UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escuela Técnica Superior de Ingeniería Informática

Universitat Politècnica de València

CONFIGURACIÓN Y PROGRAMACIÓN DE UN DISPOSITIVO DE GEOLOCALIZACIÓN DIFERENCIAL PARA VEHÍCULOS MÓVILES AUTÓNOMOS.

Proyecto final de carrera

Ingeniero en Informatica

Autor: Carlos Bataller Martí

Director: Ángel Valera Fernández

27 de Septiembre de 2012

Índice

1. Introducción

A lo largo de la historia, el modo en que nos movemos ha ido evolucionando en pos de recorrer mayores distancias con menor esfuerzo y más seguridad.

Primero se andaba, luego se usaron caballos, siguieron los carros, barcos, trenes, coches, aviones… y dentro de cada medio de transporte se crearon útiles para hacer el viaje más llevadero y seguro. Zapatos, sillas para los caballos, se paso del remo a la vela y luego al motor en los barcos, radares para evitar colisiones en los aviones, y mejoras en el motor a los coches y su estructura para hacerla más cómoda y aerodinámica, añadiendo cinturones y airbags para hacerlo más seguro, e incluyeron dispositivos para mejorar la experiencia del viaje, como un mechero o un navegador GPS para no perderse y llegar antes al destino.

Esta evolución es constante y nunca termina, siempre hay gente buscando la manera de mejorar la forma que tiene el hombre de moverse en cualquiera de sus ámbitos. Incluso cosas creadas para mejorar un aspecto de la vida, más tarde se descubre que también se puede utilizar para otras muchas.

Los hay quien su motivación es simplemente la de descubrir o la de ayudar a la gente, y también hay quien tiene motivaciones de carácter militar. Esto último normalmente son instituciones o proyectos creadas o subvencionadas por los gobiernos, buscando siempre la forma de transportar a la mayor cantidad posible de gente a la vez, mejorando la seguridad para perder el mínimo número de personas posibles y creando vehículos que se adapten mejor a cualquier tipo de terreno. Estas mejoras de ámbito militar suelen contar con más recursos y, al final, siempre acaban adaptándose para que la gente civil pueda aprovecharse de ellas también.

Muchos de los inventos de la humanidad se pensaron en principio para uso militar, viéndose más tarde que también se le podía dar un uso cotidiano. Claros ejemplos de esto son la comida enlatada, la gabardina, el clínex, el vehículo todoterreno o el GPS. Casi todos ellos de uso común para todo el mundo. Es de este último, el GPS, del que hablaremos ahora.

El GPS fue creado por el gobierno Estadounidense para poder seguir en todo momento el movimiento de sus tropas, siendo de gran ayuda para poder coordinarse y localizar unidades. Contamos con el uso civil debido a un incidente ocurrido en que un avión de pasajeros coreano fue derribado por las fuerzas soviéticas al penetrar en su espacio aéreo debido a un fallo de navegación. Desde entonces el sistema de navegación GPS está disponible para usos civiles, en pos de que no vuelva a ocurrir una desgracia como esta.

Más tarde, los rusos y los europeos crearon sus propios sistemas de navegación por satélite, mientras los chinos están empezando a crear el suyo propio en la actualidad.

Con esta gran cantidad de satélites y sistemas de navegación a los que es fácil acceder, la inventiva del hombre se ha puesto en marcha y ha empezado a crear útiles para facilitar la vida a la gente a partir de este gran invento.

Desde poder localizar tu móvil vía GPS por si lo has perdido, hasta dispositivos de navegación integrados en los coches con mapas que le buscan a uno la ruta más corta desde donde se encuentra hasta donde quiere ir, indicándole giros y cambios de carretera según se avanza.

Esto ya de por sí es un gran avance, es la primera vez en la historia en que todo el mundo puede contar con alguien que le indique el camino por lugares por donde no había estado antes ni había planeado pasar. Antes con el uso de mapas se podía alcanzar también el destino, pero con el GPS, el porcentaje de gente que se pierde por el camino o tiene que parar a preguntar se ha reducido drásticamente, mejorando la comodidad del viajero y la experiencia del viaje.

Como se ha visto, el progreso no se detiene, y nosotros vamos a intentar formar parte de él.

Todo el mundo ha visto esas películas futuristas de ciencia ficción en que los coches van solos a su destino. Nosotros, con la ayuda del GPS, además de otros sensores, queremos hacerlo realidad.

Justificación

El hecho de que un vehículo pueda conducirse sin la intervención de ningún humano presenta muchas ventajas posibles.

Por una parte, mejora la comodidad al viajar, ya que la persona ya no tendrá que preocuparse en cómo llegar a su destino, el vehículo irá solo. Ya no se tendrá que estar atento al tráfico, ni si quiera aprender a conducir.

También se producirán menos accidentes relacionados con errores humanos, como pueda ser distraerse o quedarse dormidos al volante.

Además, si todos los vehículos llevasen el mismo sistema, podrían coordinarse entre ellos para no chocar.

Incluso se podrán enviar vehículos no tripulados por las calles sin peligro, simplemente para transportar objetos sin que nadie tenga que desplazarse para ello.

Todas estas ideas serán posibles si conseguimos construir un vehículo que se mueva de manera autónoma. Para ello, el GPS es una parte indispensable, ya que indica al vehículo constantemente de su posición, pudiendo calcular rutas o seguir coordenadas ya establecidas que marquen un recorrido. Además se podría informar a otros vehículos de nuestra posición para que se coordinasen para realizar maniobras o evitar chocar.

Para conseguirlo, se necesita una precisión en la posición muy alta, por lo que hará falta un dispositivo GPS que esté a la altura. También hará falta un programa que recoja la información que transmite el GPS, ya que si no se procesa la información, ésta no sirve de nada.

Debido a que el vehículo será autónomo, pudiendo programarlo para que realice tareas, este podría entrar dentro de la denominación “Robot”.

Objetivos

- Estudiar qué tipos de GPS existen y elegir uno que se avenga a nuestras exigencias.
- Analizar la información que se puede sacar de este y su calidad, y si nos puede llegar a ser útil para nuestro objetivo.
- Realizar un programa informático que recoja la información del GPS para poder procesarla debidamente para que un vehículo pueda llegar a ser autónomo.

2. Apartado teórico

Fue en 1921 cuando se escucho por primera vez la palabra “robot”, utilizada por el escritor checo Karel Capek en su obra de teatro R.U.R. La palabra viene del vocablo polaco “Robata”, que significa trabajo, con connotaciones de servidumbre, trabajo forzado o esclavitud.

Tras esto, a toda máquina autómata creada a partir de entonces, ha sido llamada robot.

La primera fue en 1961, George Devol patentaba el primer robot programable de la historia, conocido como Unimate, estableciendo las bases de la robótica industrial moderna.

Estos robots fueron evolucionando, llegando a utilizarse en prácticamente todas las áreas productivas. No solo esto sino que también salieron del ámbito industrial, convirtiéndose en apoyo para muchas tareas en otros ámbitos.

2.1. Robots industriales

La creciente utilización de robots industriales en el proceso productivo, ha dado lugar al desarrollo de controladores industriales rápidos y potentes, basados en microprocesadores, así como un empleo de servos en bucle cerrado que permiten establecer con exactitud la posición real de los elementos del robot y su desviación o error. Esta evolución ha dado origen a una serie de tipos de robots, que se citan a continuación:

2.1.1. Manipuladores

Son sistemas mecánicos multifuncionales, con un sencillo sistema de control, que permite gobernar el movimiento de sus elementos de los siguientes modos:

- **Manual:** Cuando el operario controla directamente la tarea del manipulador.
- **De secuencia fija:** cuando se repite, de forma invariable, el proceso de trabajo preparado previamente.
- **De secuencia variable:** Se pueden alterar algunas características de los ciclos de trabajo

Existen muchas operaciones básicas que pueden ser realizadas de forma óptima mediante manipuladores. Por ello, estos dispositivos son utilizados generalmente cuando las funciones de trabajo son sencillas y repetitivas.

Figura 1: Robot manipulador

2.1.2. Robots de repetición o aprendizaje

Son manipuladores que se limitan a repetir una secuencia de movimientos, previamente ejecutada por un operador humano, haciendo uso de un controlador manual o un dispositivo auxiliar. En este tipo de robots, el operario durante la fase de enseñanza se vale de una pistola de programación con diversos pulsadores o teclas, o bien de joysticks, o bien utiliza un maniquí, o desplaza directamente la mano del robot. Actualmente, los robots de aprendizaje son los más conocidos en algunos sectores de la industria, y el tipo de programación que incorporan recibe el nombre de "*gestual*".

Figura 2: Robot Pingüino de aprendizaje

2.1.3. Robots con control por computador

Son manipuladores o sistemas mecánicos multifuncionales, controlados por un computador, que habitualmente suele ser un microordenador. El control por computador dispone de un lenguaje específico de programación, compuesto por varias instrucciones adaptadas al hardware del robot, con las que se puede diseñar un programa de aplicación utilizando solo el ordenador. A esta programación se le denomina "*textual*" y se crea sin la intervención del manipulador.

Las grandes ventajas que ofrece este tipo de robots, hacen que se vayan imponiendo en el mercado rápidamente, lo que exige la preparación urgente de personal cualificado, capaz de desarrollar programas de control que permitan el manejo del robot.

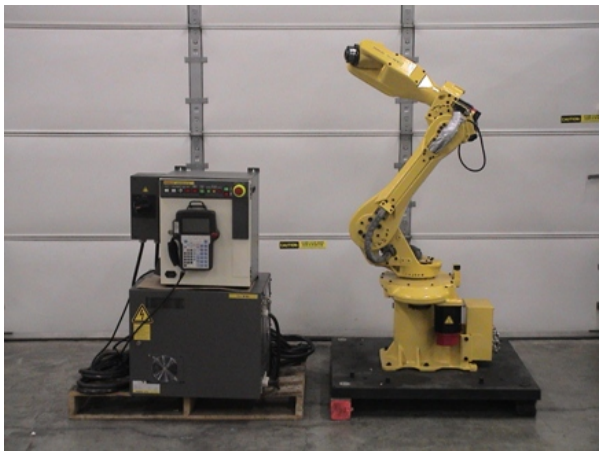

Figura 3: Robot FANUC controlado por computador

2.2. Robots inteligentes

Son similares a los del grupo anterior, pero tienen la capacidad de poder relacionarse con el mundo que les rodea a través de sensores y de tomar decisiones en función de la información obtenida en tiempo real. De momento, son muy poco conocidos en el mercado y se encuentran en fase experimental, donde grupos de investigadores se esfuerzan por hacerlos más efectivos, al mismo tiempo que más económicamente asequibles. El reconocimiento de imágenes y algunas técnicas de inteligencia artificial son los campos que más se están estudiando para su posible aplicación en estos robots.

Figura 4: Robot ASIMO de Honda

2.3. Micro-robots

Con fines educacionales, de entretenimiento o investigación, existen numerosos robots de formación o micro-robots a un precio muy asequible, cuya estructura y funcionamiento son similares a los de aplicación industrial.

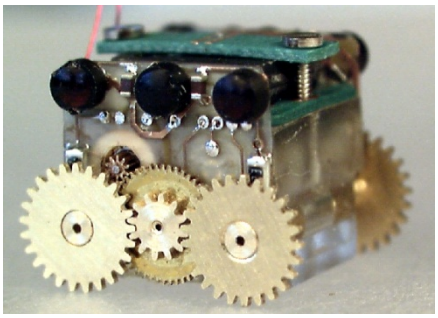

Figura 5: Robot Smoovy

Estos robots que un día se hicieron un hueco en universidades y centros de investigación, puesto que eran una forma económica de experimentar con múltiples tareas robóticas, hoy en día se pueden encontrar en centros docentes de todo tipo, incluidas escuelas de primaria e institutos. El personal de dichos centros ha apostado por este tipo de robots para estimular el interés de sus alumnos por la ciencia y la tecnología cuyo resultado es, como se ha podido observar, altamente satisfactorio.

Figura 6: Robot LEGO

2.4. Robótica móvil

En los apartados anteriores se ha realizado una definición de los diferentes tipos de robots existentes atendiendo a su aplicación, pero más allá de este aspecto práctico hay otro hecho característico de los robots modernos que les confiere un mayor grado de libertad y utilidad. Esta característica es el movimiento en el espacio físico, es decir, la posibilidad de desplazarse por el entorno para observarlo e interactuar con él, y de esta forma emular con mayor fidelidad las funciones y capacidades de los seres vivos.

Debido a la naturaleza de este proyecto se realizara con mayor profundidad un estudio de este tipo de robots.

