronósticos actuales convergen en su evaluación de que los dominios "Fabricación y cadena de suministro" y "Ciudades inteligentes" constituirán los mercados más importantes en términos de ingresos, se prevé que representarán aproximadamente € 2,5 trillones y € 1,3 trillones en 2020, respectivamente. Aunque es probable que genere ingresos compuestos mucho más bajos para el año 2020, los estudios actuales pronostican que el mercado de IoT Healthcare será testigo de la mayor tasa de crecimiento, con una CAGR de 38%

La expansión de IoT en el terreno de la logística (vial, marítima, ferroviaria, aviación) cambiará significativamente la forma en que opera la cadena de suministro, conectando personas, procesos, datos y cosas a través de dispositivos y sensores.
La ventaja clave se refiere a la visibilidad en tránsito, a decir, la provisión de identidad, ubicación y otra información de seguimiento desde el fabricante hasta el minorista. La combinación de datos de sensores en tiempo real con datos ambientales permite una toma de decisiones eficiente que impulsa la productividad general y mueve la cadena de suministro de un modo reactivo a uno proactivo. Los beneficios adicionales incluyen una gestión mejorada de almacenes y flotas y ahorro de combustible.

.

Debemos tener presente, que la precisión en cuanto a la localización de los dispositivos IoT, viene definida por la tecnología empleada para resolver su posición. Podemos utilizar la siguiente generalización para determinar las prestaciones de cada tecnología:

Main absolute positioning technologies and accuracy

	Indoor	Outdoor	Accuracy
Network based	Cell-ID		200-5000m
	Cell Tower Triangulation		50-1000m
Handset based		GNSS	1 - 50m
Hybrid		A-GNSS	
Infrastructure based	Wi-Fi		3-10m /20-50m
	Bluetooth		3-10m
	UWB		20 cm-10 m
	RFID		<3m

Figura 40. Precisión en la localización por tecnología

El mercado de aplicaciones IoT, es muy amplio y como vemos está creciendo de manera muy rápida, por lo que analizar los requerimientos en cuanto a posicionamiento GNSS de los dispositivos IoT para las distintas aplicaciones disponibles, es una tarea inabordable, así pues, se ha decidido segmentar el mercado en los siguientes campos que se consideran los más representativos y con un fuerte impacto en el mercado presente y futuro:

- UAVs (Drone Delivery)
- Asset Tracking (Objects & Peopple)
- Healthcare Monitoring
- Emergency Applications
- Precision Agriculture
- Ubiquity

6.1. Unmanned Aerial Vehicle, UAV (Drone Delivery)

El uso de drones empieza a ser frecuente en muchas actividades comerciales, donde están sustituyendo los medios tradicionales de operación. Según la consultora (PwC, PriceWaterhouseCoopers) [16] una de las 4 grandes consultoras a nivel mundial, estima que las aplicaciones comerciales tienen un mercado total de 127 billones de dólares a nivel mundial. Se estima que la flota de drones industriales en Europa y Estados Unidos será de 50 billones para el año 2050 [17].
Entre los usos comerciales de los drones nos encontramos entre otros con:

- Mapeo 3D: Creación de mapas 3D, topografiar lugares y zonas y toma de fotografías a “vista de pájaro”
- Inspecciones: Se pueden utilizar para la inspección de lugares de difícil acceso, contaminados, análisis aéreo de edificios e infraestructuras, haciendo uso de cámaras termográficas, de sensores etc.
- Transmisión de Datos: Pueden amplificar las señales de red de internet y acceder a lugares remotos u áreas dispersas.
- Seguridad: Se puedes usar para patrullar áreas remotas, alta mar etc

Otro de los principales usos de drones se encuentra en los mercados de distribución de mercancías (Delivery), en dónde se utilizan los drones para la entrega de material y servicios médicos, por ejemplo, entrega de vacunas en países africanos que carecen de infraestructura [18]. Se consideran además, el futuro de la entrega de última milla para bienes de consumo, ya que reduciría el costo por entrega y el tiempo de entrega. Se estima que supondrán un ahorro del 40% en los costos de envío podría traducirse en un aumento del 15-20% en el margen de beneficio y una disminución del precio resultante del 15-20%. [19]. Hay varias grandes compañías que están trabajando en el uso de drones para la entrega de mercancías. Entre las principales nos encontramos con Amazon Prime Air [20] que serán capaces de entregar paquetes de hasta 2 kg, en 30 min, volando a alturas de 120 m, con destinos alejados hasta 16 km. Otro ejemplo sería UPS [21] con un sistema similar de reparto de paquetería en la última milla, donde el dron se lanza desde camiones. Google con su Project Wing [22] es otra de las compañías que está apostando fuerte con el uso de drones para entrega de mercancías.
Entre las principales limitaciones en el uso de drones para entregas de mercancías están entre otras la carga que puede transportar y el consumo de energía, ambas se ven afectadas y afectan al tipo/tamaño de la batería.
A este respecto se puede calcular el consumo en vatios de un dron con un solo rotor [23]:

$$P^* = \frac{T^{\frac{3}{2}}}{\sqrt{\rho\tau}}$$

Donde T sería el empuje:

$$T = (w + m)g$$

Siendo w la carga a transportar (Kg), m el peso de la batería (Kg) y g la gravedad (N), ρ es la densidad del aire Kg/m^3 y τ es el área del disco del rotor en m^2

Para el caso de n rotores, la potencia consumida la podemos determinar por:

$$P = nP^*$$

En las siguientes figuras podemos ver la distancia en Km que puede recorrer el UAV en función de la carga y así como el consumo [24]:

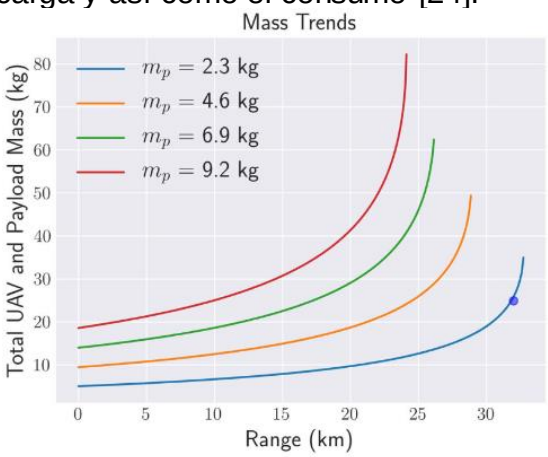

Figura 41. Payload del UAV vs Km recorridos

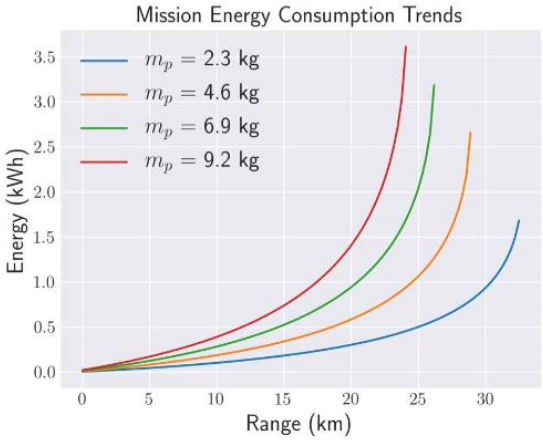

Figura 42. Consumo de Energía UAV vs Km recorridos

En cuanto a la precisión, para el caso de una solución GPS stand alone, la precisión exigida en lo relativo al posicionamiento, se mueve en el rango de 1 m [25], llegando a valores de precisión centimétrica en el caso del uso de métodos como GPS Diferencial (DGPS). Por ejemplo, en el caso de Amazón Prime Air y UPS el espacio para el aterrizaje se trataría de una plataforma de $2m^2$

En cuanto a los startups, que están desarrollando los drones para su uso en la entrega de mercancías, filmación seguridad etc., hay también propuestas muy importantes como Matternet [26] que junto con el uso de furgonetas de reparto ha conseguido en colaboración con la empresa automovilística Mercedes [27] y la empresa de venta online Siroop [28]. Otro ejemplo sería la empresa CyPhy [29] aunque en este caso están más centrados en misiones de vigilancia y apoyo, otros ejemplos similares sería DJI [30] y 3DR [31]

6.2. Seguimiento Activo (Asset Tracking):

Otra de las aplicaciones importantes en el IoT es el seguimiento activo de personas y objetos. En el caso de objetos, no se trata sólo de mercancías, si no que las empresas están empezando hacer uso de herramientas de seguimiento también para la localización de equipos informáticos, herramientas en general, flota de vehículos etc. De esta manera se consigue un importante ahorro en tiempo y dinero. Con el seguimiento activo las compañías obtienen información muy valiosa de quien está usando qué, cuándo y dónde, lo que les permite programar calendarios de mantenimiento y calibración, necesidades de nuevo equipamiento etc. La información obtenida mediante el seguimiento activo les ayuda a tomar mejores decisiones de mercado, previene el uso de pérdidas de material y mercancías, maximiza el tiempo de utilización de los recursos de la empresa, etc.

Hay varias industrias que se pueden beneficiar del uso del seguimiento activo, nombramos a continuación algunas de las más evidentes:

- Construcción: Con el seguimiento de las herramientas y materiales usados en trabajos con múltiples emplazamientos.

- Cuidados de la Salud: Con el seguimiento de la medicamentos, tratamientos y equipos de diagnóstico

- Alquiler: Con el seguimiento de los equipos prestados de manera que se pueda generar al cliente una factura más precisa en función del uso del equipo.

- Seguridad: Podemos hacer el seguimiento de objetos como por ejemplo vehículos y detectar su posición en caso de robo

El mercado de productos disponibles es tremendamente amplio, y su configuración muy variada, desde dispositivos que se alimentan del suministro eléctrico general, a dispositivos que hacen uso de baterías. La duración de las baterías depende, además de su tamaño, de su tasa de refresco de posición, pudiendo configurarse desde real time hasta 1 año en el caso de actualizar su posición 10 o 15 veces al día. Como vemos no hay una autonomía estándar definida para las aplicaciones de asset tracking, así que vamos a tomar un producto tipo, para ver los requerimientos en cuanto a batería y precisión en la medida. En este caso se trata de un módulo GPS para asset tracking de Live View GPS [32]

Flash Trac BAT Hardware:

The Flash Trac BAT features a high accuracy GPS, a 3G cellular module, accelerometer and is ruggedized and weatherproof. Powered by 3 user relplaceable AA Lithium batteries, making field deployement a breeze.

The BAT in its standard configuration will report its location 2x per day, on motion start/stop, and every 10 minutes while in motion. Cellular communication is done on ht ATT 3G cellular network.

Flash Trac Web Based Tracking Portal:

The Flash Trac portal is an intuitive and a very easy to use web based interface, that can be accessed from any computer or mobile device with Internet access. View device locations and device status at anytime - 24/7 from any web enabled computer. Set and receive location based alerts direct to your phone or email.

Call us today for a FREE demo, direct at 1.661.294.6805.

Flash Trac BAT Features:

High Accuracy GPS Tracker
Powered By (4) AA Lithium Batteries
ATT 3G Network
Weatherproof - IP69K, IP68 Rated
Easy and Simple to Deploy
Motion & Movement Detection
Coverage Area: United States

Flash Trac BAT Device Specifications:

GPS Technology	GPS
Receiver	56 channels
Tracking Sensitivity	-162 dBm
Location Accuracy	2.0m CEP
Power	(4) Standard User Replaceable AA Lithium Batteries
Data Transmission	3G HSPDA cellular radio
Sensor	High Sensitivity Accelerometer - 0.01G resolution
Dimensions	5.3" L x 3.5" W x 1.1" H
Weight	8 ounces with batteries
Environmental	IP69K, IP68, SAEJ1455
Operating Temp	-22F to 167F
Mounting	Screw, VHB tape, DIN rail
Certifications	PTCRB, AT&T, Rogers, FCC

Figura 43. Características Tipo. Asset Tracking Module

Vemos como la duración de la batería depende en gran medida del número de reportes diarios que el dispositivo envía, de manera aproximada podemos ver la duración representada en la siguiente gráfica:

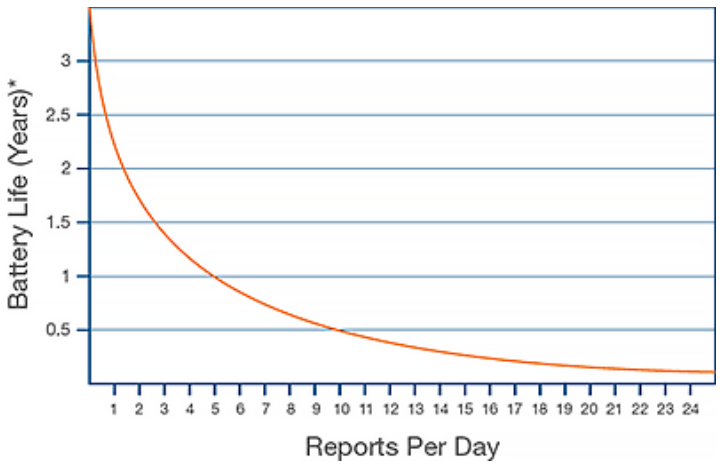

Figura 44. Duración Batería/Reportes diarios

6.3. Cuidado de la Salud (healtcare), GNSS e IoT [33]

Un área donde el GNSS y la tecnología médica trabajan cada vez más juntos es el Internet of Things (IoT) o, en este caso, el 'Internet of Medical Things'.
Existen varios proyectos que haciendo uso del posicionamiento GNSS pueden transmitir información relevante a los servicios médicos. Algunos de los más interesantes serían un dispositivo para personas mayores, Unaliwear [34], que consta de un acelerómetro y un receptor GPS, en el caso de que detecte algo inusual, como podría ser una caída, envía por GPS los datos de localización del usuario. El principal inconveniente es que muchas veces el usuario se encuentra en un recinto interior, por lo que la señal GNSS viene muy atenuada. Dispone también del servicio "Guide me Home" que ayuda al paciente a encontrar el camino a casa. En este mismo sentido hay otras opciones en el mercado con soporte además de para GPS también para Glonass, Beidou, Galileo, como Trax [35], e incluso soluciones discretas dónde el receptor GPS se esconde en la suela del zapato. En general todas estas aplicaciones se basan en posicionamiento GNSS en el exterior (con precisiones de unos pocos metros) y Cell-ID en el interior. Los requerimientos en cuanto a precisión en la medida y vida de la batería se pueden comparar a los de un reloj con GPS. A modo de ejemplo podemos ver a continuación las características de uno de estos dispositivos estándar, se trata del chipset SIRF V, un módulo GPS montado por varios relojes con GPS como Suunto [36], Polar [37] o Tomton [38]:

Feature	Content	Description
Chipset	GSD5e/ROM base	SiRF starV low power single chipset
General	Frequency GPS	L1, 1575.42 MHz
	Frequency GLONASS	L1, 1602~1615 MHz
	C/A code	1.023 MHz chip rate
	Channels	48
	Sensitivity GPS	-164 dB
	Sensitivity GLONASS	-161 dB
Accuracy	Position	<2.5 meters
	Velocity	0.01 meters/second
	Time	1 microsecond synchronized to GPS time
Datum	Default	WGS-84
	Other	Selectable for other Datum
Time to First Fix (TTFF/-122dBm) (Open Sky & Stationary Requirements)	Reacquisition	0.1 sec., average
	Snap start	1 sec., average
	Hot start	1 ~ 2 sec.
	Warm start	9 ~ 15 sec.
	Cold start	25 ~ 35 sec.
Dynamic Conditions	Altitude	18,000 meters (60,000 feet) max.
	Velocity	515 meters/second (1000 knots) max.
	Acceleration	4g, max.
	Jerk	20 meters/second^3, max.
Power	Main power input	1.71 ~ 1.89 VDC input
	Power Consumption	Average: 60mA (Tracking Mode)
	Backup Power(V_RTC)	1.71 ~ 1.89 VDC input
Serial Port	Electrical interface	Default UART
	Protocol messages	NMEA-0183@9600bps

Figura 45. Módulo GPS SIRF V

6.4. Agricultura de Precisión, AP

Podemos definir la agricultura de precisión como la gestión de parcelas agrícolas sobre la base de la observación, la medida y la actuación frente a la variabilidad inter e intra-cultivo. Requiere un conjunto de tecnologías formado por el Sistema Global de Navegación por Satélite (GNSS), sensores e imagen tanto satelital como aerotransportada, junto con Sistemas de Información Geográfica (SIG) para estimar, evaluar y entender dichas variaciones. La información recolectada puede ser empleada para evaluar con mayor precisión la densidad óptima de siembra, estimar la cantidad adecuada de fertilizantes o de otros insumos necesarios, y predecir con más exactitud el rendimiento y la producción de los cultivos [39].

El papel de los Sistemas de Navegación Global por Satélite en la AP es clave, se utilizan tanto para la toma de información en campo, con equipos portátiles de mano, como montados en las máquinas que realizan los trabajos agrícolas (tractor, cosechadora o cualquier tipo de máquina autopropulsada), permitiendo que el usuario y otros dispositivos de control de abordo puedan conocer en cada momento la posición en la que se encuentran dentro de la parcela que trabajan, o en la ruta de transporte hacia/desde ella. La gestión posterior de esta información por parte del usuario y/o los dispositivos de control que la reciben dependerá del objetivo final perseguido.

La precisión de los receptores GNSS en relación a las coordenadas que proporcionan es una de sus características más importantes, debido al antagonismo que existe respecto a su precio. Podemos encontrar en el mercado multitud de modelos con precisiones que van desde 3 metros hasta 3 cm, dependiendo de lo sofisticado de su técnica, el tipo de señal de corrección que utilicen e incluso el estado de la constelación de satélites en la que se basa este sistema. [40]

Podemos ver a continuación una de las posibles clasificaciones existentes en función del desempeño que se espera del dispositivo [41]:

Equipment category	Purpose	Use	Accuracy required
High End	Automated steering assist systems	Planting irrigated cotton	2 cm
Mid range	Visual guidance systems	Controlled traffic farming	10-20 cm or sub-metre
		Sprayer guidance & mapping	
Harvest monitors	Yield monitoring	Broadacre cropping	2.5 - 10 m
		Hand held receivers	

Figura 46. Clasificación AP según propósito [41]

A modo de ejemplo vamos a ver un sensor GNSS (GPS y GLONASS) comercial de Toponcare [42], en el que se detallan los valores de consumo de potencia y precisión en el caso de uso stand alone (sólo GNSS). Se trata del sensor SGR-1, cuyas características son la siguientes:

Storage Temperature	-40° C to +85° C
Voltage	+9 to +28 VDC
Power	2.5 W@12VDC typical
Antenna	L1 GPS/GLONASS with L-band compatibility
RF Input Frequencies	1530 MHz – 1610 MHz
PPS Output	5 ns resolution, ≤30 ns pulse to-pulse precision, LVTTL, configurable polarity and period
Radar Output	variable frequency, simulated ground speed output
Data Input/Output	RS-232, 2 ports up to 460.8 kbps w/o flow control CAN, 1 port, NMEA2000 compliant
SV Tracking Channels	32 GPS L1 C/A, GLONASS L1 C/A code & carrier, SBAS
Acquisition Time (TTFF)	hot <10 sec, warm <35 sec, cold <60 sec reacquisition <1 sec
Data Formats	NMEA 0183 versions 2.1, 2.2, 2.3 and 3.0 output proprietry (TPS) data format , NMEA2000 compliant RTCM SC104 versions 2.1, 2.2, 2.3 and 3.0 input/output geoid & magnetic variation models, grid coordinates
Data Rates	raw measurements and position, up to 20 Hz
ACCURACY	
Position Standlone	Horizontal 2m Vertical 3m
Position Code Differential	(DGPS with external correction) horizontal 0.4m, vertical 0.6m
Position OmniSTAR VBS	Horizontal 0.9m Vertical

Figura 47. Características receptor GNSS Harvesting

6.5. Ubicuidad

La mejor manera que tenemos para definir el término ubicuidad sería “En cualquier lugar, en cualquier momento y a través de cualquier dispositivo”. Posicionamiento y navegación juegan un papel cada vez más importante y dominante en nuestra vida cotidiana permitiendo nuevas aplicaciones en Diferentes campos.
En algunos casos, GNSS ha hecho esto problema trivial. En otros, por ejemplo dentro o cerca de edificios, posicionamiento preciso sigue siendo un considerable desafío técnico.
La navegación ubicua requiere una estimación precisa y oportuna de un posición del usuario en todo momento, en todos entornos y en todos los modos de transporte.

Como es bien sabido, uno de los dispositivos más extendidos en el mundo es el smartphone. En el 2017, el número de usuarios de smartphone en el mundo ascendía a los 4,9 mil millones [43]. Debido a su movilidad, y al gran número de sensores que incluyen, entre ellos receptores GNSS, los smartphones se han convertido en los dispositivos ideales para dar solución a servicios basados en el posicionamiento (LBS, Location Based Services), que se pueden englobar de manera genérica en estas cinco categorías [44]:

- Mapeado y Navegación
- Búsqueda e Información
- Seguimiento (tracking)
- Redes Sociales
- Publicidad móvil

La precisión, así como la frecuencia con la que debe actualizarse la posición, depende por completo de la aplicación y el entorno, pero podemos afirmar que hay cabida para todo tipo de requerimientos debido a la infinidad de aplicaciones y entornos existentes.

6.6. Escenarios de Trabajo

Hemos visto una serie de aplicaciones IoT, cuyo escenario de trabajo podría englobarse dentro de aplicaciones outdoor. Para este tipo de aplicaciones, el valor nominal típico de C/No suele situarse por encima de los 44 dBHz [45]
Podemos ver a continuación, una clasificación de los distintos escenarios indoor de trabajo, así como el valor típico de atenuación y de C/No de la señal recibida.

[TABLE 2] CLASSIFICATION OF TYPICAL INDOOR WORKING SCENARIOS.

SCENARIO	ATTENUATION LOSSES (DB)	RECEIVED C/N_0 (DB-HZ)	EXAMPLE: THE RECEIVED SIGNAL PENETRATES THROUGH...
SOFT-INDOOR	10–20	35–25	... A NEARBY WINDOW.
INDOOR	20–35	25–10	... A REINFORCED CONCRETE WALL.
DEEP-INDOOR	≤ 35	≤ 10	... A REINFORCED CONCRETE ROOF WITH ASPHALT MEMBRANES.

Figura 48. Escenarios trabajo indoor típicos [45]

En cada uno de los siguientes escenarios se va a dar una valor de la precisión que podemos conseguir sin tener en cuenta ningún parámetro que contribuya a la degradación en la medida, como son los efectos troposférico, ionosférico, etc. Por lo tanto los escenarios y medidas proporcionadas en los siguientes subapartados corresponden a la mejor precisión posible

6.6.1 Escenarios Outdoor.

En este escenario podríamos englobar las aplicaciones relacionadas con agricultura de precisión, es decir, vamos a suponer que nos encontramos en una aplicación típicamente outdoor, con los satélites a la vista y sin ningún tipo de obstáculo que pueda propiciar errores e interferencias multicamino.
De igual manera vamos a suponer que nos encontramos en un receptor GNSS-multicelda (1023x20), con un ancho de banda de 4MHz y vamos a suponer que

nos encontramos ante una aplicación Harvest Monitoring que como vimos (figura 43) requiere una precisión típica de 2,5 a 10m.

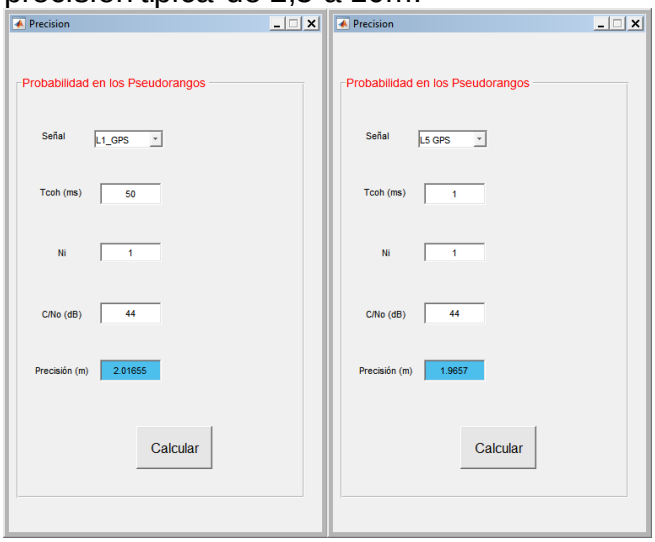

Figura 49. AP Precisión (≈2,5m) GPS L1, GPS L5

En la figura anterior podemos ver, dados un tiempo de integración coherente (Tcoh) de 50 ms, y Ni=1 integraciones coherentes que conseguimos la precisión requerida por la aplicación de Agricultura de Precisión (AP) en torno a 2,5 m, con un receptor con un ancho de banda de 4MHz

Para GPS L1, el tiempo total de señal necesario para conseguir dicha precisión será:

$$T_{TOTAL_{GPSL1}} = T_{coh} * Ni = 50 * 1 = 50\ ms$$

Para GPS L5, el tiempo total de señal necesario para conseguir dicha precisión será:

$$T_{TOTAL_{GPSL5}} = 1 * 1 = 20 * 1 = 1\ ms$$

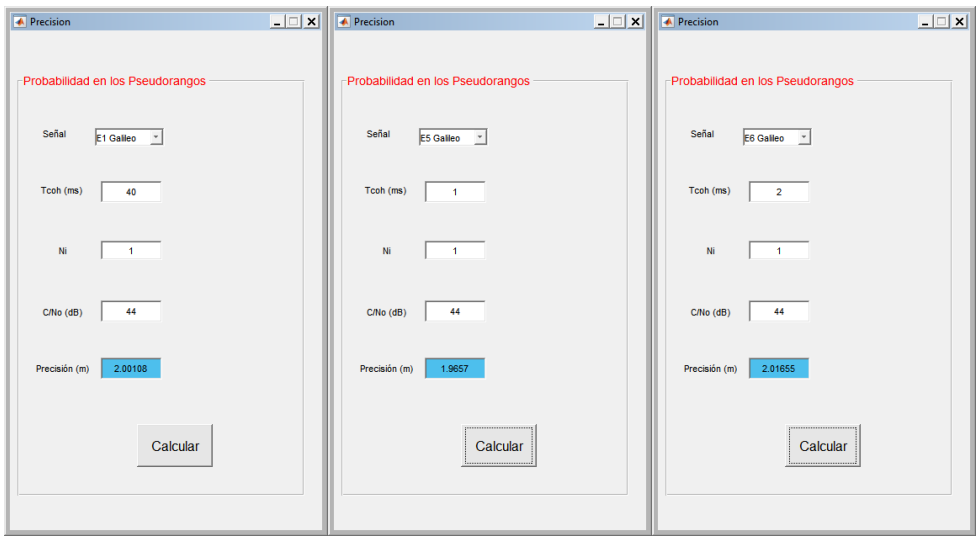

Figura 50. AP Precisión (≈2,5m) Galileo E1, E5 y E6.

Para Galileo E1 el tiempo total de señal necesario para conseguir dicha precisión será:

$$T_{TOTAL_{GalileoE1}} = T_{coh} * Ni = 40 * 1 = 40\ ms$$

Para Galileo E5 el tiempo total de señal necesario para conseguir dicha precisión será:

$$T_{TOTAL_{GalileoE5}} = T_{coh} * Ni = 1 * 1 = 1\ ms$$

Para Galileo E6 el tiempo total de señal necesario para conseguir dicha precisión será:

$$T_{TOTAL_{GalileoE6}} = T_{coh} * Ni = 2 * 1 = 2\ ms$$

6.6.2 Escenarios Outdoor entorno urbano

En este escenario podríamos situar aplicaciones relacionadas con UAV, del tipo drone delivery o posicionamiento de vehículos autónomos y seguimiento de mercancías que se desplazan por entornos urbanos y semi-urbanos.
En este tipo de entorno, sin tener además en cuenta otro tipo de variables como el multicamino, la precisión en el posicionamiento resulta obviamente más complicada, ya que entre otros factores el número de satélites visibles se puede reducir debido a la presencia de obstáculos como edificios, árboles, etc. El valor estimado de la C/No en este tipo de entornos se sitúa en torno a los 20-30 dBHz Utilizando una app de Android llamada GPS Test [46] se ha medido el valor de la señal en un smarthphone Huawei P10, con sensor para GPS / Glonass / Galileo / BDS [47].
Ha de tenerse en cuenta, que la medición se ha hecho en una ciudad pequeña como es Teruel, donde no predominan los edificios altos
Para el caso de GPS, obtenemos los siguientes valores de C/No:

Figura 51. Valor de C/No en entorno urbano. GPS Test

Si tomamos los tres valores más altos y promediamos:

$$\frac{C}{No} = \frac{37 + 27 + 22}{3} \approx 28\ dBHz$$

Tomando ese valor de C/No como referencia y considerando una precisión exigida por la aplicación de 2m, vamos a ver los tiempos de señal necesarios con GPS y Galileo:

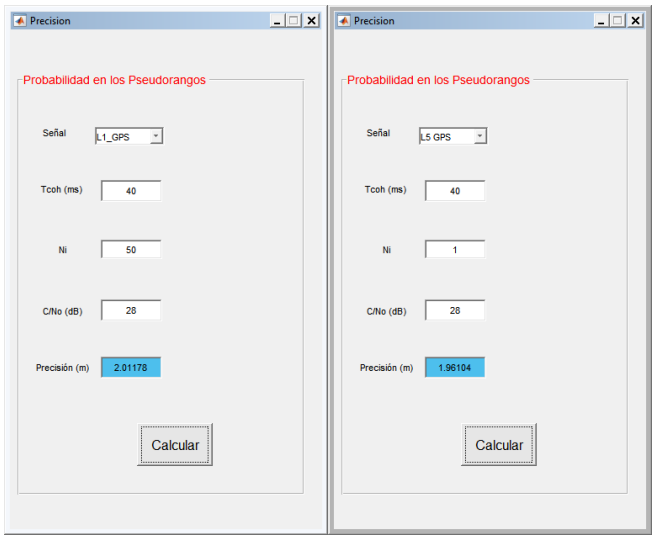

Figura 52. UAV Precisión (≈2m) GPS L1, GPS L5 Tcoh=40ms

Para GPS L1, el tiempo total de señal necesario para conseguir dicha precisión será:

$$T_{TOTAL_{GPSL1}} = T_{coh} * Ni = 40 * 50 = 2\ s$$

Para GPS L5, el tiempo total de señal necesario para conseguir dicha precisión será:

$$T_{TOTAL_{GPSL5}} = T_{coh} * Ni = 40 * 1 = 40\ ms$$

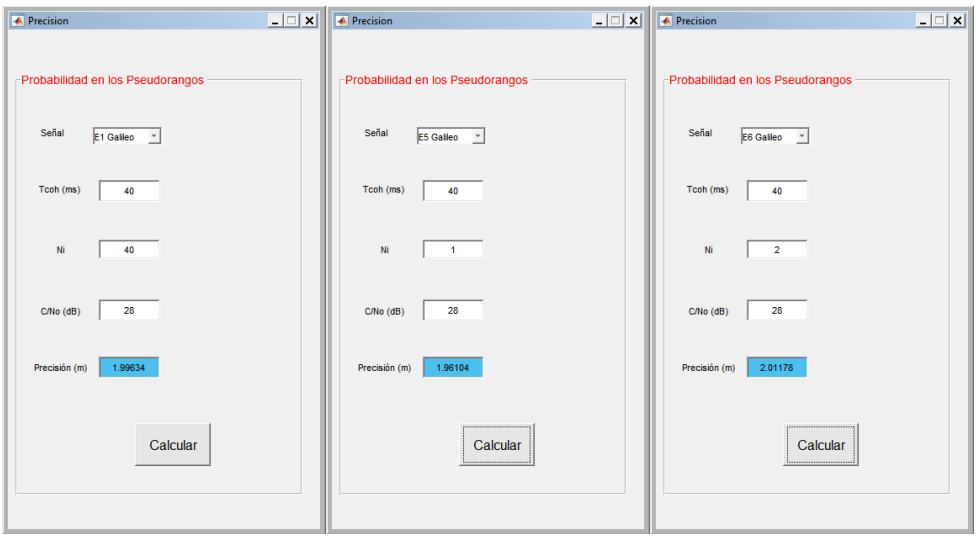

Figura 53. UAV Precisión (≈2m) Galileo E1, E5 y E6 Tcoh=40ms

Para Galileo E1 el tiempo total de señal necesario para conseguir dicha precisión será:

$$T_{TOTAL_{GalileoE1}} = T_{coh} * Ni = 40 * 40 = 1{,}6\ s$$

Para Galileo E5, el tiempo total de señal necesario para conseguir dicha precisión será:

$$T_{TOTAL_{GalileoE5}} = T_{coh} * Ni = 40 * 1 = 40\ ms$$

Para Galileo E6, el tiempo total de señal necesario para conseguir dicha precisión será:

$$T_{TOTAL_{GalileoE6}} = T_{coh} * Ni = 40 * 2 = 80\ ms$$

Puede apreciarse que para aplicaciones que necesiten posicionamiento en tiempo real o casi real una solución GNSS Stand-alone con señales GPSL1 y Galileo E1, cuando nos encontramos en un escenario urbano con edificios y obstáculos (Canyon Enviroments), estas medidas no son los suficientemente precisas si el Tseñal no es suficientemente alto, por lo que se hace necesario utilizar otro tipo de técnicas, por ejemplo receptores multisatélite para tener mayor probabilidad de tener satélites en vista, otra técnica podría ser por ejemplo para determinar la altitud usar la información de altímetros externos, o hacer uso de sensores inerciales. También es posible haciendo uso por ejemplo de un filtro EKF (Extended Kalman Filtering) [48] estimar la posición del vehículo (sin utilizar ningún otro sensor) con sólo 2 satélites a la vista.

7. Procesamiento de la señal GNSS en la nube

7.1. Motivación

Como hemos mencionado a lo largo de este trabajo, cada vez están más extendidas las aplicaciones donde el posicionamiento juega un papel importante. En parte gracias a la explosión del Internet de las Cosas (IoT), de las Ciudades Inteligentes (Smart City) y las aplicaciones M2M (Machine to Machine). Hemos visto también que las señales GNSS, son señales con muy baja potencia y a veces son difíciles de discernir entre el ruido, para lo cual es necesario aumentar los tiempos de integración de la señal o el número de integraciones no coherentes de manera que la señal pueda ser adquirida por el receptor GNSS. Este aumento en los tiempos de integración supone un aumento en las necesidades de computación o procesamiento de la señal, lo que choca de manera frontal con los sensores en aplicaciones IoT, Smart City y M2M, donde la tendencia es hacia la miniaturización y a la reducción del consumo de potencia, lo que impone por tanto una reducción en las capacidades computacionales de los dispositivos. Ante esta problemática, se abre una nueva posibilidad y es que las tareas relacionadas con el procesamiento de la señal se hagan de manera distribuida, en la nube, convirtiendo el receptor en un mero captador de la señal GNSS, lo cual favorece esa tendencia hacia la miniaturización y el bajo consumo en los receptores. Además, este procesamiento en la nube tiene la ventaja, de que, al ser altamente escalable, los recursos computacionales, por tanto, pueden ser ilimitados, señalar también la robustez, y tolerancia a fallos que implica el uso de la nube.

Para analizar las posibilidades del procesamiento de las señales GNSS en la nube. Se ha tomado como referencia el documento Demostration of Cloud GNSS Signal Processing [49]

7.2. Procesamiento en la nube

Podemos definir el procesamiento en la nube [50] como la provisión de servicios IT (servidores, almacenamiento, bases de datos, software, análisis etc.) a través de internet ("la nube"). Existen compañías que ofrecen estos servicios informáticos y se les denomina proveedores de nube y facturan en función del tipo y propiedades de los servicios ofrecidos.
Como hemos visto una de las ventajas que presenta el procesamiento en la nube es la escalabilidad de forma elástica, lo que significa ofrecer la cantidad adecuada de recursos de IT (por ejemplo, mayor o menor capacidad de proceso, almacenamiento, ancho de banda) en el momento justo de acuerdo con la demanda de computación del usuario.

La mayoría de los servicios informáticos en la nube entran dentro de tres categorías generales: infraestructura como servicio (IaaS), plataforma como servicio (PaaS) y software como servicio (SaaS).

Infraestructura como servicio (IaaS)

IaaS, se alquila infraestructura de IT (servidores, máquinas virtuales, almacenamiento, redes, sistemas operativos) a un proveedor de nube y se paga por uso.

Plataforma como servicio (PaaS)

PaaS hace referencia a los servicios informáticos en la nube que suministran un entorno a petición para desarrollar, probar, entregar y administrar aplicaciones. PaaS está diseñado para facilitar a los desarrolladores la creación rápida de aplicaciones web o móviles, sin necesidad de preocuparse por la configuración o administración de la infraestructura de servidores subyacente, el almacenamiento, la red y las bases de datos necesarias para el desarrollo.

Software como servicio (SaaS)

SaaS es un método de entrega de aplicaciones a través de Internet a petición y, normalmente, con una suscripción. Con SaaS, los proveedores de nube hospedan y administran las aplicaciones y la infraestructura subyacente, y se encargan del mantenimiento, como la aplicación de actualizaciones de software y revisiones de seguridad. Los usuarios se conectan a la aplicación a través de Internet, normalmente con un explorador web en su teléfono, tableta o PC.

En el caso del procesamiento de las señales GNSS tomando como infraestructura de cloud el modelo SaaS, esta puede ser a su vez dividida en dos componentes, componente **front-end** que sería la parte visible para el usuario y desde donde se produce la interacción con la nube y el componente **back-end** que sería la parte donde están localizados los recursos IT y se llevan a cabo las tareas de computación de la señal.

En cuanto a proveedores de servicios en la nube, el mercado es suficientemente amplio como para adaptarse a las necesidades específicas de cada cliente. Entre los principales proveedores nos encontramos a Microsoft, con Azure [51] , Amazon con Amazon Web Services (AWS) [52],IBM con IBM Cloud [53], Salesforce.com con Sales Cloud [54], SAP con SAP Cloud Platform [55], Google con Google Cloud Platform [56], etc

Tomando como posible proveedor de SaaS a AWS, podemos observar que ofrece un gran número de servicios: computación, almacenamiento, bases de datos, análisis, networking, herramientas de desarrollo, herramientas de gestión, etc. Vamos a analizar los servicios de computación y almacenamiento ofrecidos por AWS.

Servicios de Computación AWS: Elastic Compute Cloud (EC2)

Amazon EC2 proporciona una amplia selección de tipos de instancias (Máquina Virtual) optimizados para adaptarse a diferentes casos de uso. Los tipos de instancia abarcan varias combinaciones de capacidad de CPU, memoria, almacenamiento y redes [57].

Una comparativa de las distintas instancias puede encontrarse en https://www.ec2instances.info/?region=eu-central-1

De modo general podemos agrupar dichas instancias según su uso:

- Uso General: son instancias de desempeño con ráfagas que proporcionan un nivel base de desempeño de la CPU con la posibilidad de alcanzar ráfagas por encima del nivel básico.
- Optimizadas para Informática: están optimizadas para cargas de trabajo de uso informático intensivo y ofrecen un desempeño alto y rentable con una buena relación de rendimiento informático/bajo precio.
- Optimizadas para memoria: están optimizadas para bases de datos de alto desempeño, bases de datos en memoria y otras aplicaciones empresariales con uso intensivo de memoria
- Información Acelerada: están optimizadas para aplicaciones con uso intensivo de gráficos.
- Optimizadas para Almacenamiento: Presentan características especiales que optimizan el almacenamiento, desde tiempos de latencia reducidos, desempeño de E/S muy altos y procesamiento de lectura secuencial muy alto.

Amazon EC2 está hospedado en varias ubicaciones de todo el mundo. Dichas ubicaciones se componen de regiones y zonas de disponibilidad. Cada *región* es un área geográfica independiente. Cada región tiene varias ubicaciones aisladas conocidas como *zonas de disponibilidad*. Amazon EC2 ofrece la posibilidad de colocar recursos, como instancias y datos, en varias ubicaciones. Los recursos no se replican en las regiones. [58]

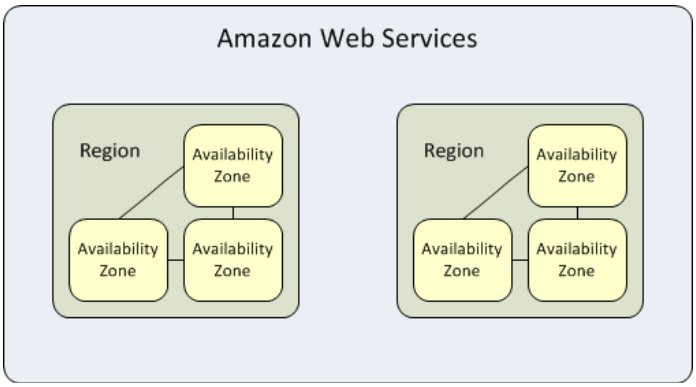

Figura 54. Regiones y Zonas AWS

Para poder lanzar una instancia de Amazon es necesario una Imagen de Máquina de Amazon (AMI) [59].
Una imagen de máquina de Amazon (AMI) proporciona la información necesaria para lanzar una instancia, que es un servidor virtual en la nube. Se debe especificar una AMI de origen al lanzar una instancia. Cuando se necesitan varias instancias con la misma configuración, se pueden lanzar desde una misma AMI. Cuando necesite instancias con distintas configuraciones, se pueden usar distintas AMI para lanzarlas.

Una AMI incluye lo siguiente:

- Una plantilla para el volumen raíz de la instancia (por ejemplo, un sistema operativo, un servidor de aplicaciones y aplicaciones)
- Permisos de lanzamiento que controlan qué cuentas de AWS pueden utilizar la AMI para lanzar instancias
- Un mapeo de dispositivos de bloques que especifica los volúmenes que se van a adjuntar a la instancia cuando se lance

En el siguiente diagrama, se resume el ciclo de vida de la AMI. Después de crear y registrar una AMI, esta se puede utilizar para lanzar nuevas instancias. Además, se puede copiar una AMI dentro de la misma región o en regiones diferentes. Cuando ya no necesite una AMI, se puede cancelar su registro.

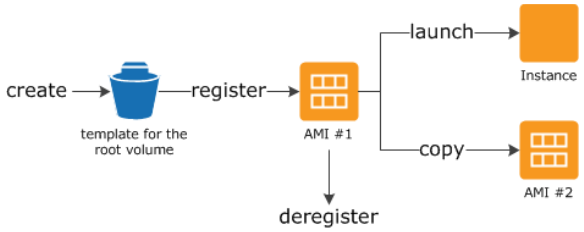

Figura 55. Ciclo de vida de una AMI

Servicios de Almacenamiento EC2

Amazon EC2 ofrece distintas opciones de almacenamiento de datos flexibles para sus instancias. Cada opción presenta una combinación exclusiva de desempeño y durabilidad. Las opciones de almacenamiento se pueden utilizar de forma independiente o combinadas para ajustarse a sus requisitos. Entre las opciones de almacenamiento se encuentran las siguientes:

Amazon Elastic Block Storage (EBS)

Amazon EBS ofrece volúmenes de almacenamiento duradero de nivel de bloque que se pueden adjuntar a una instancia en ejecución. Se puede utilizar Amazon EBS como dispositivo de almacenamiento principal para los datos que requieran actualizaciones frecuentes y detalladas. Por ejemplo, Amazon EBS es la opción de almacenamiento recomendada al ejecutar una base de datos en una instancia.

Un volumen de EBS se comporta como un dispositivo externo, sin formatear y sin procesar que puede adjuntar a una única instancia. El volumen persiste, independientemente de la vida de ejecución de una instancia. Después de adjuntar un volumen de EBS a una instancia, se puede usar como cualquier otro disco duro físico

Para mantener un backup de los datos, se puede crear una *instantánea* de un volumen de EBS, que se almacena en Amazon S3. Se puede crear un volumen de EBS a partir de una instantánea y adjuntarlo a otra instancia.

Almacén de instancias Amazon EC2

Muchas instancias pueden obtener acceso al almacenamiento desde discos adjuntos físicamente al equipo host. Este almacenamiento de disco se denomina *almacén de instancias*. El almacén de instancias ofrece un almacenamiento de nivel de bloques temporal para las instancias. Los datos en un volumen de almacén de instancias persisten sólo durante la vida de la instancia asociada; si detiene o termina una instancia, cualquier dato en los volúmenes del almacén de instancias se pierde.

Sistema de archivos de Amazon Elastic File System (EFS)

Amazon EFS proporciona almacenamiento de archivos escalable para utilizar con Amazon EC2. Se puede crear un sistema de archivos de EFS y configurar las instancias para montar el sistema de archivos. Se puede usar un sistema de archivos de EFS como un origen de datos común para aplicaciones y cargas de trabajo que se ejecutan en varias instancias.