2.4.1. Tipos de robot móvil

- **Robots rodantes**

Son aquellos que, como su nombre indica, se desplazan haciendo uso de ruedas, generalmente montadas por pares en una configuración 2+2 como las de un vehículo por mera simplicidad. Habitualmente solo dos de sus ruedas presentan tracción y las otras dos dirección, de forma que sea posible maniobrar el robot con un solo servomotor.

Figura 7: Robot rodante dotado de 4 ruedas en configuración 2+2

También es frecuente encontrar distribuciones de ruedas montadas en modo triciclo, donde una rueda sirve para la dirección y las otras dos aportan la tracción. Otra opción es que la tercera rueda simplemente sea una rueda 'loca' y las otras dos aporten tanto la tracción como la dirección, mediante el método de las orugas tratado más adelante.

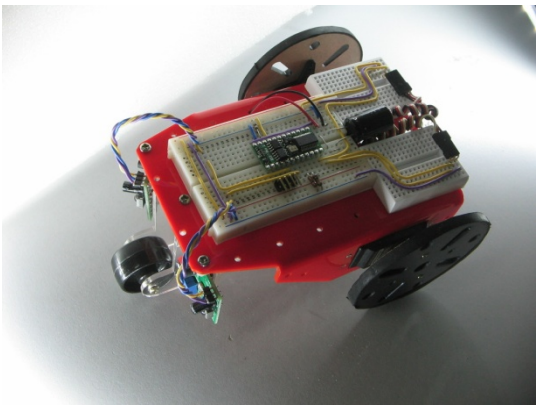

Figura 8: Robot rodante con las ruedas en configuración triciclo

Existen algunos casos especiales en los que se usan otras configuraciones que dotan al robot de mejor adaptación a terrenos difíciles. En estos casos los algoritmos de control de movimiento adquieren una mayor complejidad, proporcional al número de elementos direccionables de forma independiente.

Figura 9: Robot rodante de 6 ruedas con independencia frontal y trasera

Por último cabría considerar a los robots con orugas como un tipo de robot rodante en el que se substituyen las ruedas por un mecanismo de oruga para la tracción. La dirección se consigue parando una de las orugas o haciéndolas girar en sentido contrario.

Figura 10: Robot rodante dotado de orugas

El robot rodante es el tipo de robot móvil más común, ya que es más asequible y estable que los demás, por lo que ofrece muchas más posibilidades.

- ## Robots andantes

Respecto a los robots construidos a imagen y semejanza humana, con dos piernas, las técnicas de control necesarias son varias, pero todas ellas hacen uso de complejos algoritmos para poder mantener el equilibrio y caminar correctamente. Todos ellos son capaces de caminar bien sobre suelos planos y subir escaleras en algunos casos, pero no están preparados aun para caminar en suelos irregulares.

Algunos incluso pueden realizar tareas como bailar, luchar o practicar deportes, pero esto requiere una programación sumamente compleja que no siempre está a la altura del hardware del robot y de su capacidad de procesamiento.

Figura 11: Robot humanoide Robonova

- ## Robots reptadores

Una clase curiosa de robots, creados basándose en animales como las serpientes. Su forma de desplazarse es también una imitación de la usada por estos animales. Están

formados por un número elevado de secciones que pueden cambiar de tamaño o posición de forma independiente de las demás pero coordinada, de forma que en conjunto provoquen el desplazamiento del robot.

Figura 12: Robot reptador

- ## Robots nadadores

Estos robots son capaces de desenvolverse en el medio acuático, generalmente enfocados a tareas de exploración submarina en zonas donde no es posible llegar pos ser de difícil acceso o estar a profundidades que el cuerpo humano no tolera.

Figura 13: Robot pez

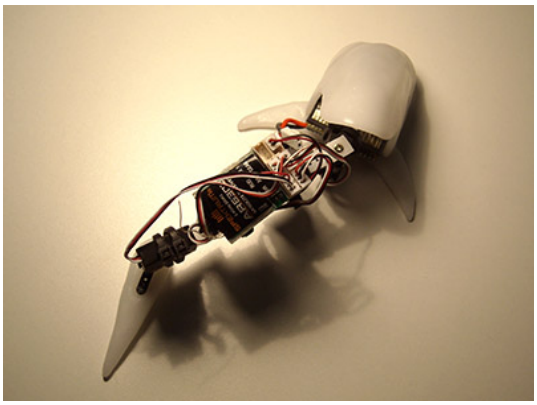

Figura 14: Robot ballena

Aparte de lo puramente anecdótico, se ha demostrado que la estructura corporal de los peces así como el movimiento que realizan durante su desplazamiento en el agua, es uno de los métodos más óptimos de movimiento submarino dado que aprovecha la energía de forma muy eficiente y permite mayor control en la navegación, produciendo mucho menos ruido y turbulencias.

- ## Robots voladores

Conquistados los dominios del mar y la tierra solo queda una meta por alcanzar en el mundo de la robótica, ser capaces de poner robots en el cielo. Para ello y por el momento existen dos aproximaciones, en función de su principio de vuelo y estructura:

- **Helicópteros:** habitualmente helicópteros RC convencionales a los que se les añade la electrónica necesaria para tener visión artificial y capacidad de toma de decisiones autónoma.

Figura 15: Helicóptero robótico

- **Drones o aviones no tripulados:** actualmente en uso por el ejército de EEUU para tareas de logística en operaciones militares, apoyo cartográfico así como tareas de espionaje. Aunque no es habitual pueden estar dotados tanto de contramedidas para repeler agresiones como de armamento para realizar ataques.

Figura 16: Drone (robot volador militar no tripulado)

2.4.2. Componentes de un robot móvil

Vistos los principales tipos de robots móviles que se construyen en la actualidad, a continuación se detallan las partes constituyentes de los robots, tanto estructurales, como mecánicas y electrónicas.

- **Estructura**

La estructura es el esqueleto, el soporte fundamental que constituye tanto la forma como la funcionalidad del robot. Sirve de sujeción para toda la electrónica, sensores, actuadores y también es parte del aparato motriz como prolongación de los actuadores. Está formada generalmente por una mezcla de partes rígidas y flexibles, fijas y móviles y entre sus materiales destacan los plásticos, metales y aleaciones resistentes a la par que ligeras como la fibra de carbono o derivados del aluminio.

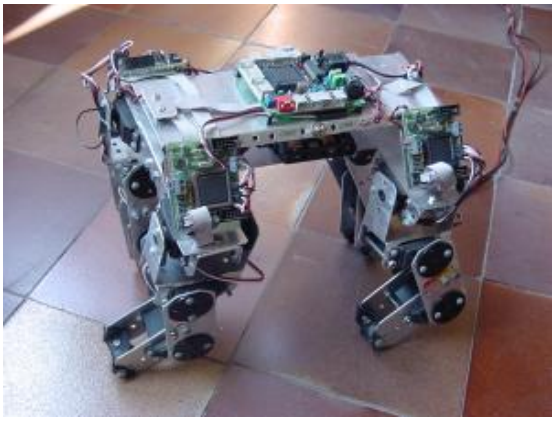

Figura 17: Estructura de aluminio para un robot cuadrúpedo

Determina el medio en el que se va a poder desenvolver el robot, así como las actividades que será capaz de realizar. También protege partes sensibles de la

electrónica de golpes, polvo, agua y otros agentes externos. Como ya se ha comentado, debe ser un compromiso entre resistencia y ligereza, adaptándose de la mejor manera posible al tipo de tareas para las que se diseña el robot que va a poseer dicha estructura.

Figura 18: Estructura de un vehículo móvil

Se debe considerar un amplio espectro de lo que puede ser una estructura para un robot, ya que aunque la vista en la figura 27 pertenece a un vehículo, este puede convertirse en robot móvil si se le añaden los suficientes servos y sensores, además de poder dotarlo de propia autonomía.

- ## Sensores

Estos elementos son los encargados de adquirir información del entorno y transmitirla a la unidad de control del robot. Una vez esta es analizada, el robot realiza la acción correspondiente a través de sus actuadores.

Los sensores constituyen el sistema de percepción del robot, es decir, facilitan la información del mundo real para que el robot la interprete. Los tipos de sensores más utilizados son los siguientes:

- **Sensor de proximidad:** Detecta la presencia de un objeto ya sea por rayos infrarrojos, por sonar, magnéticamente o de otro modo.

Figura 19: Sensores de proximidad

Normalmente se utilizan para avisar de la presencia de obstáculos y evitar colisionar con ellos. También pueden ser utilizados para lo contrario, detectar un objeto e interactuar con él.

- **Sensor de Temperatura:** Capta la temperatura del ambiente, de un objeto o de un punto determinado.

Figura 20: Sensores de temperatura

- **Sensor magnéticos:** Captan variaciones producidas en campos magnéticos externos. Se utilizan a modo de brújulas para orientación geográfica de los robots.

Figura 21: Sensores magnéticos

• **Sensores táctiles, piel robótica:** La piel robótica se trata de un conjunto de sensores de presión montados sobre una superficie flexible. Se utiliza para detectar la forma y el tamaño de los objetos que el robot manipula.

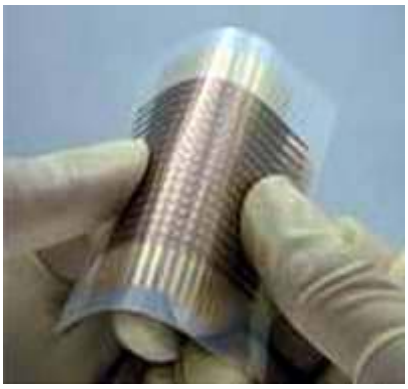

Figura 22: Sensor táctil flexible111

• **Sensores de iluminación:** Capta la intensidad luminosa, el color de los objetos, etc. Es muy útil para la identificación de objetos. Es parte de la visión artificial y en numerosas ocasiones son cámaras.

Figura 23: Sensores de luz11

• **Sensores de velocidad, de vibración (Acelerómetro) y de inclinación:** Se emplean para determinar la velocidad de actuación de las distintas partes móviles del propio robot o cuando se produce una vibración. También se detecta la inclinación a la que se encuentra el robot o una parte de él.

Figura 24: Sensores de velocidad de reluctancia variable

- **Sensores de fuerza:** Permiten controlar la presión que ejerce la mano del robot al coger un objeto.

Figura 25: Sensor de presión

- **Sensores de sonido:** Micrófonos que permiten captar sonidos del entorno.

Figura 26: Micrófono

- **Micro interruptores:** Muy utilizados para detectar finales de carrera.

Figura 27: Diferentes tipos de micro interruptores

- **Dispositivo GPS:** Dispositivo que recibe la señal de los satélites orbitando alrededor de la tierra. Los hay de distintas precisiones y en general proporcionan la posición del aparato. Se pueden utilizan para que el robot y el usuario conozcan en todo momento su posición, pero dependiendo de la calidad del dispositivo y su capacidad de procesamiento, se puede usar también para obtener otro tipo de datos a partir de ésta, como puede ser la velocidad del robot, su aceleración, dirección en la que se mueve… Toda esta información puede ser procesada y, dependiendo del resultado, los actuadores podrán reaccionar de una forma u otra.

Figura 28: Dispositivos GPS

Los dispositivos GPS utilizan el sistema SPG o GPS (Global Positioning System: sistema de posicionamiento global) o NAVSTAR-GPS. Es un sistema global de navegación por satélite (GNSS) que permite determinar en todo el mundo la posición de un objeto, una persona o un vehículo con una precisión hasta de centímetros (si se utiliza GPS diferencial), aunque lo habitual son unos pocos metros de precisión. El sistema fue desarrollado, instalado y actualmente operado por el Departamento de Defensa de los Estados Unidos.

El GPS funciona mediante una red de 24 satélites en órbita sobre el planeta tierra, a 20.200 km, con trayectorias sincronizadas para cubrir toda la superficie de la Tierra. Cuando se desea determinar la posición, el receptor que se utiliza para ello localiza automáticamente como mínimo tres satélites de la red, de los que recibe unas señales indicando la identificación y la hora del reloj de cada uno de ellos. Con base en estas señales, el aparato sincroniza el reloj del GPS y calcula el tiempo que tardan en llegar las señales al equipo, y de tal modo mide la distancia al satélite mediante "triangulación" (método de trilateración inversa), la cual se basa en determinar la distancia de cada satélite respecto al punto de medición. Conocidas las distancias, se determina fácilmente la propia posición relativa respecto a los tres satélites. Conociendo además las coordenadas o posición de cada uno de ellos por la señal que emiten, se obtiene la posición absoluta o coordenadas reales del punto de medición. También se consigue una exactitud extrema en el reloj del GPS, similar a la de los relojes atómicos que llevan a bordo cada uno de los satélites.