Amazon Simple Storage Service (S3)

Amazon S3 ofrece acceso a una infraestructura de almacenamiento de datos que se ha diseñado para facilitar la computación escalable basada en web, al hacer posible que almacene y recupere cualquier cantidad de datos, en cualquier momento, desde Amazon EC2 o desde cualquier lugar de la web. Por ejemplo, se puede utilizar Amazon S3 para almacenar backups de datos y aplicaciones. En el contexto del procesamiento de señales GNSS en la nube, S3 se puede usar, por ejemplo, para almacenar los datos raw GNSS a procesar

En la siguiente figura se muestra la relación entre los distintos tipos de almacenamiento:

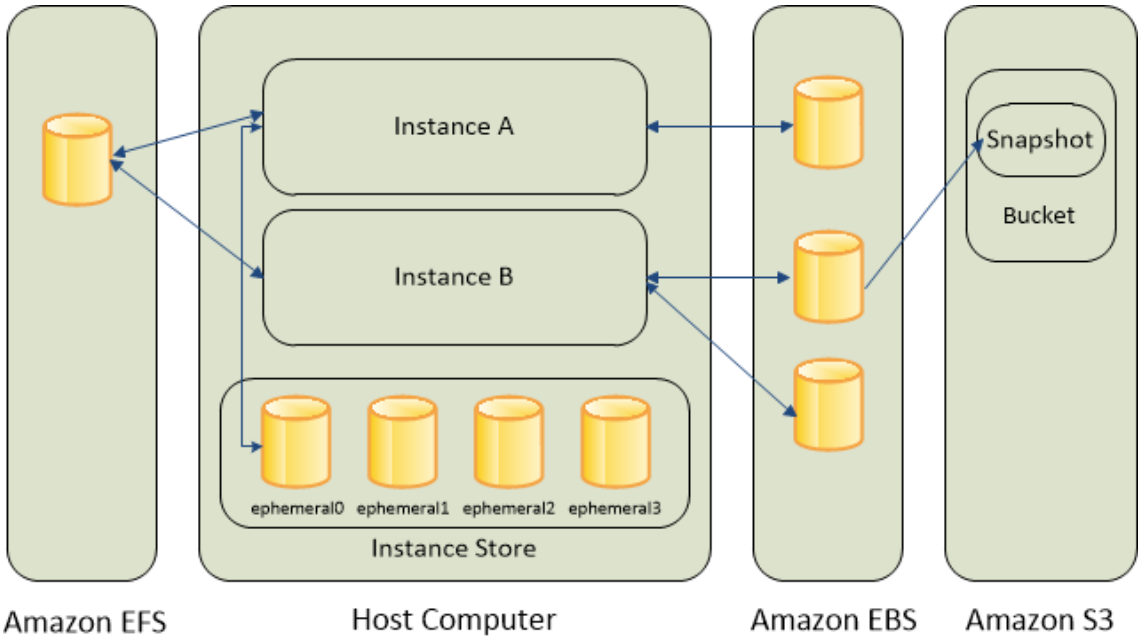

Figura 56. Tipos de Almacenamiento AWS

AWS Simple Queue Service (SQS)

SQS es un servicio de colas de mensajes que facilita el desacople y la coordinación los componentes de una aplicación en la nube, por ejemplo, entre el front-end y el back-end.
Una cola de mensajes es una forma de comunicación asíncrona de servicio a servicio que se usa en arquitecturas de microservicios y sin servidor. Los mensajes se almacenan en la cola hasta que se procesan y eliminan. Cada mensaje se procesa una vez sola, por un solo consumidor.

En la arquitectura de la nube moderna, las aplicaciones se desacoplan en bloques pequeños e independientes que son más fáciles de desarrollar, implementar y mantener. Las colas de mensajes proporcionan la comunicación y la coordinación para estas aplicaciones distribuidas. Las colas de mensajes pueden simplificar de forma significativa la escritura de código para aplicaciones desacopladas y, a la vez, mejorar el rendimiento, la fiabilidad y la escalabilidad.

Las colas de mensajes permiten a diferentes partes de un sistema comunicarse y procesar las operaciones de forma asíncrona. Una cola de mensajes ofrece un búfer ligero que almacena temporalmente los mensajes, y puntos de enlace que permiten a los componentes de software conectarse a la cola para enviar y recibir mensajes [60]

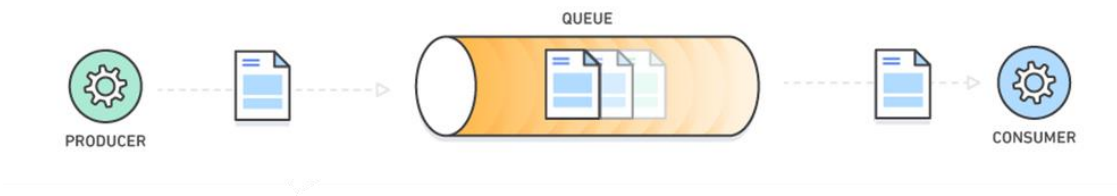

Figura 57. AWS: Simple Queue Service

En el ámbito del procesamiento de señales GNSS en la nube SQS se utiliza como elemento de comunicación entre la interface front-end (servidor web) con el back-end (el núcleo del sensor GNSS en la nube, donde se realiza el procesamiento de la señal GNSS)

7.3. Arquitectura de un receptor GNSS basado en la nube

Vamos a describir la arquitectura de un receptor experimental [61] que ha sido testeado de manera exitosa con señales GNSS reales y diferentes cargas de trabajo. El núcleo del este sensor está basado en el en el receptor software de alta sensibilidad HS-GNSS desarrollado en el proyecto DINGPOS.
De manera general podemos decir que este receptor basado en la nube está compuesto por tres elementos, el terminal de usuario, el cloud front-end y el cloud back-end

- **Terminal de usuario:** Es el elemento hardware que capta la señal GNSS y las envía a la nube en crudo para su procesado posterior. Está compuesto por un front-end RF sintonizado a las bandas GNSS de interés, un data grabber que digitaliza las señales GNSS y un módulo de comunicación con la nube. Una vez que las muestras de la señal GNSS han sido procesadas en la nube, se envía un informe PVT (Position Velocity and Time) y un análisis del nivel de la señal vía email a la dirección indicada por el usuario
- **Cloud front-end:** Es la interface entre el usuario y el procesamiento en la nube. La comunicación con la nube se hace a través de un servicio web HTTP que corre en una instancia EC2, a la que los usuarios se pueden logear. El usuario accede a una interface desde la cual puede configurar una serie de parámetros y lanzar una nueva ejecución para que el back-end procese la señal con los parámetros introducidos en la interface. La información a rellenar por el usuario en la interface es la siguiente:
    - Muestras de la señal GNSS: El archivo con las muestras de la señal GNSS en crudo, que puede ser subido a la nube desde un archivo local del propio dispositivo de usuario o bien se puede descargar automáticamente introduciendo directamente una URL donde se encuentre el archivo. El usuario debe especificar también el formato del archivo, la frecuencia de muestreo, la frecuencia central, la frecuencia central, el valor de cuantificación etc.
    - Ajustes HS-GNSS: El usuario debe configurar el receptor HS-GNSS de acuerdo con las necesidades del análisis a ejecutar y las condiciones de trabajo en las cuales se han obtenido las muestras de la señal GNSS. Los parámetros a proporcionar serían del tipo banda de la señal GNSS a procesar, tiempo de integración, etc.
    - Información de asistencia: Como por ejemplo información sobre la lista de satélites a buscar, con lo que se puede evitar hacer la búsqueda completa de todos los satélite con el consiguiente ahorro de tiempo. También se puede proporcionar información aproximada sobre la frecuencia Doppler de los satélites visibles. En el caso de que se vaya a computar un análisis PVT un archivo RINEX (Receiver Independent Exchange Format) debe ser adjuntado o bien desde el receptor del usuario o bien automáticamente el receptor en la nube puede descargarlo desde una lista de servidores GDC disponibles.
- **Cloud back-end:** Es el lugar donde el conjunto de tareas de procesamiento se llevan a cabo. Esto incluye el procesamiento de las señales muestras en crudo de la señal GNSS y también generar los informes de salida. Cada uno de los nuevos trabajos generados en la interface de usuario es recibido y gestionado por un administrador de recursos a cargo de asignar los recursos EC2 en función de la carga de trabajo actual. Cuando se lanza una nueva instancia, una imagen pre-configurada guardada en el servicio EBS se inicia con todo el software necesario para el procesamiento del archivo de muestras GNSS en crudo. Por lo tanto, las nuevas instancias están listas para funcionar justo después de ser lanzadas con una latencia muy pequeña. El receptor del software HS-GNSS se implementa como una aplicación independiente mediante MATLAB Compiler Runtime (MCR). Se trata de un conjunto

independiente y gratuito de bibliotecas compartidas que permite la ejecución de aplicaciones compiladas de MATLAB o componentes en computadoras que no tienen instalado MATLAB

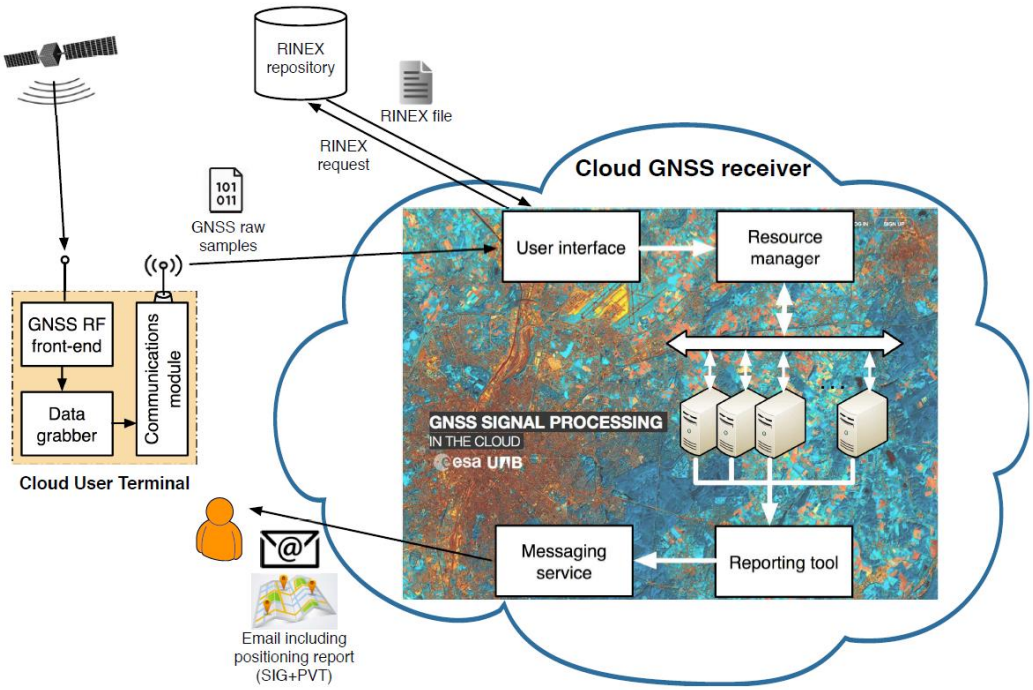

Figura 58. Arquitectura de un receptor cloud GNSS [61]

8. Soluciones comerciales para cloud GNSS

En este punto vamos a tratar de ofrecer una pequeña muestra de las soluciones comerciales disponibles. En una primera aproximación, vamos a ver una serie de receptores habitualmente usados en GNSS-SDR (GNSS, Software Defined Radio) [62] pero que con la configuración adecuada podemos utilizarlos para la captura de la señal en crudo GNSS y su posterior envío a la nube para el procesamiento:

A modo de síntesis, las funciones que debe hacer este receptor RF serán:

- Recibir la señal GNSS
- Acondicionamiento de la señal (Amplificar, bajar a frecuencia intermedia FI, …etc.)
- Digitalización señal (samples en fase y cuadratura)
- Transferir la señal digitalizada

Los cabezales RF que vamos a ver son los siguientes:

Front end	Interface	Frequency range	Bandwidth	ADC resolution	IC	Mode
ADALM-Pluto	USB 2.0	325 MHz - 3.8 GHz	61.44 MHz	12 bit I/Q	Analog Devices AD9363	Full duplex
Airspy Mini	USB 2.0	24 MHz - 1.8 GHz	6 MHz	12 bit I/Q	Rafael Micro R820T2	Receive only
BladeRF	USB 3.0	300 MHz - 3.8 GHz	40 MHz	12 bit I/Q	LimeMicro LMS6002D	Full duplex
FreeSRP	USB 3.0	70 MHz - 6 GHz	61.44 MHz	12 bit I/Q	Analog Devices AD9364	Full duplex
HackRF	USB 2.0	1 MHz - 6 GHz	20 MHz	8 bit I/Q	Maxim MAX2837 / 5864	Half duplex
LimeSDR	USB 3.0/PCIe	100 kHz – 3.8 GHz	61.44 MHz	12 bit I/Q	Lime Micro LMS7002M	Full duplex
RTL-SDR	USB 2.0	24 MHz - 1.766 GHz	2.4 MHz	8 bit I/Q	Realtek RTL2832U	Receive only
SDRplay	USB 2.0	1 kHz - 2 GHz	6 (10) MHz	14 (8) bit I/Q	Mirics Msi001	Receive only
Sidekiq	Mini PCIe	70 MHz - 6 GHz	50 MHz	12 bit IQ	Analog Devices AD9361*	Full duplex
SiGe GN3S	USB 2.0	1.57542 GHz	8 MHz	4 bit I/Q	SiGe SE4110L	Receive only
USRP N210	Gigabit Ethernet	0 - 6 GHz	25 MHz (50 MHz)	14 (8) bit IQ	Analog Devices AD9361	Full duplex
USRP X310	Gigabit Ethernet	0 - 6 GHz	200 MHz	14 bit IQ	Analog Devices AD9361	Full duplex
XTRX	Mini PCIe	100 kHz – 3.8 GHz	160 MHz	14 bit IQ	Lime Micro LMS7002M	Full duplex

Figura 59. Receptores GNSS-SDR [63]

8.1 ADALM-Pluto [64]

Se trata de un dispositivo pensado inicialmente para labores de aprendizaje en los fundamentos de SDR. Ofrece un canal de recepción y un canal de transmisión que pueden operarse en dúplex completo, capaces de generar o medir señales analógicas de RF desde 325 hasta 3800 MHz, hasta 61,44 mega muestras por segundo (MSPS) con un ancho de banda de 20 MHz. El PlutoSDR es completamente autónomo, tiene un tamaño muy reducido, y está completamente alimentado por USB

Figura 60. Transceptor ADALM-Pluto[64]

Está basado en el transceptor AD9363 [65], podemos ver el diagrama de bloques en la siguiente figura:

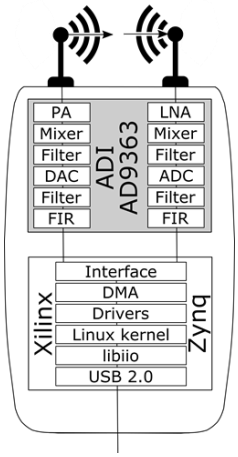

Figura 61. Diagrama de bloques ADALM-Pluto [64]

8.2 Airspy Mini [66]

Se trata de otro cabezal RF que nos permite recibir las muestras GNSS en crudo para digitalizarlas y enviarlas a la nube, con una tasa de muestreo de 3 Msps.

Figura 62. AirSpy Mini [66]

En este caso el cabezal podríamos, por ejemplo, montarlo junto con una raspberry pi 3 [67] como Dongle USB, de manera que haciendo uso del módulo WiFi pudiéramos enviar las muestras digitalizadas a la nube.

8.3 Blade RF Nuand [68]

Se trata de otra plataforma SDR utilizada en el ámbito de la educación pero que también podría utilizarse para el procesamiento de las señales GNSS en la nube

Figura 63. Blade RF [68]

Blade RF permite también conexión con Raspberry pi 3 [67], lo que nos permite usar la conectividad wifi de la misma para enviar los samples digitalizados a la nube. La velocidad de sampleo de la señal es hasta 40 Msps y una resolución del ADC de 12 bits

8.4 FreeSRP [69]

Otro posible tranceptor RF que podría ser utilizado para digitalizar señales GNSS, sería el proporcionado por el proyecto FreeSRP:

Figura 64. FreeSRP [69]

FreeSRP se integra dentro de la familia de proyectos MyriadRF [70] que es una plataforma de proyectos hardware y software abiertos para comunicaciones sin hilos. FreeSRP ofrece un dispositivo que permite la captura de señales RF en el rango de frecuencias que va desde los 70 MHz a los 6 GHz con una frecuencia de muestreo de 61.44 Megasamples por segundo y una resolución del conversor AD de 12 bits

8.5 HackRF One [71]

Otro transceptor disponible en el mercado y utilizado para SDR es HackRF One que es capaz de recibir y transmitir señales en un amplio rango de frecuencias que van desde 1 MHz a 6 GHz. Tiene una tasa de muestreo de 20 Megasamples por segundo y un conversor A/D con una resolución de 8 bits

Figura 65. HackRF One [71]

8.6 LimeSDR [72]

Se trata de un transceptor SDR de bajo coste capaz de trabajar con señales RF en un rango de frecuencias de 100 KHz a 3,8 KHz y un conversor A/D de 12 bits de resolución de 61,44 Megasamples por segundo (limitado por la tasa de transferencia del USB 3.0)

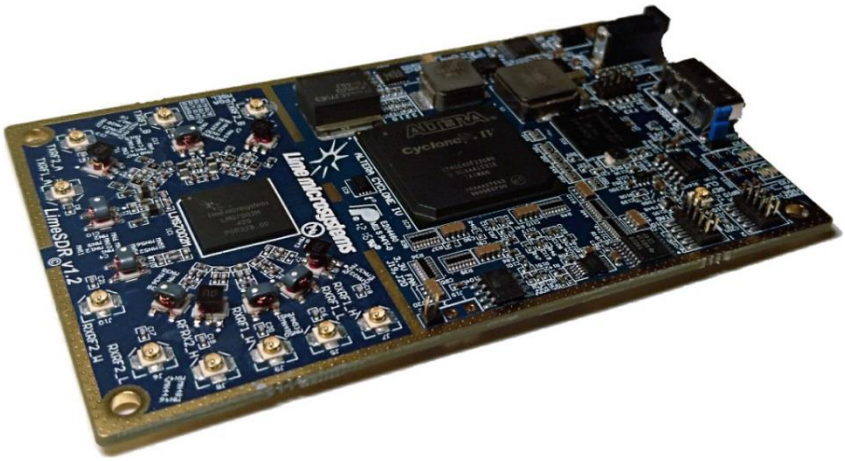

Figura 66. LimeSDR [72]

8.7 RTL-SDR [73]

En este caso se trata de un Dongle USB, capaz de capturar señales en un rango de frecuencias que va desde 500 KHz a los 1,75 GHz y una tasa de muestreo máxima de 3.2 Megasamples por segundo. Cuenta además con un conversor A/D con 8 bits de resolución

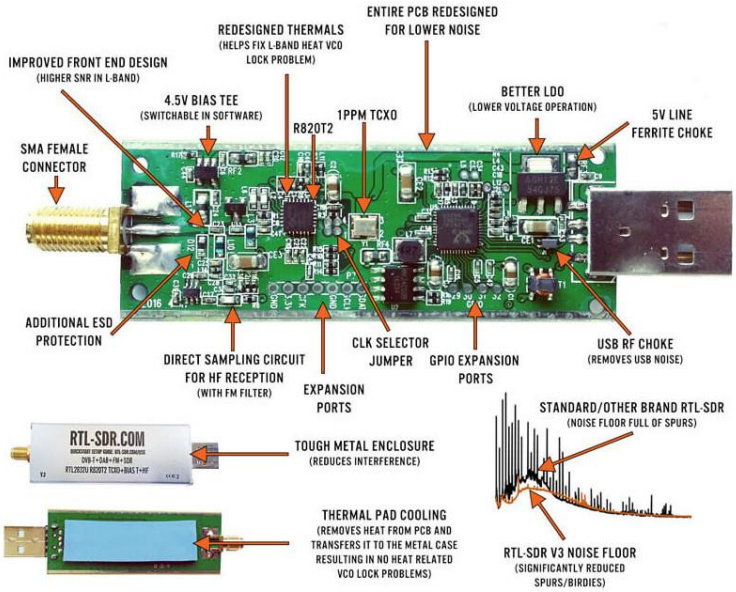

Figura 67. RTL-SDR [73]

8.8 SDRplay (RSP1A) [74]

Se trata de un receptor RF capaz de trabajar con un amplio rango de frecuencias que va desde 1KHz a los 2GHz. Cuenta además con un conversor A/D con 14 bits de resolución y un ancho de banda de 10 MHz

Figura 68. RSP1A [74]

8.9 Sideqik [75]

Es un transceptor capaz de trabajar con frecuencias que van desde los 70 MHz a los 6 GHz, con un ancho de banda de 50 MHz con una tasa de muestreo de hasta 61.44 Megasamples por segundo, cuenta además con un conversor A/D de 12 bits, y cuenta con un interfaz PCIe y otro USB 2.0

Figura 69. Sideqik [75]

8.10 SiGe GN3S Sampler v3 [76]

En este caso tenemos un dispositivo desarrollado por la universidad de Colorado capaz de capturar señales GNSS crudas en un archivo binario, igual que en los dispositivos anteriores, es necesario el uso de una antena para capturar las señales GNSS y la transferencia del archivo se hace a través de una interfaz USB.

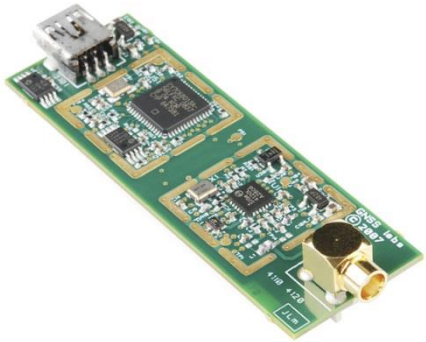

Figura 70. SiGe GN3S Sampler V3 [76]

Una de sus principales características es que presenta un consumo de energía muy bajo. Cuenta además con un filtro IF integrado y un conversor A/D de 2 bits de resolución

8.11 Ettus USRP N210 [77]

Este dispositivo USRP (Universal Software Radio Peripheral) permite captura señales RF desde continua a 6 GHZ. Cuenta con un conversor A/D con una resolución de 14 bits y un ancho de banda de 100 MHz. La tasa de muestreo es de hasta 100 Megasamples por segundo y cuenta con una interfaz ethernet con un troughput de hasta 50 Megasamples por segundo.

Figura 71 Ettus USRP N210 [77]

8.12 Ettus USRP X310 [78]

Muy similar al dispositivo anterior, se trata de otro dispositivo USRP de Ettus pero con un ancho de banda de 160 MHz. Permite trabajar con señales desde DC a 6 GHz. Presenta varias interfaces ethernet, PCIe, ExpressCard.

Figura 72. Ettus USRP X310 [78]

8.13 XTRX [79]

Se trata de otro dispositivo SDR económico y de reducidas dimensiones. En este caso sólo cuenta con interfaz PCIe y es capaz de trabajar con señales RF en el rango de los 100 KHz a los 3.8 GHz. Presenta una frecuencia de sampleo de 160 MSps y un CAD con una resolución de 12 bits,

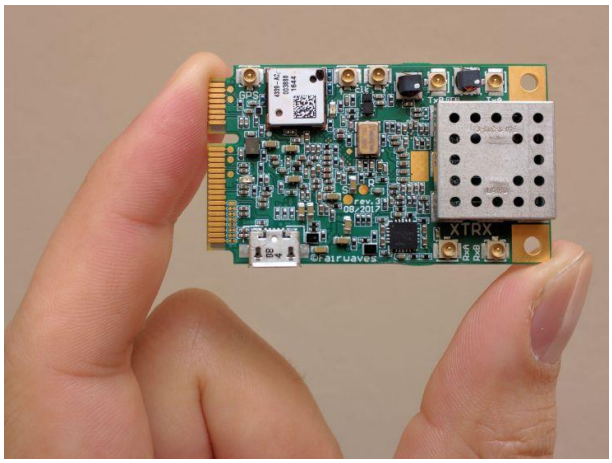

Figura 73. XTRX [79]

9. Consumos de Energía

Para calcular los consumos de energía, primero hemos de calcular el tamaño de la muestra que se envía a la nube. Ese tamaño se calcula según [49]:

$$Chunk_{size} = T_{signal} \cdot Fs \cdot Quantificación$$

Tomando esta ecuación, se muestran a continuación el tamaño de los chunk a enviar a la nube, en función de la Fs de los dispositivos analizados en el punto *8. Soluciones comerciales para cloud GNSS* de este TFM. Debe tenerse en cuenta que el ancho de banda de las señales GPS y Galileo es:

- GPS L1= 20.46 MHz
- GPS L5= 20.46 MHz
- Galileo E1= 24.552 MHz
- Galileo E5= 51.150 MHz
- Galileo E6= 40.920 MHz

Por lo tanto, en aquellos casos en los que el BW del dispositivo está por encima del BW de la señal se ha asumido que se ha tomado como BW el de la señal.

Para los tiempos de señal (Tsignal) se han tomado los tiempos calculados en el punto *6.6 Escenarios de Trabajo:*

- *Outdoor Opensky→ C/No=44dB*
- *Outdoor Urbano→ C/No=28 dB*

GPS/L1

Dispositivo	Aplicación	Precisión requerida	C/No	Tsignal (ms)	Quantization	BW	chunk_size (KB)
ADALM_Pluto	outdoor opensky	2	44	10	2	61,44 (20,46)	153,6
Airspy Mini	outdoor opensky	2	44	35	2	6	26,25
BladeRF	outdoor opensky	2	44	10	2	40 (20,46)	100
Free_SRP	outdoor opensky	2	44	10	2	61,44 (20,46)	153,6
HackRF	outdoor opensky	2	44	10	2	20	50
LimeSDR	outdoor opensky	2	44	10	2	61,44 (20,46)	153,6
RTL-SDR	outdoor opensky	2	44	90	2	2,4	207,36
Sideqik	outdoor opensky	2	44	10	2	50(20,46)	153,6
SiGe	outdoor opensky	2	44	26	2	8	26,598
Ettus 210	outdoor opensky	2	44	10	2	25(20,46)	250
Ettus x310	outdoor opensky	2	44	10	2	200(20,46)	500
XTRX	outdoor opensky	2	44	10	2	160(20,46)	400
ADALM_Pluto	outdoor urbano	2	28	400	2	61,44(20,46)	6144
Airspy Mini	outdoor urbano	2	28	2700	2	6	2025
BladeRF	outdoor urbano	2	28	400	2	40(20,46)	4000
Free_SRP	outdoor urbano	2	28	400	2	61,44(20,46)	6144
HackRF	outdoor urbano	2	28	400	2	20	2000
Lime SDR	outdoor urbano	2	28	400	2	61,44(20,46)	6144
RTL-SDR	outdoor urbano	2	28	3700	2	2,4	8524,8
Sideqik	outdoor urbano	2	28	400	2	50(20,46)	6144
SiGe	outdoor urbano	2	28	1000	2	8	1023
Ettus 210	outdoor urbano	2	28	400	2	25(20,46)	10000
Ettus x310	outdoor urbano	2	28	400	2	200(20,46)	20000
XTRX	outdoor urbano	2	28	400	2	160(20,46)	16000

Figura 74. Chunk Size GPS/L1

Galileo/E5							
Dispositivo	Aplicación	Precisión requerida	C/No	Tsignal (ms)	Quantization	BW	chunk_size (KB)
ADALM_Pluto	outdoor opensky	2	44	1	2	61,44 (51,150)	15,36
Airspy Mini	outdoor opensky	2	44	4	2	6	3
BladeRF	outdoor opensky	2	44	1	2	40	10
Free_SRP	outdoor opensky	2	44	1	2	61,44(51,150)	15,36
HackRF	outdoor opensky	2	44	1	2	20	5
LimeSDR	outdoor opensky	2	44	1	2	61,44(51,150)	15,36
RTL-SDR	outdoor opensky	2	44	24	2	2,4	55,296
Sideqik	outdoor opensky	2	44	1	2	50	15,36
SiGe	outdoor opensky	2	44	2	2	8	2,046
Ettus 210	outdoor opensky	2	44	1	2	25	25
Ettus x310	outdoor opensky	2	44	1	2	200(51,150)	50
XTRX	outdoor opensky	2	44	1	2	160(51,150)	40
ADALM_Pluto	outdoor urbano	2	28	35	2	61,44(51,150)	537,6
Airspy Mini	outdoor urbano	2	28	160	2	6	120
BladeRF	outdoor urbano	2	28	20	2	40	200
Free_SRP	outdoor urbano	2	28	40	2	61,44(51,150)	614,4
HackRF	outdoor urbano	2	28	40	2	20	200
Lime SDR	outdoor urbano	2	28	14	2	61,44(51,150)	215,04
RTL-SDR	outdoor urbano	2	28	900	2	2,4	2073,6
Sideqik	outdoor urbano	2	28	17	2	50(51,150)	261,12
SiGe	outdoor urbano	2	28	40	2	8	40,92
Ettus 210	outdoor urbano	2	28	40	2	25	1000
Ettus x310	outdoor urbano	2	28	40	2	200(51,150)	2000
XTRX	outdoor urbano	2	28	40	2	160(51,150)	1600

Figura 75. Chunk Size Galileo/E5

En las figuras anteriores, se puede ver el tamaño del chunk a transmitir en los distintos escenarios outdoor. Aunque la resolución del ADC de los distintos dispositivos es en general mayor de 2 bits, carece de sentido utilizar más de 2 bits de cuantificación por lo que se supone se ha configurado como valor de cuantificación 2 bits para evitar la perdida de recursos.

Puede verse en las tablas anteriores que tenemos ciertos escenarios donde debido a que la señal viene con una C/No muy baja, son necesarios tiempos de señal a procesar elevados para detectar el pico de correlación. Estos tiempos de señal elevados hacen que el tamaño del chunk sea elevado. Se ha señalado en rojo aquellos chunks cuyo tamaño está por encima del MB.
Tambien podemos ver dispositivos cuya frecuencia de muestreo Fs, es muy baja (por debajo de los 2 MHz para GPS/L1 o por debajo de 20 MHz para Galileo/E5) dichos dispositivos e han marcado en rojo y no se incluyen en el análisis de consumos.

Por otra parte, no resulta sencillo determinar el consumo de potencia de cada uno de los sipositivos anteriores, por lo que por simplicidad y como referencia se va a tomar un consumo promedio de dispositivos IoT [80], en este caso el interfaz con la nube sería LTE, no obstante, el dispositivo analizado es un dispositivo USB 3.0 cat0, como gran parte de los dispositivos analizados en el apartado anterior.