La antigua Unión Soviética construyó un sistema similar llamado GLONASS, ahora gestionado por la Federación Rusa.

Actualmente la Unión Europea está desarrollando su propio sistema de posicionamiento por satélite, denominado Galileo.

A su vez, la República Popular China está implementando su propio sistema de navegación, ya tienen 8 en órbita y pronto prevén tener hasta 30.

Muchos de los dispositivos GPS se pueden conectan a varios de estos sistemas, lo que reporta en un beneficio a la precisión al poder contar con mas satélites para posicionarnos.

Normalmente un dispositivo GPS cuenta con un receptor y un cable para conectarlo. El receptor recibe la información de los satélites y procesa la información para transmitirla por cable. Este puede ser serie, USB o transmitirlo incluso vía Bluetooth.

Figura 29: Receptor GPS y cable

También existen dispositivos con GPS integrado, como pueda ser móviles o tablets, actuando ellos mismos como receptores.

Figura 30: Dispositivo GPS integrado

No obstante hay dispositivos GPS que cuentan con una antena que actúa de receptor, procesándose la información en una unidad aparte. Esto ocurre para dispositivos normalmente más caros y de más potencia que es mejor tenerlos más protegidos, dejando solo visible la antena, o para aquellos que pueden tener más dispositivos, aparte de la antena, que les puedan proporcionar datos.

Figura 31: Dispositivo GPS con antena y 2 cámaras

Sobre todo a estos últimos se les puede conectar otros aparatos que proporcionen más datos. Actúan como procesadores que, al estar dedicados en exclusiva a esta tarea, pueden ser más potentes que si lo hiciésemos con un ordenador.

Existen muchos accesorios que pueden resultar útiles, aquí se detallan varios de ellos:

- **Inerciales:**

IMU: O unidad de medida inercial, contiene acelerómetros y giroscopios que indican la aceleración del accesorio y la velocidad de giro que está sufriendo dicho accesorio. Es de ayuda ya que detecta cuando el dispositivo GPS esta en movimiento y cuando no, y en caso de estarlo, hacia qué dirección va. Esto ayuda a procesar mejor las coordenadas que llegan de los satélites.

- **Módulos para señales de entrada:**

Algunos dispositivos también aceptan entradas digitales y analógicas estipuladas por el usuario, lo cual permite introducir cualquier dato que consideremos oportuno para que lo procese también o nos informe de él.

- **Dispositivos de audio y video:**

También se pueden conectar cámaras o micrófonos para registrar todo lo que se ve y oye a la vez que la posición en la que nos encontramos. Esto puede ayudar tanto a la hora de monitorizar la actividad si nos encontramos lejos del robot, o a la hora de visionar más tarde la información para ver en que situaciones se ha encontrado.

- **Enlaces de radio:**

Antenas adicionales que pueden recoger (o transmitir) otras señales aparte de las de GPS, como pueden ser de radio o Bluetooth.

DGPS: El DGPS (Differential GPS), o GPS diferencial, es un sistema que proporciona a los receptores de GPS correcciones de los datos recibidos de los satélites GPS, con el fin de proporcionar una mayor precisión en la posición calculada.

El fundamento radica en el hecho de que los errores producidos por el sistema GPS afectan por igual (o de forma muy similar) a los receptores situados próximos entre sí. Los errores están fuertemente correlacionados en los receptores próximos.

Un receptor GPS fijo en tierra (referencia) que conoce exactamente su posición basándose en otras técnicas, recibe la posición dada por el sistema GPS, y puede calcular los errores producidos por el sistema GPS, comparándola con la suya, conocida de antemano. Este receptor transmite la corrección de errores a los receptores próximos a él, y así estos pueden, a su vez, corregir también los errores producidos por el sistema dentro del área de cobertura de transmisión de señales del equipo GPS de referencia.

• Actuadores

Los actuadores son los sistemas de accionamiento que permiten el movimiento de las articulaciones del robot. Se clasifican en tres grupos, dependiendo del tipo de energía que utilicen:

- **Hidráulicos:** se utilizan para manejar cargas pesadas a una gran velocidad. Sus movimientos pueden ser suaves y rápidos.

- **Neumáticos:** son rápidos en sus respuestas, pero no soportan cargas tan pesadas como los hidráulicos.

- **Eléctricos**: son los más comunes en los robots móviles. Un ejemplo son los motores eléctricos, que permiten conseguir velocidades y precisión necesarias.

Ejemplos de actuadores son motores, relés y contadores, electro válvulas, pinzas, etc.

• Sistemas de control

El control de un robot puede realizarse de muchas maneras, pero generalmente se realiza por medio de un ordenador industrial altamente potente, también conocido como unidad de control o controlador. El controlador se encarga de almacenar y procesar la información de los diferentes componentes del robot industrial.

La definición de un sistema de control es la combinación de componentes que actúan juntos para realizar el control de un proceso. Este control se puede hacer de forma continua en todo momento o de forma discreta, o lo que es lo mismo, cada cierto tiempo. Si el sistema es continuo, el control se realiza con elementos continuos. En cambio, cuando el sistema es discreto el control se realiza con elementos digitales,

como el ordenador, por lo que hay que digitalizar los valores antes de su procesamiento y volver a convertirlos tras el procesamiento.

Existen dos tipos de sistemas, sistemas en bucle abierto y sistemas en bucle cerrado.

- **Sistemas en bucle abierto**: son aquellos en los que la salida no tiene influencia sobre la señal de entrada.

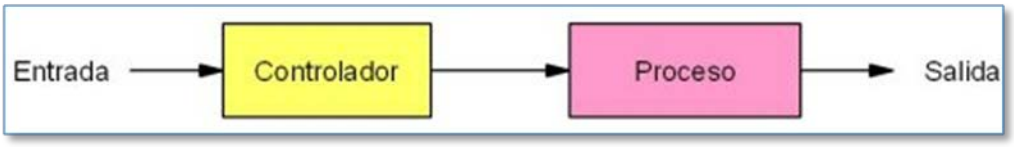

Figura 32: Esquema de un controlador en bucle abierto

- **Sistemas en bucle cerrado:** son aquellos en los que la salida influye sobre la señal de entrada.

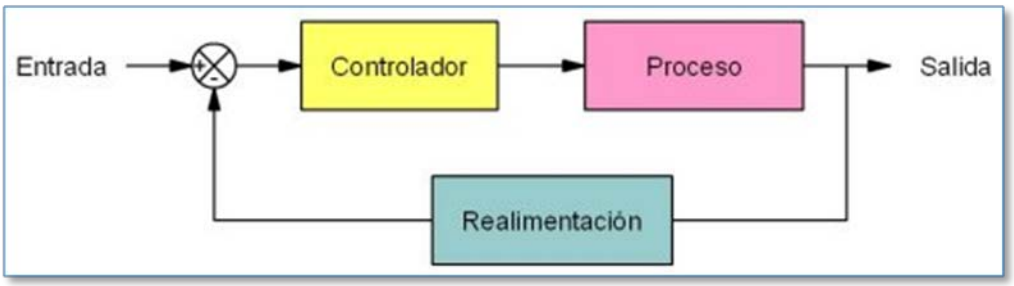

Figura 33: Esquema de un controlador en bucle cerrado

- **Sistemas discretos:** son aquellos que realizan el control cada cierto tiempo. En la actualidad se utilizan sistemas digitales para el control, siendo el ordenador el más utilizado, por su fácil programación y versatilidad.

Generalmente, el control en los robots se realiza mediante sistemas discretos en bucle cerrado, realizados por computador. El ordenador procesa la información captada por los sensores y activa los actuadores en intervalos lo más cortos posibles, del orden de milisegundos.

• Comunicaciones

El robot suele poder conectarse a un ordenador para poder configurarlo o poder obtener de éste los datos de sus sensores o sus actuadores. Las formas más comunes de comunicación son estas:

- **Serie:** Un puerto serie o puerto serial es una interfaz de comunicaciones de datos digitales, frecuentemente utilizado por computadoras y periféricos, donde la información es transmitida bit a bit enviando un solo bit a la vez.

A través de este tipo de puerto la comunicación se establece usando un protocolo de transmisión asíncrono. En este caso, se envía en primer lugar una señal inicial anterior al primer bit de cada byte, carácter o palabra codificada. Una vez enviado el código correspondiente, se envía inmediatamente una señal de stop después de cada palabra codificada.

La señal de inicio (*start*) sirve para preparar al mecanismo de recepción o receptor, la llegada y registro de un símbolo, mientras que la señal de stop sirve para predisponer al mecanismo de recepción para que tome un descanso y se prepare para la recepción del nuevo símbolo.

- **USB:** El Universal Serial Bus (bus universal en serie) o Conductor Universal en Serie (CUS), abreviado comúnmente USB, es un puerto de comunicación de periféricos a un computador. Fue creado en 1996 con la finalidad de eliminar la necesidad de adquirir tarjetas separadas para poner en los puertos de bus ISA o PCI y mejorar las capacidades plug-and-play permitiendo a esos dispositivos ser conectados o desconectados sin necesidad de reiniciar el sistema. Sin embargo, en aplicaciones donde se necesita ancho de banda para grandes transferencias de datos los buses PCI o PCIe salen ganando e igual sucede si la aplicación requiere robustez industrial.

- **WiFi:** WLAN (Wireless Local Area Network, en inglés) es un sistema de comunicación de datos inalámbrico flexible, muy utilizado como alternativa a las redes LAN cableadas o como extensión de éstas. Utiliza tecnología de radiofrecuencia que permite mayor movilidad a los usuarios al minimizar las conexiones cableadas. Las WLAN van adquiriendo importancia en muchos campos, como almacenes o para manufactura, en los que se transmite la información en tiempo real a una terminal central. También son muy populares en los hogares para compartir el acceso a Internet entre varias computadoras.

Se utilizan ondas de radio para llevar la información de un punto a otro sin necesidad de un medio físico guiado. Al hablar de ondas de radio nos referimos normalmente a portadoras de radio, sobre las que va la información, ya que realizan la función de llevar la energía a un receptor remoto. Los datos a transmitir se superponen a la portadora de radio y de este modo pueden ser extraídos exactamente en el receptor final.

Si las ondas son transmitidas a distintas frecuencias de radio, varias portadoras pueden existir en igual tiempo y espacio sin interferir entre ellas. Para extraer los datos el receptor se sitúa en una determinada frecuencia, frecuencia portadora, ignorando el resto

- **Bluetooth:** Estándar global que posibilita la transmisión de voz, imágenes y en general datos entre diferentes dispositivos en un radio de corto alcance y lo que le hace más atractivo, a muy bajo coste.

Funciona bajo radio frecuencias pudiendo atravesar diferentes obstáculos para llegar a los dispositivos que tenga a su alcance. Opera bajo la franja de frecuencias 2.4 – 2.48 GHz o como también es conocida como "Banda ISM" que significa "Industrial, Scientific and Medical" que es una banda libre usada para investigar por los tres organismos anteriores. Pero esto tiene sus consecuencias, ya que al ser libre puede ser utilizada por cualquiera se pueden crear muchas interferencias. Para evitarlas Bluetooth utiliza una técnica denominada salto de frecuencias.

El funcionamiento es ir cambiando de frecuencia y mantenerse en cada una un "slot" de tiempo para después volver a saltar a otra diferente. Es conocido que entre salto y salto el tiempo que transcurre es muy pequeño, concretamente unos 625 microsegundos con lo que al cabo de un segundo se puede haber cambiado 1600 veces de frecuencia.

Cuando coinciden más de un dispositivo Bluetooth en un mismo canal de transmisión se forma lo que se llaman "Piconets" que son redes donde hay un maestro que es el que gestiona la comunicación de la red y establece su reloj y unos esclavos que escuchan al maestro y sincronizan su reloj con el del maestro.

3. Apartado práctico

3.1. Hardware

3.1.1. VBOX 3i

El dispositivo GPS que se va a utilizar es el VBOX 3i de Racelogic. Es capaz de registrar y transmitir datos hasta 100 veces por segundo, pudiendo hacerlo por puerto USB 2.0, puerto serie y Bluetooth, además de la posibilidad de almacenar los datos en una Tarjeta Flash extraíble.
Para obtener una mayor precisión también es capaz de comunicarse con los satélites GLONASS (aparte de los satélites GPS) y de conectarse con otros dispositivos en tiempo real para obtener una mayor precisión. Entre los que usaremos se encuentra la IMU (Unidad de Medida Inercial) o el DGPS (GPS diferencial).

Figura 34: VBOX 3i

Se ha elegido dicho GPS debido en base a estas características: una latencia de muestreo muy alta y, una precisión de pocos centímetros de error gracias a los periféricos externos que se explicaran en el apartado “3.1.2 Periféricos”. Además, también ofrece otros datos como la velocidad, dirección, aceleración e inclinación entre otros, aparte de las coordenadas.

Por si solo, el VBOX 3i tiene una precisión de unos 3 metros el 95% CEP de las veces en condiciones en las que pueda tener la cobertura de suficientes satélites. 95% CEP significa que el 95% de los datos que obtengamos estarán dentro de un circulo de 3 metros de radio con centro en las coordenadas exactas de la antena.

Se explicarán varias características del modelo que se utilizarán durante el proyecto, como pueden ser la antena, los botones, la Tarjeta Flash o los puertos serie. Para conocer más características y los datos técnicos del VBOX 3i se ruega dirigirse al "Anexo A". Para instrucciones sobre su montaje se recomienda leer el "Anexo E".

- ## Antena

La antena es la que recoge los datos de los satélites y se los envía al VBOX 3i para su procesamiento. Hay que asegurarse de colocarla en el punto más alto del vehículo para evitar el mayor número de obstáculos.

Cuenta con un imán situado en su base que, además de ser útil para colocarlo en superficies planas y metálicas (como el techo de un vehículo), crea un campo que ayuda a evitar recibir señales reflejadas en objetos cercanos, lo cual reduciría la precisión.

Figura 35: Antena GPS

El led **SATS** situado en el panel frontal nos informa el número de satélites que está siguiendo en ese momento.

- Rojo: Si no encuentra ningún satélite.
- Verde: Ha encontrado tantos satélites GPS como parpadeos seguidos haya.
- Amarillo: Ha encontrado tantos satélites GLONASS como parpadeos seguidos haya.

Habrá una pausa más larga entre cada secuencia.

Estos ejemplos están ilustrados en la figura 36:

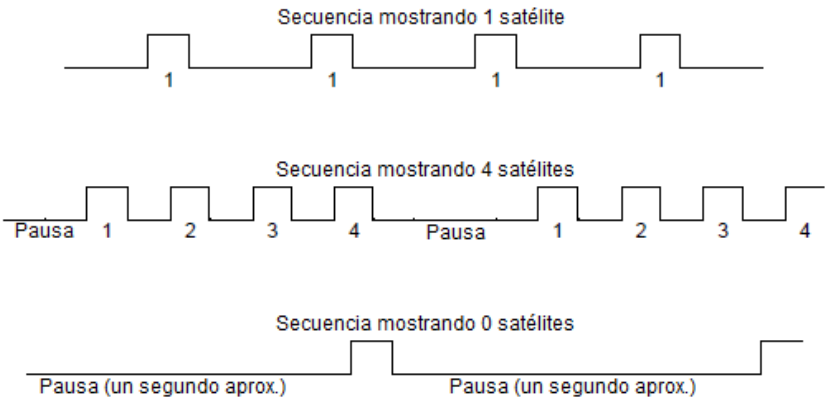

Figura 36: Ejemplos de secuencias de número de satélites.

Las nubes no obstruirán la señal de los satélites, no obstante si lo harán los árboles y túneles, por lo que en dichos momentos no se podrán obtener datos de nuestra posición. Para ello nos servirán de ayuda otros dispositivos como la IMU, que se explicara más adelante en el apartado “3.1.2 Periféricos - IMU”.

- ## Botones

Hay dos botones en el panel frontal del VBOX 3i: **LOG** y **FUNC**.

Figura 37: Botones **LOG** y **FUNC**

- **LOG**: Se utiliza para iniciar y detener el registro de datos en la Tarjeta Flash. Cada vez que se empiece un registro se generara un nuevo archivo. Cuando hay información registrándose en la tarjeta, el led azul **CF** parpadea. No es recomendable apagar o quitar la tarjeta flash mientras este led esta parpadeando.
- **FUNC**: Si se mantiene apretado 5 segundos, se utiliza para cambiar entre los periodos de muestreo de 20Hz (transmite información una vez cada cinco centésimas de segundo) y 100Hz (una vez cada centésima de segundo). Si se aprieta y suelta inmediatamente, muestra el periodo de muestreo actual con la ayuda del led. Un parpadeo por segundo significa 20Hz, cinco parpadeos significa 100Hz, y un parpadeo constante significa que tiene otra configuración. Igualmente se puede cambiar desde el programa VBOXTools, lo cual se explicará en el apartado “3.2.1 VBOXTools - Configuración”.

- **LOG**+**FUNC**: Si se mantienen apretados ambos a la vez durante cinco segundos, se resetea el equipo y se vuelve a la configuración de fábrica.

• Tarjeta flash

VBOX 3i acepta tarjetas Compact Flash Tipo I como la de la figura 38 para registrar los datos. La información se almacena en formato de fichero de texto ASCII y puede ser cargada directamente en el VBOXTools, (explicado en apartado “3.2.1 VBOXTools – Post-procesamiento”) o en cualquier otro programa, como el Excel.

Figura 38: Tarjeta flash.

Dicha tarjeta se puede insertar cuando apetezca, tanto si el aparato está apagado o encendido, comenzando a registrar datos inmediatamente. No obstante, para retirarla debemos esperar a que el led **CF** deje de parpadear, de lo contrario, el fichero en el que se está escribiendo podría no cerrarse correctamente y perder los datos. Deberemos pulsar el botón **LOG** para que cierre el fichero y deje de recoger datos antes de retirar la tarjeta (visto en el apartado anterior “Botones”). Se puede dar que el GPS esté configurado para que solo registre datos cuando se está moviendo (Explicado en el apartado “3.2.1 VBOXTools – Configuración”), en tal caso si se está parado también se apagara el led **CF** y se podrá retirar la tarjeta.

• Puertos serie

El VBOX 3i está equipado con dos buses CAN y dos puertos serie RS232.

El puerto primario se utiliza para la comunicación entre el VBOX y el ordenador y está preparado para transmitir datos en tiempo real. Esta marcado como **SER** en el panel frontal.

El puerto secundario se utiliza para la conexión con otros periféricos, como la IMU o la antena del DGPS detallados en el apartado “3.1.2 Periféricos”. Esta marcado como **CAN** en el panel frontal.

Ambos puertos son intercambiables configurándolos con el programa VBOXTools (Explicado en la página 23 del “Anexo D”).

- **Datos y su transmisión**

El VBOX 3i tiene una latencia de transferencia de datos de 100Hz tanto a la hora de grabarlos en la tarjeta flash como de transmitirlos por USB y Bluetooth a un ordenador.

El contenido de la tabla de la figura 39 será enviado 100 veces por segundo. Esto no es del todo exacto ya que se puede transmitir a una latencia menor o contener menos campos de datos si el usuario así lo especifica (Explicado en el apartado "3.2.1 VBOXTools - Configuración").
En caso de agregar algún periférico adicional, puede que se añadan nuevos campos a la tabla de datos, los cuales serán especificados en sus respectivos apartados (Ver "3.1.2 Periféricos").

Datos transmitidos en tiempo real (VBOX 3i)
Número de satélites
Tiempo
Latitud
Longitud
Velocidad
Dirección
Altura
Momento de inicio de evento
Velocidad vertical
Aceleración en la longitud
Aceleración en la latitud
Número de satélites GLONASS
Número de satélites GPS
Calidad de la velocidad
Tipo de solución
Estado del filtro Kalman
Datos recibidos por **CAN**

Figura 39: Datos transmitidos en tiempo real (Tarjeta Flash, USB, Bluetooth)

Sin embargo, si se utiliza el cable serie para la transmisión de datos, se dará el caso de que solo se transmitirán unos pocos campos. Esto se debe a las limitaciones del puerto serie, el cual no es capaz de transmitir tanta información en un periodo de tiempo tan corto. Habrá que configurar el envío de datos a 20Hz (20 veces por segundo) o menor para que pueda enviar todos los datos. También hay que usar esta frecuencia en el caso de querer recibir datos de la IMU por ejemplo. Se tiene la desventaja de que se obtendrá la información en un intervalo mayor de tiempo (Ver el apartado "3.2.1

VBOXTools - Configuración para saber cómo cambiar la frecuencia del puerto serie). En la tabla de la figura 40 muestra que campos se pueden enviar según la frecuencia de transmisión.

100Hz	**50Hz**	**20Hz**
Número de satélites	Número de satélites	Número de satélites
Tiempo	Tiempo	Tiempo
Velocidad	Velocidad	Velocidad
Momento de inicio de evento	Momento de inicio de evento	Momento de inicio de evento
	Latitud	Latitud
	Longitud	Longitud
	Dirección	Dirección
	Altura	Altura
		Velocidad vertical
		Aceleración en la longitud
		Aceleración en la latitud
		Número de satélites GLONASS
		Número de satélites GPS
		Calidad de la velocidad
		Tipo de solución
		Estado del filtro Kalman
		Datos recibidos por **CAN**

Figura 40: Datos transmitidos en tiempo real (Serie)

En este proyecto se utilizará la conexión vía puerto serie. Se recomienda utilizarlo a 20Hz y la tarjeta flash a 100Hz.

Todos estos datos son transmitidos por el GPS de forma ordenada, y según cuántos y cuáles sean, la traza (información que transmite el VBOX 3i en un periodo de muestreo) tendrá un tamaño determinado de bytes. La más larga que se puede crear (con tan solo las lecturas del VBOX 3i, sin ningún periférico adicional) es de 76 bytes. Si hay algún campo que hemos configurado para que no se transmita, dichos bytes desaparecerán, y la traza se acortaría proporcionalmente, pudiendo llegar a ser como mínimo de 60 bytes. El único caso en que la traza puede ser menor de 60 bytes es utilizando el cable serie a 100Hz, con lo que ésta se reduciría a tan solo 47 bytes. En la figura 41 vemos el esquema de cómo estaría formada la traza más larga.

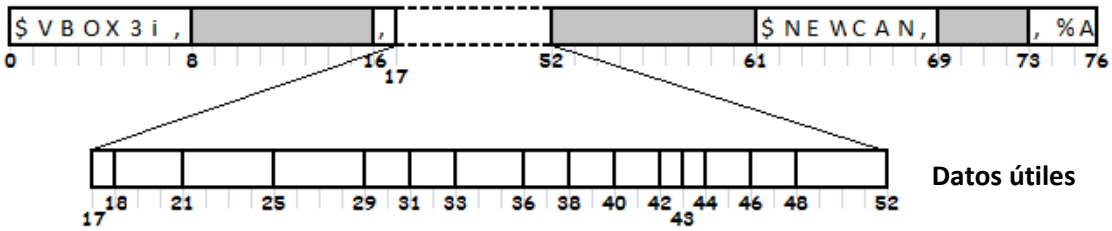

Figura 41: Esquema de una traza completa

Los bytes comprendidos entre los números 0-17 y entre 52-76 son cabeceras (las zonas con caracteres) y espacios de uso interno del programa (las zonas grises) los cuales se encuentran en toda traza que transmita el VBOX 3i sin variar en número.

Los bytes que se encuentran entre 17-52 son los datos que nos indican la localización y situación del dispositivo. En la figura 42 encontramos una tabla con una descripción sobre cada campo de esta cadena.

Nombre	Descripción	Tamaño en bytes	Posición en la traza
Número de satélites	El número de satélites que tiene localizados y usa en ese momento.	1	17-18
Tiempo	Hora en ese instante. Empezará a contar desde 0 cuando se encienda, y se pondrá en hora cuando encuentre algún satélite.	3	18-21
Latitud	La latitud en la que se encuentra la antena del VBOX 3i.	4	21-25
Longitud	La longitud en la que se encuentra la antena del VBOX 3i.	4	25-29
Velocidad	Velocidad a la que se mueve la antena del VBOX 3i. Se calcula a partir de la latitud y la longitud.	2	29-31
Dirección	Dirección en la que se mueve la antena del VBOX 3i. Se calcula a partir de la latitud y la longitud.	2	31-33
Altura	Altura sobre el nivel del mar en que se encuentra la antena del VBOX 3i. No es muy preciso.	3	33-36
Velocidad vertical	Velocidad a la que se mueve en vertical la antena del VBOX 3i. Se calcula a partir de la altura.	2	36-38
Aceleración en la longitud	Aceleración en las coordenadas de longitud.	2	38-40
Aceleración en la latitud	Aceleración en las coordenadas de latitud.	2	40-42
Número de satélites GLONASS	El número de satélites GLONASS que tiene localizados y usa en ese momento.	1	42-43

Número de satélites GPS	El número de satélites GPS que tiene localizados y usa en ese momento.	1	43-44
Estado del filtro Kalman	Número que indica el estado del filtro Kalman.	2	44-46
Tipo de solución	No data (no se reciben datos) = -1 No solution (no se encuentran satélites) = 0 Stand alone (todo correcto) = 1 Stand alone (con el DGPS on) >= 3	2	46-48
Calidad de la velocidad	Varía cuando se produce error	4	48-52

Figura 42: Campos de datos que se encuentran en la traza.