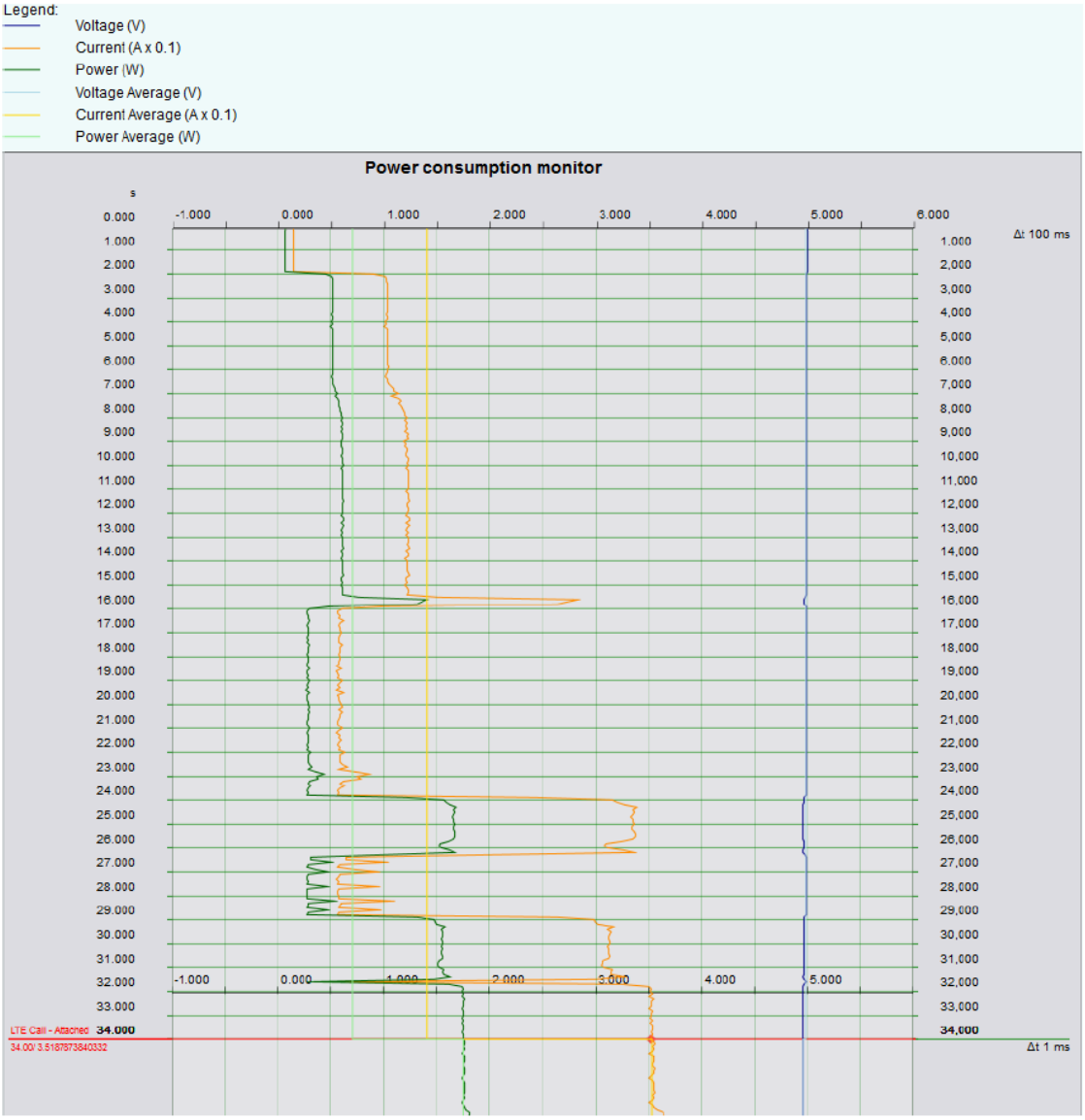

Figura 76. Consumo IoT device USB 3.0 [80]

Podemos ver que el dispositivo en el caso de estar en modo activo 1,5 W. En la siguiente imagen podemos ver que ese pico de consumo coincide con la transferencia de datos del dispositivo (uplink)

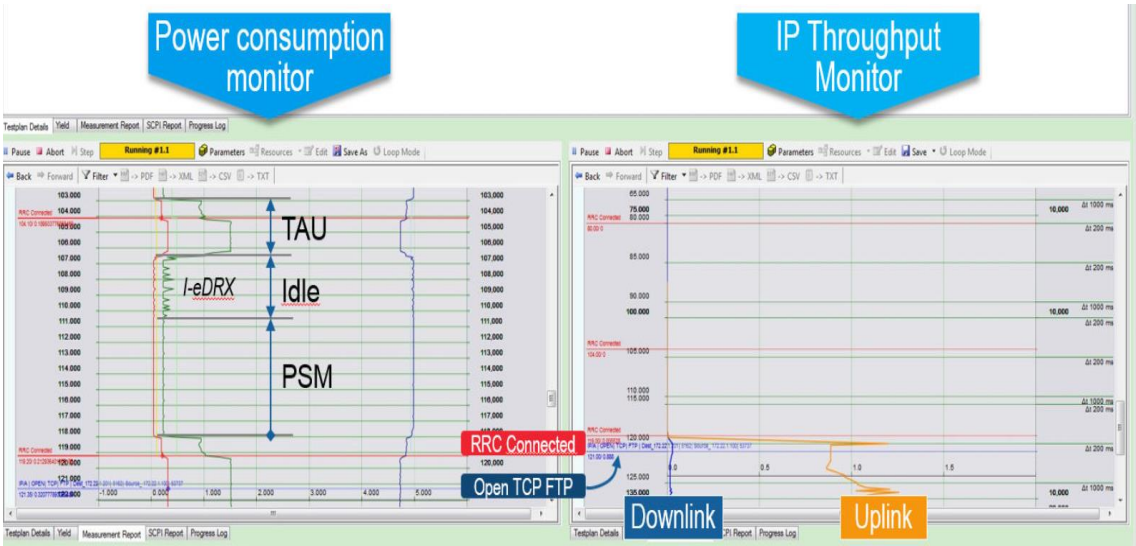

Figura 77. Consumo IoT con trafico IP [80]

Así pues, tomado ese 1,5 W de consumo obtenemos la siguiente comparativa en cuanto a los consumos de los dispositivos analizados anteriormente, en los escenarios opensky y urbano para enviar los distintos chunks:

<u>Escenario Opensky GPS/L1</u>

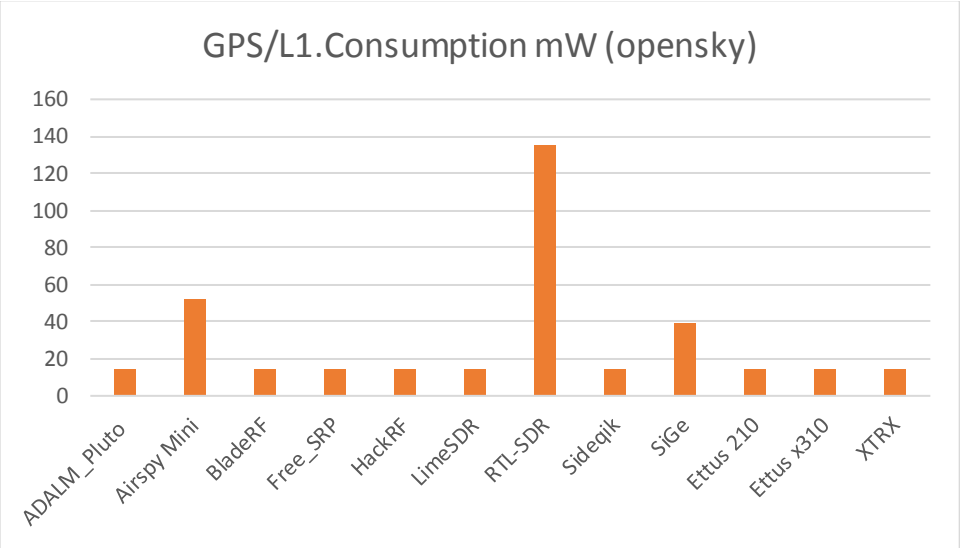

Figura 78. Consumos dispositivos opensky. GPS/L1

Escenario Urbano GPS/L1

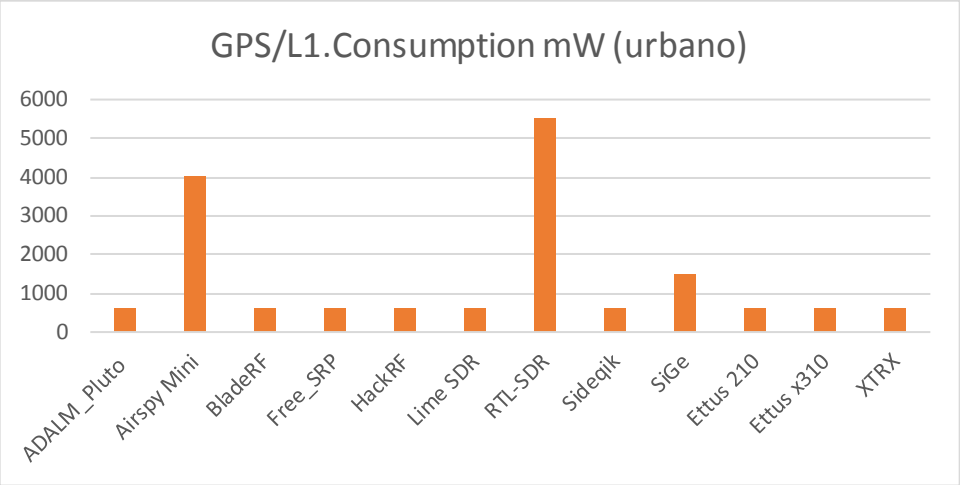

Figura 79. Consumos dispositivos urbano.GPS/L1

Escenario Opensky Galileo/E5

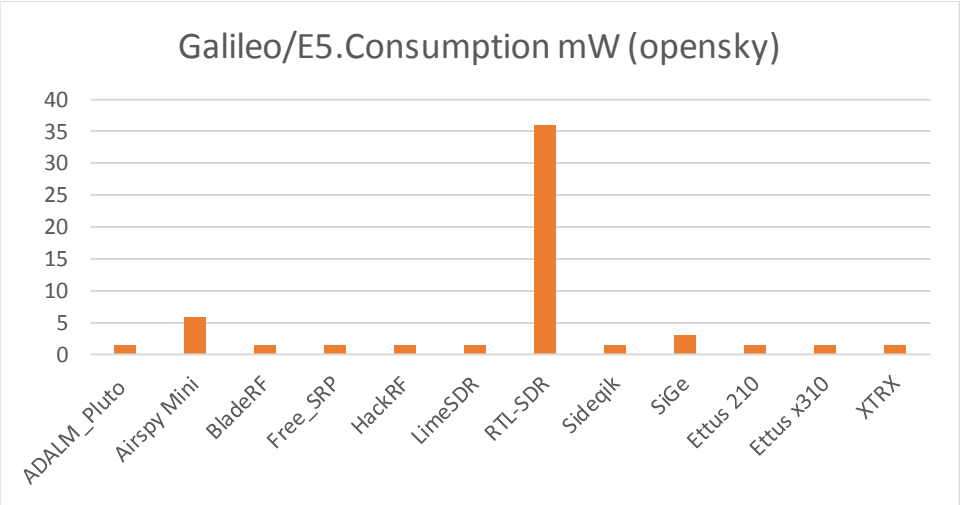

Figura 80. Consumos dispositivos opensky. Galileo/E5

Escenario urbano Galileo/E5

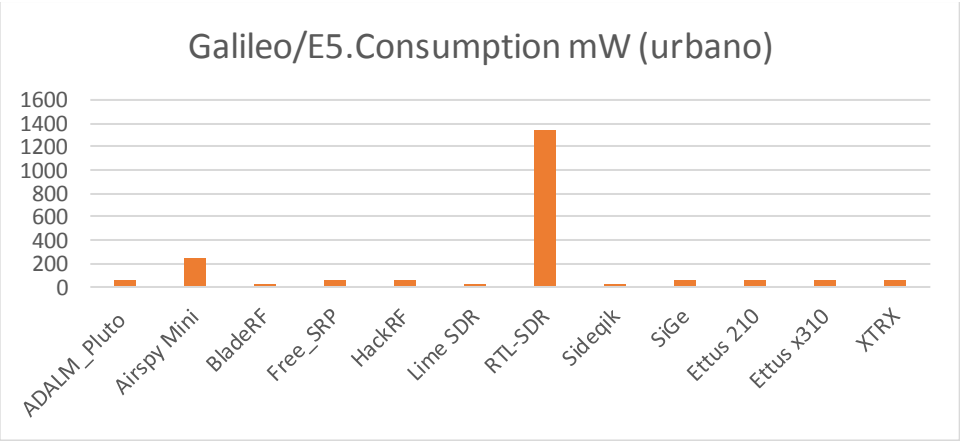

Figura 81. Consumos dispositivos urbano. Galileo/E5

10. Conclusiones

A lo largo de esta memoria, se ha tratado de segur la planificación inicial en la que se planeó dividir el TFM en dos bloques con una fuerte relación entre ellos, pero a su vez bien diferenciados.
En un primer bloque hemos hecho un breve repaso sobre los principios de funcionamiento de los sistemas GNSS, probablemente haciendo mayor hincapié en GPS dado que es, en el momento de realizar este trabajo, el sistema más extendido de posicionamiento por satélite.
Hemos visto cómo se conforma la señal GPS, así como los efectos de degradación que sufre la misma hasta llegar al receptor. Posteriormente hemos desglosado los diferentes bloques de los que está conformado un receptor GPS tradicional, en el que queda de manifiesto que tenemos un primer bloque denominado front-end, cuyo objetivo es capturar la señal RF y acomodarla para poder llevar a cabo su posterior procesamiento.
En la parte de procesamiento de la señal nos hemos centrado básicamente en el proceso de adquisición de la señal y hemos visto que, para detectar la débil señal satélite entre el ruido, se hace necesario un proceso de correlación de la señal recibida con las copias locales almacenadas en el receptor. Este proceso de correlación representa, en varias órdenes de magnitud, el proceso que demanda de mayor coste computacional y por tanto un mayor consumo de potencia.
Se ha realizado un análisis de las prestaciones de la etapa de adquisición de un receptor GNSS para las señales de GPS L1/L5 y Galileo E1/E5/E6. Se ha analizado cual es la probabilidad de detección (Pd) de la señal en función de distintos parámetros de adquisición como son el tiempo de integración coherente (Tcoh), el número de integraciones no coherentes (Ni), la relación portadora-densidad espectral de ruido (C/No), la probabilidad de falsa alarma global (PFA) y el número de celdas de búsqueda.
Haciendo uso de MATLAB se han programado una serie de funciones muy básicas que nos han permitido representar de manera gráfica la Probabilidad de detección de la señal en función de los parámetros mencionados con anterioridad (Tcoh, Ni, C/No) así como la precisión en los pseudorangos en función de estos mismos parámetros.
Posteriormente se ha programado también en MATLAB una sencilla GUI que permite definir un escenario típico en entornos IoT (definir la C/No, PFA, etc) y ver cómo va variando la probabilidad de detección en función de los distintos parámetros, así como la precisión en la medida.