Los datos se encuentran en formato complemento a dos.

Los campos “Número de satélites”, “Tiempo”, “Latitud”, “Longitud”, “Velocidad”, “Dirección” y “Altura” se transmitirán siempre, haciendo llegar la traza a los 60 bytes mínimos. Sin embargo solo “Número de satélites”, “Tiempo” y “Velocidad” se transmitirán si la frecuencia de muestreo se configura para que sea de 100Hz usando un cable serie.

Otros datos que se pueden configurar para ser transmitidos como el momento de inicio de evento, momento de frenado, estado del DGPS o la memoria usada, se conocen a partir de otros datos de la traza o mediante cálculos analizados en tiempo real y no afectan a la longitud de ésta.

3.1.2 Periféricos

El VBOX 3i está diseñado para poder utilizar y aprovechar distintos dispositivos que se conecten a él, ya sea para mejorar su precisión o proporcionar nuevos datos, todo ello procesado y transmitido por el VBOX en tiempo real.

Por su utilidad y disponibilidad, se estudiará como se mejora la precisión gracias a dos dispositivos con los que contamos, la IMU y el DGPS. Se podrían usar ambos a la vez pero sería necesario un hardware que no tenemos (un adaptador de dos puertos series en uno) por lo que los estudiaremos por separado.

- **IMU**

Unidad de Medida Inercial (IMU), instrumento con acelerómetros y giroscopios dentro que son capaces de ofrecernos datos adicionales sin depender de los satélites, tales como la aceleración, la inclinación y la temperatura.

Figura 43: IMU (Unidad de Medida Inercial)

Gracias a ella, datos que ofrecía el propio VBOX se pueden combinar y dar resultados más precisos ya que suaviza en gran medida el error. La información que es más exacta gracias a la IMU es la velocidad, la posición, la aceleración en la latitud y la longitud y la velocidad vertical.
Otra ventaja importante es la posibilidad de seguir indicando la posición del vehículo en casos donde se ha perdido la localización de los satélites, como pueda ser en un túnel o una arboleda, durante un máximo de 10 segundos, ya que la va calculando en función de los giros y aceleraciones que registra.

No obstante tiene un error si se utiliza sin los satélites para localizarse y corregirse, por lo que tras 10 segundos, el error acumulado hace que la información de la localización deje de ser válida, aunque se puede usar de forma orientativa.

La IMU deberá situarse en la posición más céntrica y plana del vehículo que sea posible para una captación optima.

Ahora se explicará cómo cambia la traza que envía el VBOX3i debido a la nueva información de la IMU. Para conocer más características y los datos técnicos de la IMU se ruega dirigirse al “Anexo B” para más información.

También hay una guía de montaje en el “Anexo E”.

o Datos y su transmisión

La IMU también transmite datos a 100Hz, lo que hace que el VBOX pueda utilizar esta información en cada periodo de muestreo para mejorar los datos recogidos por el mismo. En la figura 44 se puede observar la nueva traza más larga:

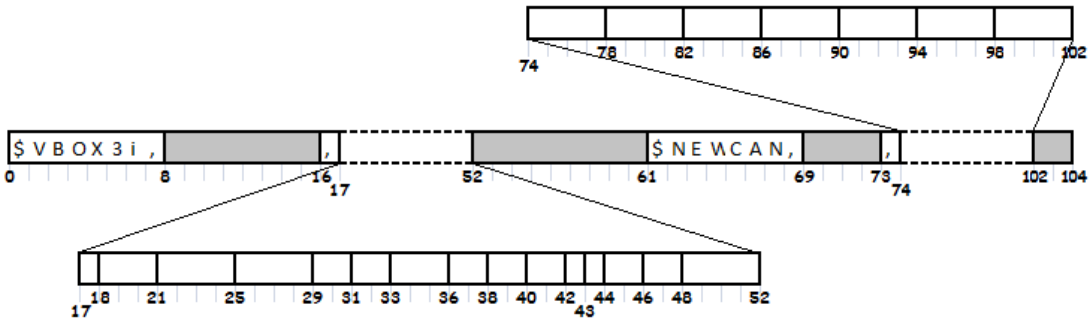

Figura 44: Esquema de una traza completa con IMU

Los bytes entre 74-102 contendrán todos los datos enviados por el VBOX 3i con respecto a la IMU. Si decidimos no mandar alguno de los datos, simplemente deberemos quitar su campo de la traza. En caso de no querer mostrar ninguno de los campos de la IMU, la traza será igual a la que devolvería si la IMU no estuviese conectada.

A continuación tenemos la figura 45, una tabla con la posición y descripción de los campos de la IMU:

Nombre	Descripción	Tamaño en bytes	Posición en la traza
Yaw	Dirección a la que mira la IMU	4	74-78
Aceleración X	Aceleración en la coordenada X de la IMU	4	78-82
Aceleración Y	Aceleración en la coordenada Y de la IMU	4	82-86
Temperatura	Temperatura de la IMU	4	86-90
Pitch	Grado de inclinación delantera y trasera	4	90-94
Roll	Grado de inclinación lateral	4	94-98
Aceleración Z	Aceleración en la coordenada Z de la IMU	4	98-102

Figura 45: Campos de datos de la IMU que se encuentran en la traza.

Estos datos se encuentran en este caso en formato IEEE coma flotante.

Hay que recalcar que los datos de “Latitud”, “Longitud”, “Velocidad”, “Aceleración en latitud”, “Aceleración en longitud” y “Velocidad vertical” que se encuentran en 17-52 están ya corregidos con respecto a la información de la IMU.

- ## DGPS

El DGPS o GPS diferencial consiste en una estación base con una antena GPS conectados a un transmisor de radio, cuya única finalidad es conocer su propia posición y con ello corregir las de los demás GPS que le consultan, consiguiendo una precisión enorme.

Figura 46: Estación base del DGPS

La estación base y la antena GPS se colocan en un punto fijo, manteniéndose ahí un tiempo suficiente como para calcular sus coordenadas exactas. Esto lo consigue almacenando todos los datos que le envían los satélites y calculando el centro de todas las coordenadas dadas.

Una vez conocido, seguirá pidiendo su posición a los satélites, pero ésta no será la misma que la calculada, esta diferencia será el error. Dicho error será muy similar al que tendrá el VBOX 3i, por lo que sabiendo el DGPS de que error se trata, le transmitirá al VBOX como corregirlo. (Al VBOX habrá que colocarle otra antena en el puerto serie **CAN** para que pueda recibir la información). En la figura 47 vemos un esquema de cómo funciona.

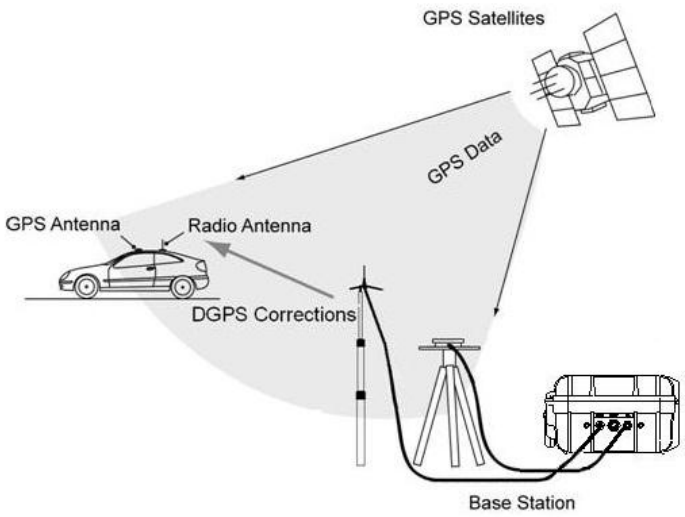

Figura 47: Esquema del funcionamiento de un DGPS

Para conocer una descripción mejor de cómo funciona, se recomienda echar un vistazo a la sección de GPS del apartado “2.4.2 Componentes de un robot móvil – Sensores”.

La estación base cuenta con una batería de varios días por lo que es recomendable dejarla en un punto fijo para así ofrecer correctamente una corrección al error. La antena de transmisión deberá estar visible por el VBOX para que le llegue la señal, por lo que es recomendable montarla en una posición elevada y en el centro de la zona que se vaya a utilizar.

El transmisor se encuentra a mitad del mástil, lo que consigue que la señal sea más fuerte y le afecte menos el ruido. Dispone de una caja cerrada para colocarlo y protegerlo de las inclemencias del tiempo.

Figura 48: Antena de radio y transmisor.

La estación base puede tener varias configuraciones de frecuencias a utilizar y puede guardar varias posiciones, para configurarlo correctamente con el equipo con el que contamos, habrá que seleccionar la configuración indicada en el recuadro:

Configuración DGPS
Handshaking: ON
DGPS Baud: 9600
Radio mode: User
Set DGPS Mode: Racelogic 2cm

También hay una guía de montaje, funcionamiento y configuración en el “Anexo E”.

Debido a que el DGPS solo transmite una corrección al VBOX 3i y ningún dato nuevo, la traza permanecerá invariable.

Para conocer más características y los datos técnicos del DGPS se ruega dirigirse al “anexo C” para más información.

3.2 Software

Para poder ver los datos que transmite el VBOX 3i en tiempo real no se puede contar con la Tarjeta Flash, ya que sus datos solo se podrán ver después de desconectarla del GPS, por ello contamos con dos programas que conectando el ordenador al GPS mediante un cable serie, un cable USB o por Bluetooth, nos muestran los datos que va procesando en tiempo real. Estos programas son el VBOXTools, que ya venía con el VBOX 3i en un CD, o un programa realizado expresamente para este proyecto, solo compatible con el VBOX 3i.

Se vio la necesidad de hacer el segundo programa ya que el proporcionado por Racelogic no transmitía los datos a otros programas, necesitando nosotros precisamente uno que lo hiciera, con el objetivo de poder hacer que un vehículo móvil pueda auto gobernarse dependiendo de la información que el GPS nos proporcione.

Además, VBOXTools solo funciona en Windows y es probable que se utilice otro sistema operativo como pueda ser Linux, por lo que se ha escrito el programa para que pueda correr en éste último.

No obstante se seguirá necesitando el VBOXTools para configurar el VBOX 3i si hace falta añadir o quitar algún periférico o cambiar algún parámetro.

3.2.1 VBOXTools

Programa proporcionado por Racelogic con distintos usos, aunque todos relacionados con el tratamiento de información de sus GPS. Se puede descargar desde http://vboxtool.sourceforge.net/. Con él podemos configurar el dispositivo, monitorizar la información transmitida en tiempo real por el VBOX 3i, almacenarla y procesar datos de GPS ya sean propios o de otros.

En la figura 49 se puede ver la ventana del programa mientras se está realizando un post-procesamiento de unos datos recogidos previamente.