En un segundo bloque, se han analizado una pequeña muestra de aplicaciones IoT, así como las exigencias en cuanto a la precisión necesaria de las mismas. Se trata de una pequeña muestra ya que el proceso de abarcar el conjunto de aplicaciones IoT que utilizan o pueden utilizar GNSS sería extensísimo y queda más allá del alcance de este trabajo.
Una vez definidos el escenario y precisiones requeridas por estas aplicaciones IoT haciendo uso de la GUI se han calculado los tiempos de integración coherentes necesarios, así como el número de integraciones no coherentes, lo que nos da una ligera idea de que aplicaciones IoT pueden hacer uso de GNSS stand-alone y cuales necesitaran del apoyo de otro tipo de tecnologías, bien

porque los tiempos de procesamiento impiden posicionamiento en tiempo real o bien por que el entorno hace que tengamos una relación C/No baja y nos obligue a unos tiempos de integración coherente o un número de integración no coherente elevado.
Así mismo, se constata que determinado tipo de aplicaciones IoT se encuentran en escenarios donde las exigencia en cuanto a tiempos de integración coherente o el número de integraciones no coherentes son elevados, lo que tiene unas implicaciones importantísimas en cuanto a las necesidades de procesamiento y por tanto en cuanto al consumo de energía, lo que choca de manera frontal con la filosofía y la tendencia de las aplicaciones IoT, que cada vez se orientan más hacia la miniaturización, dicha miniaturización va de la mano con la necesidad de reducir el consumo de los dispositivos, que a su vez va de la mano con la necesidad de reducir la carga de procesamiento que debe llevarse a cabo.
En este sentido una solución que va ganando relevancia y que parece dar una respuesta satisfactoria a esta aparente contradicción, es el uso de cloud GNSS donde los dispositivos IoT se comportan como meros front-end, es decir, su función, en cuanto al posicionamiento, sería la de capturar la señal GNSS y tras un proceso de filtrado y acondicionamiento de la señal a FI, es digitalizada y enviada a la nube, donde se llevan a cabo el resto de tareas con una alta carga de procesamiento, como por ejemplo el proceso de correlación de la señal de entrada con la copia local. De esta manera el receptor GNSS se simplifica al máximo con el consiguiente ahorro en capacidad de procesamiento, consumo y coste.
En este segundo bloque también hemos tratado una de las opciones disponibles para el procesamiento y almacenamiento de la señal en cloud, así como el posterior envío de reportes al usuario, en concreto se han analizado los servicios ofrecidos por Amazon (AWS)
Finalmente se han analizado diferentes opciones de dispositivo comerciales disponibles en el momento de redactar este trabajo que pueden utilizarse para implementar un servicio cloud GNSS.
El cloud GNSS es una idea reciente y todavía en desarrollo por lo que no es fácil encontrar cabezales RF que sean capaces de enviar las muestras de manera inalámbrica a la nube. En concreto los 11 dispositivos analizados representan una pequeña muestra de dispositivos SDR pero que ofrecen la capacidad de capturar las señales GNSS crudas y tras un proceso de adecuación, filtrado y digitalización y enviarlas a la nube. El inconveniente es que estos dispositivos por si solos son incapaces de enviar estas muestras digitalizadas de manera inalámbrica y necesitan conectarse mediante un interfaz (USB, PCIe o Ethernet) a un PC o Raspberry Pi para enviar dichas muestras a la nube. Además, otro inconveniente es la infrautilización de dichos dispositivos, ya que muchos de ellos cuentan con elementos que proporcionan capacidad de procesamiento, como FPGAs, que no se utilizan en el ámbito del cloud GNSS.
Cabe señalar que no se ha podido estudiar el uso de cloud GNSS en aplicaciones IoT en la que se usen módulos de comunicación inalámbrica eMTC, NB-IOT y EC-GSM-IoT, que parecen ofrecer ventajas como un bajo consumo de batería, bajo coste, mejoras en la seguridad (frente GSM/EDGE).

No obstante, se ha realizado un pequeño análisis del consumo de energía de esta serie de dispositivos SDR que podrían utilizarse como cabezales RF para enviar las muestras a la nube, tanto para escenarios de cielo abierto donde la señal llega con una potencia adecuada como para escenarios urbanos donde dicha señal llega con menos potencia.

11. Glosario

ADC= Analog Digital Converter.
A-GPS= Assisted GPS.
AMI=Amazon Machine Image.
AP=Agricultura de Precisión.
AWS=Amazaon Web Services.
BeiDou= Sistema de Navegación Satélite de la Republica Popular de China (Beidou es el nombre chino para la constelación de la Osa Mayor).
BDS= BeiDou Navigation Satellite System.
BPSK= Binary phase-shift keying.
CDMA =Code División Multiple Access.
C/A = Coarse/Adquisition.
CRB= Cramér–Rao bound.
CUDA =Compute Unified Device Architecture.
DNSS=Defense Navigation Satellite System.
EBS= Elastic Block Storage.
EC2= Elastic Compute Cloud.
EFS= Amazon Elastic File System.
FFT= Fast Fourier Transform.
FI= Frecuencia Intermedia
Galileo= programa europeo de radionavegación y posicionamiento por satélite
GBSC=Ground Based Scintillation Climatology.
GCCS =Geodetic Cloud Computing Service.
GLONASS= Global'naya Navigatsionnaya Sputnikovaya Sistema.
GNSS= Global Navigation Satellite System.
GPS= Global Positioning System.
GPU =Graphics Processing Unit.
GUI= Graphical User Interface.
GSA= European Global navigation Satellite system Agency.
Harvest Monitoring = Control y monitorización de los cultivos y cosechas
HS-GNSS= High Sensitivity GNSS
IaaS =Infraestructure As A Service.
IoT= Internet of Things.
IS= Ionospheric Scintillation.
KVM=Kernel-based Virtual Machine
LBS= Location Based Services.
M2M= Machine to Machine.
MCR=MATLAB Compiler Runtime.
MEMS= Micro Electro Mechanical Systems.
NB-IoT =*Narrowband IoT.*
Ni=Número de Integraciones no coherentes.
PaaS= plataforma como servicio.
Pd= Probabilidad de detección
Pfa= Probabilidad de falsa alarma (una celda)
PFA=Probabilidad de falsa alarma global (conjunto de celdas)
PNS= Pedestrian Navegation System.
PRN= Pseudorandom Noise.
PVT= Posición, Velocidad y Tiempo.
RINEX= Receiver Independent Exchange Format.

RTMC =Real Time Monitoring and Control Manager.
SaaS= software como servicio.
SDR= Software Defined Radio.
SIG= Sistemas de Información Geográfica.
SNR= signal-to-noise ratio.
SQS=Simple Queue Service.
Stand Alone= Posicionamiento haciendo uso en exclusiva de tecnología GNSS.
Tcoh= Tiempo de integración coherente de la señal.
TFM = Trabajo Fin de Master.
UAV= Unmanned Aerial Vehicle.

12. Bibliografía

[1] https://www.gsa.europa.eu/market/market-report (2018)

[2] https://www.gsa.europa.eu/system/files/reports/gnss_mr_2017.pdf (2018)
[3] https://www.gsa.europa.eu/system/files/reports/gnss_user_technology_report_webb.pdf (Año 2018)

[4] High-Sensitivity GPS - the Low-Cost Future of GNSS, Volker SCHWIEGER, Germany

[5] Evaluation of High Sensitivity GPS Receivers, Evaluation of High Sensitivity GPS Receivers

[6] DINGPOS, a GNSS-based Multi-sensor Demonstrator for Indoor Navigation: Preliminary Results

[7] http://www.gpsdancer.org/index.php (Año 2018)

[8] XYBRID CLOUD: A Cloud Based GNSS Receiver, Nima Sadrieh, Behnam Aminian and Muhammad Haris Afzal

[9] GNSS based services on cloud environment. L.Mossucca, L. Spogli, G. Caragnano, V. Romano, O.Terzo, G. de Franceschi, L.Alfonsi, E. Plakidis

[10] An Introduction to GNSS, Charles Jeffrey, NovAtel, 2010

[11] http://www.sequans.com/products-solutions/streamlitelte/cloe-lte-cat-m1-tracker-platform/ (Año 2018)

[12] https://accent-systems.com/es/producto/nbiot-tracker/ (Año 2018)

[13] http://tracktio.com/lora-tracking/ (Año 2018)

[14] Probability of Detection for GNSS Signals With Sign Transitions. Myrian Foucras, Olivier Julien, Christophe Macabiau, Bertrand Ekambi, Fayaz Bacard

[15] GNSS Market Report, issue 5, European GNSS Agency (Año 2017)

[16] https://press.pwc.com/News-releases/global-market-for-commercial-applications-of-drone-technology-valued-at-over--127-bn/s/ac04349e-c40d-4767-9f92-a4d219860cd2 (Año 2016)

[17] https://www.bcg.com/publications/2017/engineered-products-infrastructure-machinery-components-drones-go-work.aspx (Año 2017)

[18]https://www.npr.org/sections/goatsandsoda/2017/08/24/545589328/tanzania-gears-up-to-become-a-nation-of-medical-drones (Año 2017)

[19] https://www.mckinsey.com/industries/travel-transport-and-logistics/our-insights/how-customer-demands-are-reshaping-last-mile-delivery (Año 2016)

[20] https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011 (Año 2018)

[21]https://pressroom.ups.com/pressroom/ContentDetailsViewer.page?ConceptType=PressReleases&id=1487687844847-162 (Año 2018)

[22] https://x.company/projects/wing/ (Año 2018)

[23] Vehicle Routing Problems for Drone Delivery, Kevin Dorling , Jordan Heinsrichs, Geoffrey G.Messier and Sebastian Magierowski

[24] Desing Perspectives on Delivery Drones, Jia Xu, RAND Coorporation (Año 2017)

[25] http://insidegnss.com/pdf/Inside_GNSS-GNSS_Positioning_Navigation_and_Guidance-060214.pdf (Año 2014)

[26] https://mttr.net/ (Año 2018)

[27] https://www.mercedes-benz.com/en/mercedes-benz/vehicles/transporter/vans-drones-in-zurich/ (Año 2017)

[28] https://siroop.ch/ (Año 2018)

[29] https://www.cyphyworks.com/ (Año 2018)

[30] https://www.dji.com/es (año 2018)

[31] https://3dr.com/ (Año 2018)

[32] https://www.liveviewgps.com/pdf/Flash-Trac-BAT-Spec-Sheet.pdf (Año 2018)

[33] https://www.gsa.europa.eu/newsroom/news/gnss-and-internet-medical-things (Año 2016)

[34] https://www.unaliwear.com/ (Año 2018)

[35] https://traxfamily.com/tech-specs/ (Año 2018)

[36] http://www.suunto.com/es-ES/ (Año 2018)

[37] https://www.polar.com/es/productos (Año 2018)

[38] https://www.tomtom.com/es_es/sports/fitness-watches/ (Año 2018)

[39] https://es.wikipedia.org/wiki/Agricultura_de_precisi%C3%B3n (Año 2018)

[40] Agricultura de precisión: hacia la integración de datos espaciales en la producción agraria, Juan Agüera Vega, Universidad de Córdoba. Manuel Pérez Ruiz, Universidad de Sevilla, (diciembre 2013)

[41]http://www.unoosa.org/documents/pdf/psa/activities/2006/zambia/presentations/04-01-01.pdf (Año 2006, pag 37)

[42] http://topconcare.com/en/agriculture/precision-ag-gnss-receivers-base-stations/sgr-1/specifications/ (Año 2018)

[43] Informe Mobile en España y en el Mundo 2017. Ditendria (Año 2017)

[44] Ubiquitous Positioning and Mobile Location-Based Services in Smart phones. Chen, Ruizhi. Editorial IGI Global (junio 2012)

[45] Challenges in indoor Global Navigation Satellite Systems. Gonzalo Seco-Granados, José A. López Salcedo, David Jiménez Baños y Gustavo López Risueño, (febrero 2012)

[46] https://play.google.com/store/apps/details?id=cache.wind.gps&hl=es (Año 2018)

[47] https://consumer.huawei.com/es/phones/p10/specs/ (Año 2018)

[48] Autonomus Vehicle Positioning with GPS in Urban Canyons Envyronments. Y.J.Cui y S.S.Ge, Universidad de Singapur (Mayo, 2001)

[49] Demostration of Cloud GNSS Signal Processing.
V. Lucas-Sabola, G. Seco-Granados, J.A. López-Salcedo, Universitat Autònoma de Barcelona (UAB), Spain; J.A. Garcia-Molina, European Space Agency (ESA/ESTEC), TEC-ETN, and HE Space, The Netherlands M. Crisci, ESA, The Netherlands (septiembre 2016)

[50] https://azure.microsoft.com/es-es/overview/what-is-cloud-computing/ (Año 2018)

[51] https://azure.microsoft.com/es-es/ (Año 2018)

[52] https://aws.amazon.com/es/ (Año 2018)

[53] https://www.ibm.com/cloud/ (Año 2018)

[54] https://www.salesforce.com/es/products/service-cloud/overview/ (Año 2018)
[55] https://cloudplatform.sap.com/index.html (Año 2018)

[56] https://cloud.google.com/ (Año 2018)

[57] https://aws.amazon.com/es/ec2/instance-types/ (Año 2018)

[58] https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/using-regions-availability-zones.html (Año 2018)

[59] https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/AMIs.html (Año 2018)

[60] https://aws.amazon.com/es/message-queue/ (Año 2018)

[61] Cloud GNSS receivers: New advanced applications made possible, Vicente Lucas-Sabola, Gonzalo Seco-Granados, José A. López-Salcedo, José A. García-Molina, Massimo Crisci (Año 2016)

[62] https://gnss-sdr.org (Año 2018)

[63] http://gpsworld.com/innovation-the-continued-evolution-of-the-gnss-software-defined-radio/ (Año 2018)

[64] https://wiki.analog.com/university/tools/pluto (Año 2018)

[65] http://www.analog.com/media/en/technical-documentation/data-sheets/AD9363.pdf (Año 2018)

[66] https://airspy.com/airspy-mini/ (Año 2018)

[67] https://es.wikipedia.org/wiki/Raspberry_Pi (Año 2018)

[68] https://nuand.com/ (Año 2018)

[69] http://freesrp.org/ (Año 2018)

[70] https://myriadrf.org/ (Año 2018)

[71] https://greatscottgadgets.com/hackrf/ (Año 2018)

[72] https://www.crowdsupply.com/lime-micro/limesdr (Año 2018)

[73] https://www.rtl-sdr.com/buy-rtl-sdr-dvb-t-dongles/ (Año 2018)

[74] https://www.sdrplay.com/rsp1a/ (Año 2018)

[75] https://epiqsolutions.com/sidekiq/SidekiqFamily-Flyer_20160511.pdf (Año 2018)

[76] http://ccar.colorado.edu/gnss/files/SE4110L_Datasheet_Rev3.pdf (Año 2018)

[77] https://www.ettus.com/product/details/UN210-KIT (Año 2018)

[78] https://www.ettus.com/product/details/X310-KIT (Año 2018)

[79] https://xtrx.io/ (Año 2018)

[80] Rohde_and_Schawarz_IoT_consumption (Año 2017)

13. Anexos

Código MATLAB (GUI)

Menú Principal GUI

MAIN.m

```
function varargout = Main(varargin)
% MAIN MATLAB code for Main.fig
%      MAIN, by itself, creates a new MAIN or raises the existing
%      singleton*.
%
%      H = MAIN returns the handle to a new MAIN or the handle to
%      the existing singleton*.
%
%      MAIN('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MAIN.M with the given input
arguments.
%
%      MAIN('Property','Value',...) creates a new MAIN or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before Main_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Main_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Main

% Last Modified by GUIDE v2.5 06-May-2018 07:59:12

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Main_OpeningFcn, ...
                   'gui_OutputFcn',  @Main_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before Main is made visible.
function Main_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Main (see VARARGIN)

% Choose default command line output for Main
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Main wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Main_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox1
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
listbox1


% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
x=get(handles.popupmenu1,'Value');
if x==1
hGui=Pd_Ni('Visible','on');
elseif x==2
hGui=Pd_Tcoh('Visible','on');
 elseif x==3
hGui=Pd_CNo('Visible','on');
%  disp ('Sin programar');
else x==4
    hGui=Precision('Visible','on');
%  disp ('Sin programar');

end;

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu1

%get(hObject,'Value')
% x global;
% x=get(hObject,'Value');
% % disp (x);
% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Pd en función de Ni

Pd_Ni.m