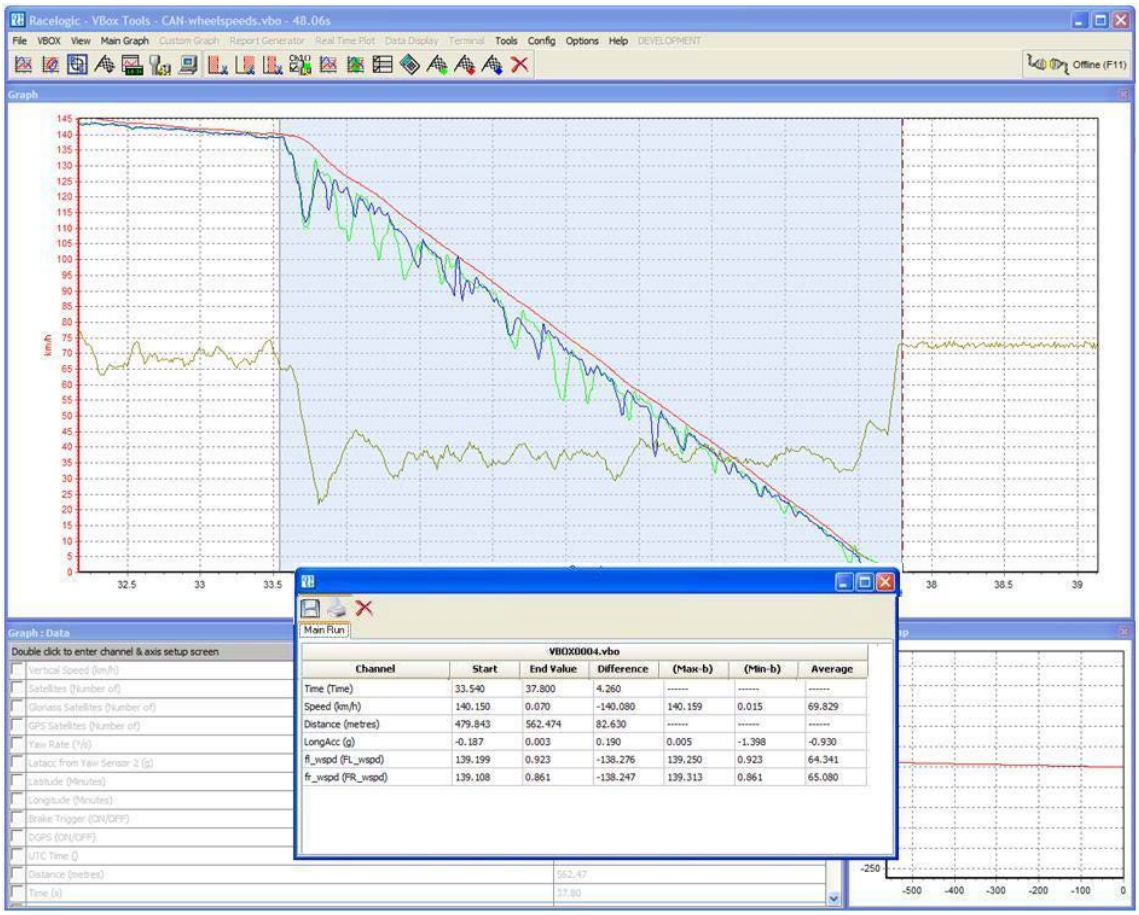

Figura 49: VBOXTools

Aquí se explicarán las opciones más útiles para nosotros, teniendo una explicación más profunda en el “Anexo D”.

- **Monitorización**

Con el programa podremos ver en tiempo real lo que está transmitiendo el VBOX 3i. Para poder monitorizar los datos que está recibiendo obviamente este deberá estar conectado al ordenador de alguna forma. Se ha experimentado con el cable serie y con el USB y con ambos el uso del programa es el mismo.

Para enlazar el programa con el dispositivo, basta con pulsar la tecla F11 oel botón de la figura 50, que nos informa de que estamos en modo desconectado.

Figura 50: Botón para comenzar la comunicación con el VBOX 3i

En caso de querer desconectarlo, se pulsará el mismo botón o la tecla F11, aunque ahora el botón lucirá como muestra la figura 51, mostrando que estamos en modo conectado.

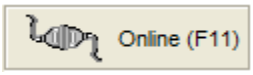

Figura 51: Botón para terminar la comunicación con el VBOX 3i

En caso de ser la primera vez que conectamos el cable o al haberlo puesto en un puerto que no sea el último usado, nos saldrá un mensaje de que el puerto COM no está disponible. Para seleccionar uno, hay que ir al menú Options -> COM Port y ahí seleccionamos el que hayamos conectado el GPS.

Una vez ya conectados, el programa nos muestra pequeñas ventanas, cada una con un dato, como por ejemplo el número de satélites o la velocidad.

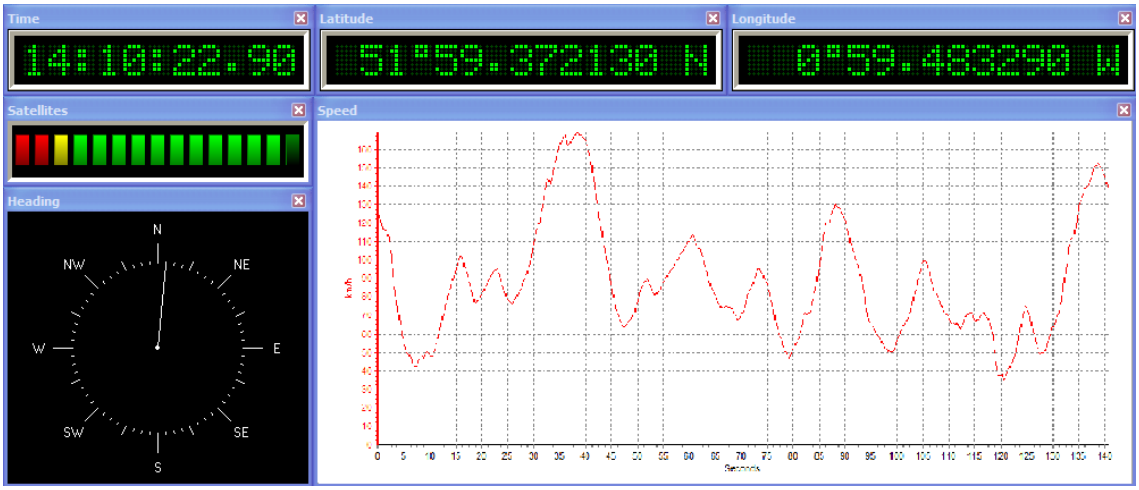

Figura 52: Ventanas con diferentes tipos de información

Si se quiere cambiar la información que muestra alguna de estas ventanas, basta con clicar con el botón derecho sobre ellas y elegir la información que queremos dentro de “Main Data”.

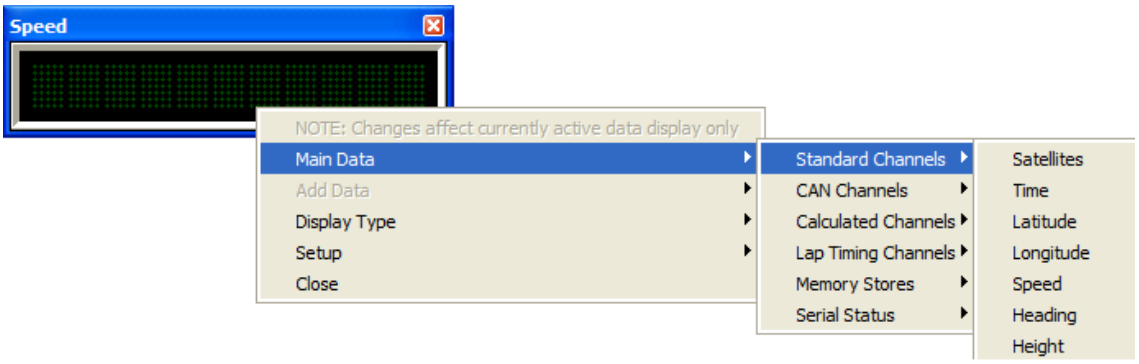

Figura 53: Como elegir que datos mostrar

Si preferimos cambiar la forma en que se muestren los datos, se debe clicar con el botón derecho sobre la ventana y elegir esta vez el menú “Display Type”.

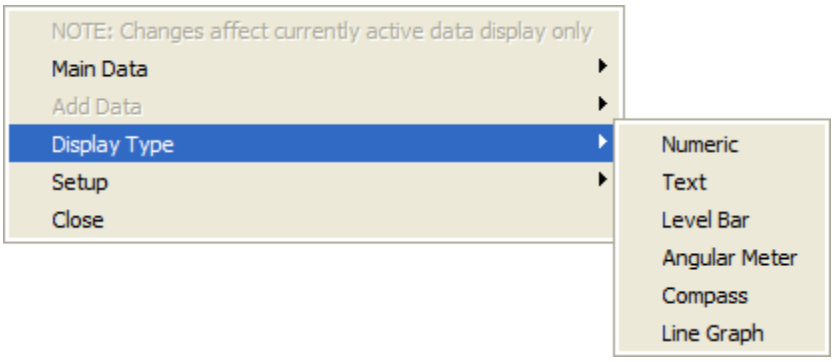

Figura 54: Cambiar de formato de visualización

Otras opciones que tenemos para monitorizar mejor la información que nos ofrece el VBOX nos las proporcionan estos botones de la figura 55:

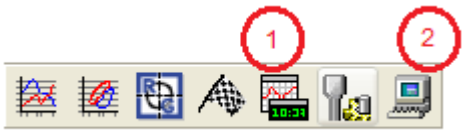

Figura 55: Botones importantes

Donde 1 crea una nueva ventana de datos y con 2 abre una ventana donde se puede ver la traza devuelta por el GPS directamente, tanto en ASCII como en hexadecimal. Este último punto es muy útil para poder estudiar su estructura y poder utilizarla en otros programas.

- ## Almacenamiento

Al estar en modo conectado (visto en el apartado anterior “Monitorizar”) aparece también una ventana con la opción a guardar la información que se recibe. Esto puede resultar útil si no contamos con una Tarjeta Flash.

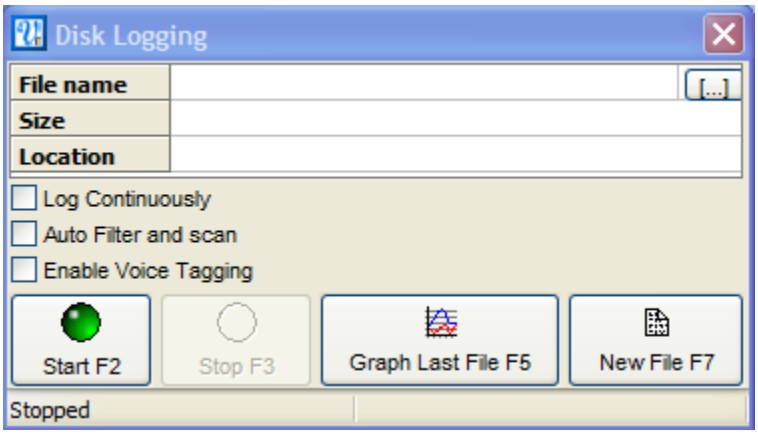

Figura 56: Ventana para guardar la información

Con darle al botón Start o a F2 ya empezaría a guardar toda la información recibida.

Se da la opción de darle un nombre especifico (por defecto se nombrará vblog_001.vbo o sucesivos en la numeración si ya existe un fichero con este nombre) al pulsar el botón […] y en los recuadros podemos elegir si deseamos que guarde información continuamente (por defecto solo guarda cuando el programa detecta que hay movimiento), aplicar un filtro para suavizar un poco el posible error que se cometa y si queremos habilitar la grabación de sonido para añadir notas que vayamos comentando a lo largo de la grabación.

- # Configuración

Para configurar el VBOX 3i deberemos ir a la ventana de Setup. Mientras está esté abierta el resto de ventanas permanecerán cerradas. Para acceder al menú Setup pulsaremos el botón 1 de la figura 57 o iremos al menú correspondiente.

Figura 57: Como acceder a la página de configuración

Lo primero que veremos al aparecer la ventana es el menú Channels (figura 58):

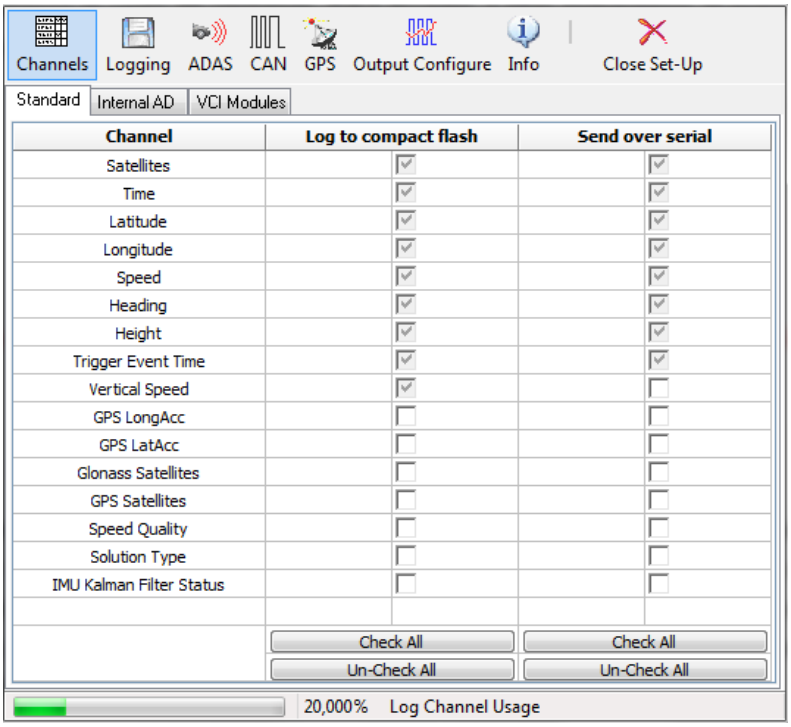

Figura 58: Channels

Channels:

En este menú se puede ver todos los datos que el VBOX 3i envía a la Tarjeta Flash y al ordenador a través del cable del puerto serie, pudiendo elegir si enviar todos ellos o solo unos cuantos. Si utilizamos el cable USB o el Bluetooth, todos los datos serán enviados. Abajo hay una barra sobre lo ocupado que esta el canal de envío de datos a la Tarjeta Flash, actuando como límite, ya que si lo sobrepasamos el GPS no tendrá tiempo suficiente para enviar toda la información en cada iteración.