```
function varargout = Pd_Ni(varargin)
% PD_NI MATLAB code for Pd_Ni.fig
%      PD_NI, by itself, creates a new PD_NI or raises the existing
%      singleton*.
%
%      H = PD_NI returns the handle to a new PD_NI or the handle to
%      the existing singleton*.
```

```
%
%      PD_NI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PD_NI.M with the given input
arguments.
%
%      PD_NI('Property','Value',...) creates a new PD_NI or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before Pd_Ni_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Pd_Ni_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Pd_Ni

% Last Modified by GUIDE v2.5 03-Jun-2018 21:04:58

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Pd_Ni_OpeningFcn, ...
                   'gui_OutputFcn',  @Pd_Ni_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Pd_Ni is made visible.
function Pd_Ni_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Pd_Ni (see VARARGIN)

% Choose default command line output for Pd_Ni
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Pd_Ni wait for user response (see UIRESUME)
% uiwait(handles.figure1);
```

```
% --- Outputs from this function are returned to the command line.
function varargout = Pd_Ni_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%i=i+1;
global indice
global l
global grafica
l={};
%grafica=[0:10];
Tcoh=str2double(get(handles.edit1,'String'))/10^3
TimeBin=str2double(get(handles.edit4,'String'))
FrecBin=str2double(get(handles.edit5,'String'))
Pfa=str2double(get(handles.edit2,'String'))/(TimeBin*FrecBin)
CNo=str2double(get(handles.edit3,'String'))
Ni=linspace (1,1000,1000)
gamma=chi2inv(1-Pfa,2*Ni)
SNR_outcohcorr1=10^(CNo/10)*Tcoh
Pd_Ni=1-ncx2cdf(gamma,2*Ni,SNR_outcohcorr1*Ni)
set(handles.togglebutton1,'Visible','on');
Hold_on= get(handles.togglebutton1,'Value')
if Hold_on==0
    indice = 1;
    hold off;
%     plot (Ni,Pd_Ni,'b','linewidth',3); %Pd En funcion de Ni
%     xlim ([0,1000]);
%     ylim ([0,1]);
%     %legend ( num2str(Tcoh));
%     ylabel(['Probabilidad de Deteccion con Tcoh= ',num2str(Tcoh),
'segundos']);
%     xlabel('Ni Integraciones no coherentes');
elseif Hold_on==1
    hold on;
%     rndclr=[rand, rand, rand]
%     plot (Ni,Pd_Ni,'color',rndclr,'linewidth',3); %Pd En funcion de
Ni
%     xlim ([0,1000]);
%     ylim ([0,1])
%     legend ('TCoherente: '+ num2str(Tcoh));
%     ylabel(['Probabilidad de Deteccion con Tcoh= ',num2str(Tcoh),
'segundos']);
%     xlabel('Ni Integraciones no coherentes');
end
rndclr=[rand, rand, rand];
% y{indice}=plot (Ni,Pd_Ni,'color',rndclr,'linewidth',3);
plot (Ni,Pd_Ni,'color',rndclr,'linewidth',3);
```

```
% l{indice}= strcat('Tcoh = ',num2str(Tcoh))
xlim ([0,1000]);
ylim ([0,1]);
% legend (num2str(Tcoh))
% indice = indice +1
%  for i= 1:indice-1
%     legend(y,l(i));
% end
%
% legend (y)
ylabel(['Probabilidad de Deteccion con Tcoh= ',num2str(Tcoh),
'segundos']);
xlabel('Ni Integraciones no coherentes');




function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1
as a double

Tcoh_aux=str2double(get(hObject,'String'));
if Tcoh_aux < 1 || Tcoh_aux>100
   set (handles.edit1,'String','No Valid');

end


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2
as a double
```

```
Pfa=str2double(get(hObject,'String'));
if Pfa < 0 || Pfa>1
   set (handles.edit2,'String','No Valid');
end

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3
as a double
CNo_aux=str2double(get(hObject,'String'))
if CNo_aux < 0
   set (handles.edit3,'String','No Valid');
end


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: place code in OpeningFcn to populate axes1
```

```
function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4
as a double

TimeBin=str2double(get(hObject,'String'));
if TimeBin < 1
   set (handles.edit4,'String','No Valid');
end
% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5
as a double

FrecuencyBin=str2double(get(hObject,'String'));
if FrecuencyBin < 1
   set (handles.edit5,'String','No Valid');
end

% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in togglebutton1.
function togglebutton1_Callback(hObject, eventdata, handles)
% hObject    handle to togglebutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of togglebutton1


% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over
togglebutton1.
function togglebutton1_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to togglebutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Pd en función de Tcoh

Pd_Tcoh.m

```
function varargout = Pd_Tcoh(varargin)
% PD_TCOH MATLAB code for Pd_Tcoh.fig
%      PD_TCOH, by itself, creates a new PD_TCOH or raises the
existing
%      singleton*.
%
%      H = PD_TCOH returns the handle to a new PD_TCOH or the handle
to
%      the existing singleton*.
%
%      PD_TCOH('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in PD_TCOH.M with the given input
arguments.
%
%      PD_TCOH('Property','Value',...) creates a new PD_TCOH or raises
the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before Pd_Tcoh_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Pd_Tcoh_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Pd_Tcoh

% Last Modified by GUIDE v2.5 04-Jun-2018 13:48:16

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
```

```
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Pd_Tcoh_OpeningFcn, ...
                   'gui_OutputFcn',  @Pd_Tcoh_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Pd_Tcoh is made visible.
function Pd_Tcoh_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Pd_Tcoh (see VARARGIN)

% Choose default command line output for Pd_Tcoh
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Pd_Tcoh wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Pd_Tcoh_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc;
Ni=str2double(get(handles.edit1,'String'))
TimeBin=str2double(get(handles.edit4,'String'))
FrecBin=str2double(get(handles.edit5,'String'))
Pfa=str2double(get(handles.edit2,'String'))/(TimeBin*FrecBin)
CNo=str2double(get(handles.edit3,'String'))
Tcoh= linspace (10^-3,100*10^-3,100000);
```

```
gamma=chi2inv(1-Pfa,2*Ni)
SNR_outcohcorr1=10^(CNo/10)*Tcoh
Pd_Tcoh=1-ncx2cdf(gamma,2*Ni,SNR_outcohcorr1*Ni)
set(handles.togglebutton1,'Visible','on');
Hold_on= get(handles.togglebutton1,'Value')

if Hold_on==0
    hold off;
else
    hold all;
end
rndclr=[rand, rand, rand];
plot (Tcoh,Pd_Tcoh,'color',rndclr,'linewidth',3); %Pd function on Tcoh
xlim ([0,0.1]);
ylim ([0,1]);
ylabel(['Probabilidad de Deteccion con Ni=',num2str(Ni)]);
xlabel('Tcoh Tiempo de Integracion coherente en s');
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1
as a double

Ni_aux=str2double(get(hObject,'String'))

if Ni_aux < 1
   set (handles.edit1,'String','No Valid');
end

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2
as a double

Pfa_aux=str2double(get(hObject,'String'))
```

```
if Pfa_aux < 0 || Pfa_aux>1
   set (handles.edit2,'String','No Valid');
end

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3
as a double
CNo_aux=str2double(get(hObject,'String'))
if CNo_aux < 1
   set (handles.edit3,'String','No Valid');
end



% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: place code in OpeningFcn to populate axes1
```

```
function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4
as a double

TimeBin=str2double(get(hObject,'String'));
if TimeBin < 1
   set (handles.edit5,'String','No Valid');
end

% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5
as a double
FrecuencyBin=str2double(get(hObject,'String'));
if FrecuencyBin < 1
   set (handles.edit5,'String','No Valid');
end
% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in togglebutton1.
function togglebutton1_Callback(hObject, eventdata, handles)
% hObject    handle to togglebutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of togglebutton1
```

Pd en función de C/No

Pd_CNo.m

```
function varargout = Pd_CNo(varargin)
% PD_CNO MATLAB code for Pd_CNo.fig
%      PD_CNO, by itself, creates a new PD_CNO or raises the existing
%      singleton*.
%
%      H = PD_CNO returns the handle to a new PD_CNO or the handle to
%      the existing singleton*.
%
%      PD_CNO('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in PD_CNO.M with the given input
arguments.
%
%      PD_CNO('Property','Value',...) creates a new PD_CNO or raises
the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before Pd_CNo_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Pd_CNo_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Pd_CNo

% Last Modified by GUIDE v2.5 23-May-2018 21:15:59

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Pd_CNo_OpeningFcn, ...
                   'gui_OutputFcn',  @Pd_CNo_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Pd_CNo is made visible.
function Pd_CNo_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Pd_CNo (see VARARGIN)

% Choose default command line output for Pd_CNo
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Pd_CNo wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Pd_CNo_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc;
Tcoh=str2double(get(handles.edit1,'String'))/10^3;
TimeBin=str2double(get(handles.edit4,'String'))
FrecBin=str2double(get(handles.edit5,'String'))
Pfa=(str2double(get(handles.edit2,'String'))/(TimeBin*FrecBin))
Ni=str2double(get(handles.edit3,'String'))
CNo=linspace (1,100,100);
gamma=chi2inv(1-Pfa,2*Ni);
SNR_outcohcorr1=2*10.^(CNo/10)*Tcoh;
Pd_CNo=1-ncx2cdf(gamma,2*Ni,SNR_outcohcorr1*Ni)

%***********************
CNo_test=15;
Tcoh_test=0.020;
Pfa_test=1e-6/(1023*20);
Ni_test=200;
gamma_test=chi2inv(1-Pfa_test,2*Ni_test);
SNR_test=2*10.^(CNo_test/10)*Tcoh_test;
Pd_CNo_test=(1-ncx2cdf(gamma_test,2*Ni_test,SNR_test*Ni_test))
```

```
%**************************

plot (CNo,Pd_CNo,'g','linewidth',3); %Pd function on Tcoh
ylabel(['Probabilidad de Deteccion con Ni= ',num2str(Ni),' y Tcoh=
',num2str(Tcoh),' ms']);
xlabel('Relacion Portadora vs Densidad Espectral de Ruido');



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1
as a double

Tcoh_aux=str2double(get(hObject,'String'))
if Tcoh_aux < 1 || Tcoh_aux>1000
   set (handles.edit1,'String','No Valid');
end


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2
as a double

Pfa_aux=str2double(get(hObject,'String'));
if Pfa_aux < 0 || Pfa_aux>100
   set (handles.edit2,'String','No Valid');
end

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3
as a double
Ni_aux=str2double(get(hObject,'String'))
if Ni_aux < 1
   set (handles.edit3,'String','No Valid');
end


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: place code in OpeningFcn to populate axes1



function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
```

```
%        str2double(get(hObject,'String')) returns contents of edit4
as a double
TimeBin=str2double(get(hObject,'String'));
if TimeBin < 1
   set (handles.edit4,'String','No Valid');
end


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5
as a double
FrecuencyBin=str2double(get(hObject,'String'));
if FrecuencyBin < 1
   set (handles.edit5,'String','No Valid');
end

% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Precisión en los pseudorangos

Precisión.m

```
function varargout = Precision(varargin)
% PRECISION MATLAB code for Precision.fig
%      PRECISION, by itself, creates a new PRECISION or raises the
existing
```

```
%      singleton*.
%
%      H = PRECISION returns the handle to a new PRECISION or the
handle to
%      the existing singleton*.
%
%      PRECISION('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in PRECISION.M with the given input
arguments.
%
%      PRECISION('Property','Value',...) creates a new PRECISION or
raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before Precision_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Precision_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Precision

% Last Modified by GUIDE v2.5 07-May-2018 22:24:09

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Precision_OpeningFcn, ...
                   'gui_OutputFcn',  @Precision_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Precision is made visible.
function Precision_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Precision (see VARARGIN)

% Choose default command line output for Precision
```

```
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Precision wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Precision_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc;
Amp=1;%Amplitud  de  la  seï¿½al,  no  es  necesaria  porque  se
simplifica  en  la  ecuaciï¿½n
c=3e8; %velocidad de la luz
Tcoh=str2double(get(handles.edit1,'String'))/10^3
Ni=str2double(get(handles.edit2,'String'))
CNo=str2double(get(handles.edit3,'String'))
CNo_lin=10^(CNo/10);
senial=get(handles.SenialTag,'Value')
disp (senial);


if senial==1 %Señal GPS L1 ancho banda receptor 4MHz
    limsup=2*10^6 %  lï¿½mite  superior  de  la  integral
    liminf=-2*10^6 %  lï¿½mite  inferior  de  la  integral
    Tchip=1/1.023e6
elseif senial==2 %Señal GPS L5 ancho banda receptor 20MHz
    limsup=10*10^6 %  lï¿½mite  superior  de  la  integral
    liminf=-10*10^6 %  lï¿½mite  inferior  de  la  integral
    Tchip=1/10.23e6
elseif senial==3 %Señal Galileo E1 ancho banda receptor 4MHz
    limsup=2*10^6 %  lï¿½mite  superior  de  la  integral
    liminf=-2*10^6 %  lï¿½mite  inferior  de  la  integral
    Tchip=1/1.23e6
elseif senial==4 %Señal Galileo E5 ancho banda receptor 20MHz
    limsup=10*10^6 %  lï¿½mite  superior  de  la  integral
    liminf=-10*10^6 %  lï¿½mite  inferior  de  la  integral
    Tchip=1/10.23e6
elseif senial==5 %Señal Galileo E6 ancho banda receptor 20MHz
    limsup=10*10^6 %  lï¿½mite  superior  de  la  integral
    liminf=-10*10^6 %  lï¿½mite  inferior  de  la  integral
    Tchip=1/5.115e6
end;
```

```
Num=integral(@(f)((2*pi.*f).^2).*abs(((Amp./(pi.*f)).*sin(pi*Tchip.*f)
)).^2,liminf,limsup);
Den=integral(@(f)abs(((Amp./(pi.*f)).*(sin(pi*Tchip.*f))).^2.),liminf,
limsup);
Bms_cuad=Num/Den
CRB=1/(2*CNo_lin*Tcoh*Ni*Bms_cuad)

precision=c*sqrt(CRB)
set(handles.edit4,'String',precision);



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1
as a double

Tcoh_aux=str2double(get(hObject,'String'));
if Tcoh_aux < 1 || Tcoh_aux>100
   set (handles.edit1,'String','No Valid');
end

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2
as a double

Ni_aux=str2double(get(hObject,'String'))

if Ni_aux < 1
   set (handles.edit2,'String','No Valid');
end
```

```
% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3
as a double
CNo_aux=str2double(get(hObject,'String'))
if CNo_aux < 1
   set (handles.edit3,'String','No Valid');
end


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: place code in OpeningFcn to populate axes1



function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4
as a double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in SenialTag.
function SenialTag_Callback(hObject, eventdata, handles)
% hObject    handle to SenialTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns SenialTag
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
SenialTag
% senial=get(handles.SenialTag,'Value')


% --- Executes during object creation, after setting all properties.
function SenialTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to SenialTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

i