En la pestaña Standard podemos encontrar todos los datos propios del VBOX 3i sin añadirle ningún periférico. Los que no se pueden quitar se debe a que con la información que transmiten es posible sacar otros datos adicionales e importantes, como por ejemplo, la latitud y longitud, necesarios para dibujar el recorrido del vehículo, o la dirección para calcular la aceleración lateral.

Las pestañas Internal AD y VCI Modules son para configurar canales de entrada análogos y canales CAN externos, los cuales no usaremos.

Si el IMU está conectado veremos otra pestaña:

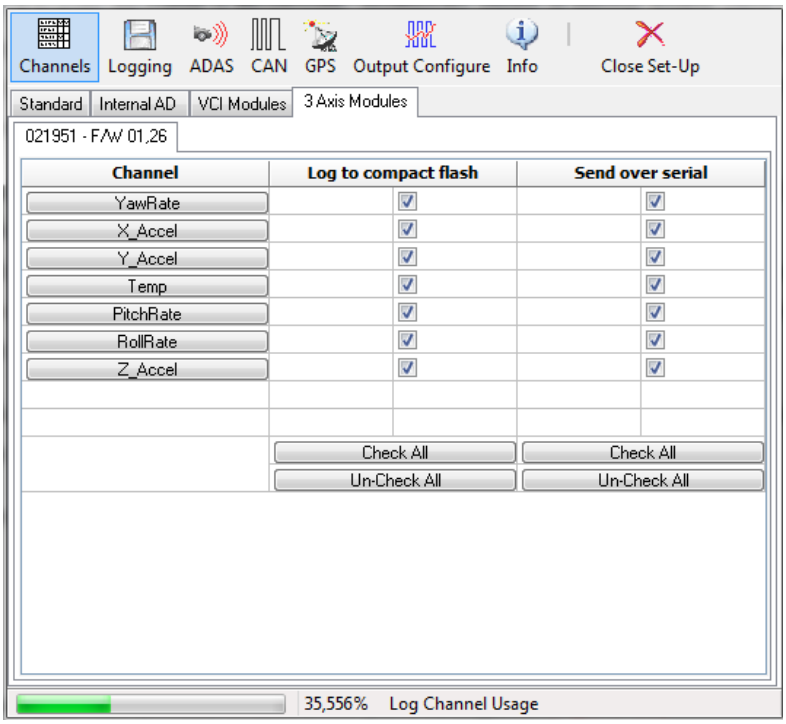

Figura 59: Pestaña de la IMU en el menú Channels

Con todos los campos seleccionados también mandaremos los datos que recoge la IMU a la Tarjeta Flash y por el cable serie. Es recomendable seleccionarlos todos ya que a veces el programa falla y envía otros datos diferentes a los seleccionados.

Logging:

En este apartado nos ofrecen 3 secciones para configurar:

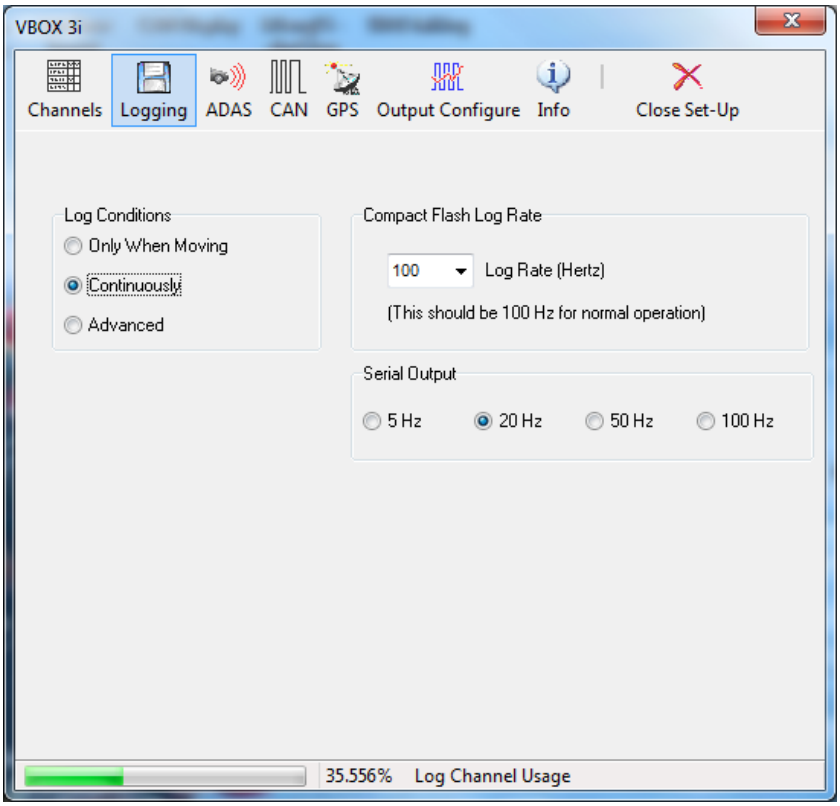

Figura 60: Logging

-**Log Conditions**: Cuando registrar la información del GPS. Tenemos la opción de hacerlo solo cuando detecta que se está moviendo (que posea una velocidad mayor de 0.5 km/h), que lo haga siempre, o avanzado, que registre datos solo cuando se cumpla cierta condición, como estar recibiendo datos de más de 5 satélites.

-**Compact Flash Log Rate**: Frecuencia de registro en la Tarjeta Flash. Se puede poner a 100Hz para obtener una muestra cada centésima de segundo sin problemas.

-**Serial Output**: Frecuencia de transferencia de datos por el puerto serie. Si se pone a 100Hz solo transmitirá los datos del número de satélites, el tiempo, la velocidad y el momento de inicio de evento. En caso de ponerlo a 50Hz, podrá transmitir todos los datos de la pestaña Standard que no se pueden deseleccionar. Si se pone a 20Hz podrá enviarla toda y también la información de los periféricos conectados, como pueda ser la IMU. La figura 42 del apartado “3.1.1 VBOX 3i – Datos y su transferencia” ilustra bien estas restricciones.

GPS:

Aquí configuraremos el uso de la **IMU** y el **DGPS** tanto como ciertos filtros del VBOX 3i:

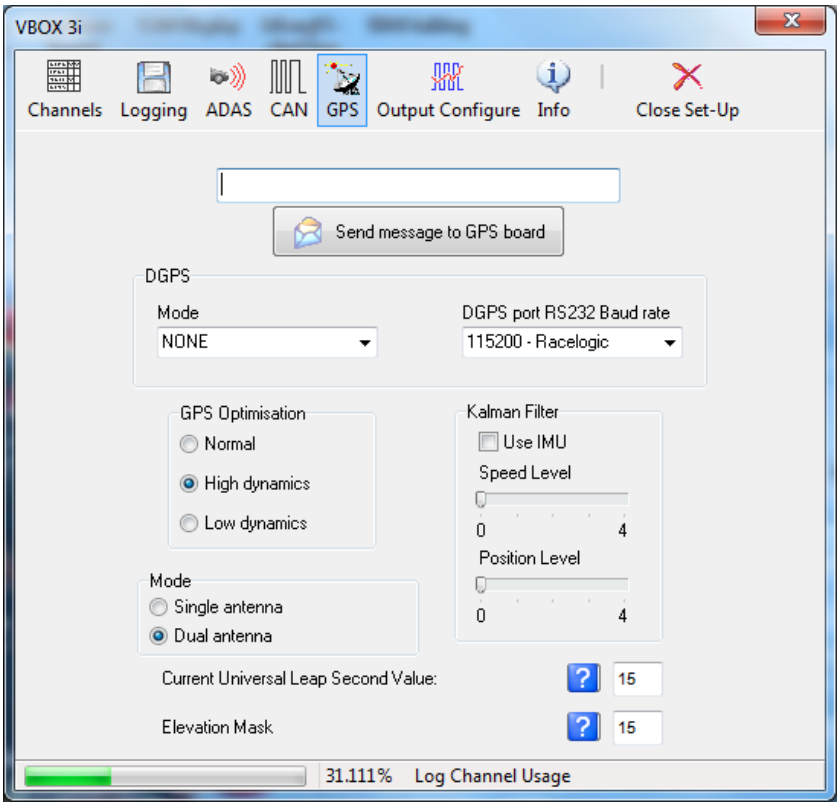

Figura 61: GPS

Nos centramos en el recuadro “Kalman Filter”. Las barras Speed Level y Position Level sirven para configurar un filtro que suaviza en cierta manera el error de precisión de utilizar un GPS sin IMU y arregla un poco las perdidas completas de cobertura de los satélites. Si seleccionamos “**Use IMU**” con la **IMU** conectada, se pasará a utilizar sus datos en tiempo real para corregir nuestra posición obteniendo más precisión.

En el recuadro DGPS nos servirá informar al VBOX de que estamos utilizando uno de estos aparatos, y hacer que corrija los datos gracias a él. Para que funcione correctamente con el que tenemos, deberemos seleccionar los indicados en el recuadro:

Configuración DGPS
Mode: Racelogic (2cm RTK)
DGPS port RS232 Baud Rate: 115200 - Racelogic

- # Post-procesamiento

Si poseemos un archivo .VBO tanto si lo hemos bajado de internet como si lo hemos sacado de la tarjeta Flash o gracias al VBOXTools, con el programa podremos abrirlo y analizar los datos de la sesión.

Para abrirlo basta con clicar sobre el archivo o ir al menú File -> Load.

El fichero se cargará pero puede que no veamos nada. Para acceder a las gráficas y demás datos deberemos pulsar el botón 1 de la figura 62.

Figura 62: Como acceder a las gráficas.

Una vez cargada la gráfica, se podrá mostrar unos datos u otros seleccionándolos de la lista que habrá aparecido abajo.

Si se clica dos veces sobre ella o se pulsa el botón Channel and Axis Setup aparecerá una ventana donde podremos configurar que datos se pueden ver, aplicarles distintos colores y mostrar sus rangos en un lateral de la gráfica.

Otra herramienta útil es el botón Measure Tool, a la izquierda de Channel and Axis Setup. Aquí se seleccionaran dos puntos de la gráfica y el programa calculara varios valores extraídos de las gráficas que se están mostrando en ese momento, entre ellos, la diferencia y la media de cada uno.

También aparecerá en la parte de abajo a la derecha una gráfica como la de la figura 63 que muestra el recorrido, según los datos de latitud y longitud, que ha realizado el GPS

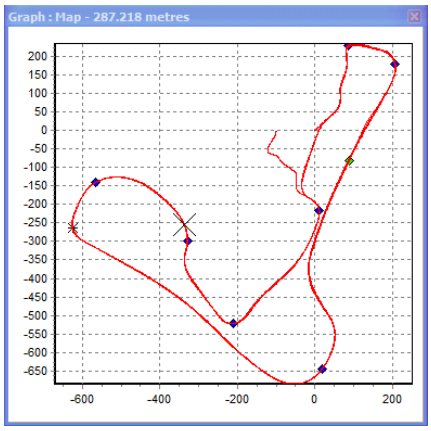

Figura 63: Recorrido del GPS

3.2.2 Librería Linux

Esta librería ha sido creada ante la necesidad de una forma simple que permita recoger los datos del GPS en tiempo real desde otros programas y no solo el poder monitorizarlo desde el VBOXTools. Se compone de los archivos gps.c y gps.h.

Debido a que aún no se conoce exactamente el entorno en el que se utilizará y las capacidades tanto del hardware como del software, se ha optado por escribirlo en lenguaje C por ser de los más simples. Como la información que ofrece esta librería será utilizada por terceros programas se ha hecho lo más sencilla posible, utilizando el terminal para mostrar datos.

La librería se ha implementado para la utilización del puerto serie del VBOX 3i, por lo que no funcionará con ninguna otra forma de conexión (ni USB ni Bluetooth) ni con otros aparatos diferentes al VBOX 3i.

Para mostrar el funcionamiento de la librería, se ha creado un programa de monitorización que utiliza las funciones básicas de ésta.

```
#include "gps.h"

int main(){
      int fd;
      datos d;
      fd = EncontrarAbrirPuerto(0x00000000 | VS | LONGACC | LATACC
| GLSAT | GPSSAT | SQ | ST | IKFS, 0x00000000 | YAW | XACEL | YACEL
| TEMP | PITCH | ROLL | ZACEL, 20);
      while (1)
      {
            d = CogerTraza(fd);
            system("clear");
            ImprimirDatos(d);
            EsperarSiguienteTraza(fd, 0.05);
      }
      close(fd);
      return 0;
}
```

Si se desea echar un vistazo al código de la librería, este se encuentra en el anexo F.

Este programa aquí escrito se conectará al VBOX 3i y nos mostrará por pantalla todos los datos que recoge cada 0.05 segundos.

Hay 4 funciones principales implementadas, las cuales se explican a continuación:

int EncontrarAbrirPuerto(int conf, int conf_imu, int hz):

Dicha función devuelve el descriptor de fichero perteneciente al VBOX 3i, en donde irá descargando toda la información que transmita al ordenador. Se le pasan tres

argumentos, conf, conf_imu y hz. Las dos primeras sirven para indicarle al programa que datos se encontrarán en la traza, determinado así el tamaño de ésta. Se sabe cuales hay porque previamente se habrá configurado el GPS con el programa VBOXTools, indicándole los datos a mandar. Por su parte, Hz indica la frecuencia en Hercios por la que se transmite por el puerto serie. Esto ayudará en los tiempos de espera entre traza y traza. Ver el apartado "3.2.1 – Configuración" para más información.

La función en si hace una comprobación de todos los puertos serie del ordenador. También comprueba los puertos USB por si se está utilizando un cable adaptador serie-USB. Solo hará una comprobación en los diez primeros puertos de cada tipo, antes de pasar al otro.

Si encuentra algún dispositivo conectado accederá a éste e intentará leer la traza que éste está mandando. Comprobando el principio de la traza se identificará o no si es la del VBOX 3i. En caso de no serlo, pasará al siguiente puerto.

Si resulta ser la traza del VBOX 3i, se realizarán diez lecturas adicionales para comprobar que se ha sincronizado bien con el dispositivo, ya que hay veces que se pierde el inicio de la traza y se empieza a capturar a mitad. En caso de fallar volveremos a empezar, mientras que si se realizan las diez comprobaciones satisfactoriamente se sale de la función devolviendo el descriptor de fichero correspondiente.

Conf: Si se envía el mínimo número de datos, conf deberá tener valor 0x00000000. Si hemos añadido alguno de los demás datos opcionales, bastara con añadirlo con un OR además de la nomenclatura del campo.

La nomenclatura de los distintos campos adicionales se puede ver en la figura 64.

VS	Velocidad vertical
LONGACC	Aceleración de longitud
LATACC	Aceleración de latitud
GLSAT	Número de satélites GLONASS
GPSSAT	Número de satélites GPS
SQ	Calidad de la velocidad
ST	Tipo de solución
IKFS	Estado del filtro Kalman

Figura 64: Nomenclatura de datos de conf.

Por ejemplo, si el VBOX está configurado para devolver la velocidad vertical y el numero de satélites GPS, conf deberá valer 0x00000000 | VS | GPSSAT.

Conf_imu: Si no tenemos la IMU conectada o no hemos configurado el VBOX para que devuelva ningún dato de ésta, el valor de conf_imu deberá ser 0x00000000.
En caso de que sí devuelva algún dato de la IMU, se añadirá también con un OR más la nomenclatura del campo. Estas nomenclaturas se pueden ver en la figura 65.

YAW	Dirección a la que mira
XACEL	Aceleración en X
YACEL	Aceleración en Y
TEMP	Temperatura
PITCH	Inclinación delantera y trasera
ROLL	Inclinación lateral
ZACEL	Aceleración en Z

Figura 65: Nomenclatura de datos de conf_imu

Un ejemplo de su uso sería que conf_imu tomase por valor 0x00000000 | TEMP | ZACEL, lo que significaría que en la traza está incluida la información de la temperatura y de la aceleración en Z.

En caso de no poner correctamente los argumentos conf y conf_imu, el resto de funciones podrían devolver información errónea o incluso podría no funcionar el programa.

Hz: Aquí se pondrá la frecuencia de muestreo del puerto serie. Cambiar su valor nos permitirá tomar valores en diversos intervalos de tiempo. Si Hz vale 20, le estamos diciendo que la frecuencia de muestreo es de 20Hz y que habrá una nueva traza cada 0.05 segundos. Los casos de Hz = 100 y Hz = 50 son especiales. Para un valor de 50, solo se mostraran los campos básicos, por lo que se ignorará lo que se haya puesto en conf y conf_imu. Si toma valor 100, solo “Número de satélites”, “Tiempo” y “Velocidad” serán mostrados.

Advertencia: Para valores de Hz = 100, se tardará en capturar la traza al conectarse utilizando la función si no se cuenta con un ordenados con un buen procesador.

datos CogerTraza(int fd);

La función coge como argumento el descriptor de fichero devuelto con la anterior función y captura la siguiente traza que se mande, desechando todas las anteriores. Devuelve la estructura datos, rellenando cada campo con la información de la traza.

De esta estructura se podrá sacar cualquier dato en cualquier momento, siempre y cuando se haya especificado en el argumento conf o conf_imu de la primera función que la traza contiene dicho dato. En caso contrario, ese campo contendrá datos no inicializados.

Para el caso de utilizar Hz = 100 en la primera función, si no se cuenta con un buen procesador, esta función puede tardar demasiado en ejecutarse y saltarse una traza.

Cuando se inicia la función, esta realiza varios read() del descriptor de fichero hasta que no encuentra más datos. De esta forma, el próximo dato que leamos será el siguiente que recibamos.

Se comprobara que la nueva traza leída sea correcta aun sin haber llegado toda. En caso de ser negativa la comprobación se desechará lo leído hasta entonces y se esperaran nuevos datos. En caso afirmativo se procederá a rellenar la estructura datos, devolviéndola al finalizar.

Mientras que los datos recogidos por el VBOX 3i se transmiten en formato complemento a 2, los recogidos por la IMU se encuentran en IEEE coma flotante, habiendo de tratar ambos tipos de datos de forma distinta.

Para los primeros de complemento a 2, si estos representan un valor positivo no hay problema, pero si representan un valor negativo, hay que hacer la conversión a formato decimal para poder introducir en la estructura unos datos legibles.

En el caso de IEEE coma flotante, habrá que realizar la conversión de todos ellos a formato decimal.

En los campos de la estructura datos, estos estarán en las unidades de medida que indica la tabla de la figura 66.

Dato	Unidad
Número de satélites	Satélites
Tiempo	Centésimas de segundo
Latitud	Minutos * 100000
Longitud	Minutos * 100000
Velocidad	Km/h * 100
Dirección	Grados * 100
Altura	Metros * 100
Velocidad vertical	Km/h * 100
Aceleración en la longitud	Fuerza G * 100
Aceleración en la latitud	Fuerza G * 100
Número de satélites GLONASS	Satélites
Número de satélites GPS	Satélites
Estado del filtro Kalman	
Tipo de solución	
Calidad de la velocidad	Km/h *100
Giro en X	Grados / s
Giro en Y	Grados / s
Giro en Z	Grados / s
Aceleración en X	Fuerza G

Aceleración en Y	Fuerza G
Aceleración en Z	Fuerza G
Temperatura	Grados Celsius

Figura 66: Tabla de unidades de medida

Las unidades del estilo de la latitud o la longitud, están en minutos pero multiplicado por 100000. Para obtener el valor real de latitud en minutos habría que dividir el valor del campo por 100000.

void ImprimirDatos(datos d);

Imprime por pantalla los datos contenidos en la estructura datos, pasada como argumento. Muy útil para monitorizar el VBOX 3i. Solo imprimirá aquellos datos que se hayan especificado que contiene la traza en la primera función.

Todas las conversiones de unidades tales como pasar la latitud a grados y minutos se hace aquí.

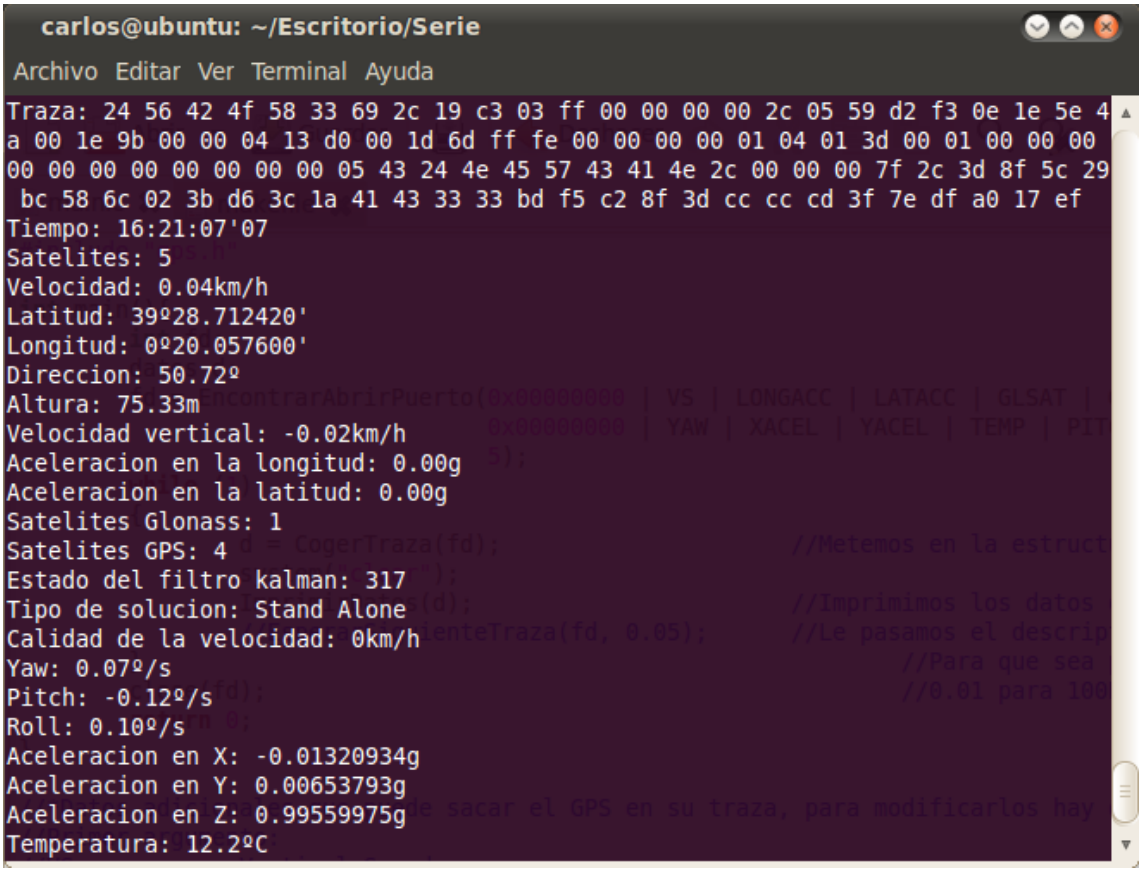

Figura 67: Salida de la función ImprimirDatos(d);

La imagen mostrada muestra el terminal ejecutando el programa. En este caso se están recogiendo todos los datos, incluidos los de la IMU.

void EsperarSiguienteTraza(int fd, double MUESTREO);

Se queda esperando hasta que pasen tantas trazas como quepan en el tiempo definido por MUESTREO. Esto sirve para elegir qué frecuencia de muestreo queremos, por si no nos hace falta obtener la información cada pocas centésimas de segundo y queremos tomar una cada x tiempo.

Debido a que se puede configurar el VBOX para que transmita por el cable serie a una frecuencia determinada, se deberá poner un valor de MUESTREO múltiplo de esta frecuencia. Si se utiliza una frecuencia de 20HZ, deberán usarse valores múltiplos de 0.05, ya que el dispositivo transmitirá cada nuevo dato cada 0.05 segundos. La figura 68 muestra que múltiplos hay que usar para cada frecuencia.

Frecuencia del puerto serie	MUESTREO debe ser múltiplo de:
100 Hz	0.01
50 Hz	0.02
20 Hz	0.05
5 Hz	0.2

Figura 68: Tabla de frecuencias y muestreo.

En caso de querer usarlo en un programa más grande, será necesario el uso de hilos o de lo contrario el programa no seguirá hasta que esta función termine.

Para el caso de utilizar Hz = 100 en la primera función, si no se cuenta con un buen procesador, esta función puede tardar demasiado en ejecutarse y saltarse una traza.

Con todo esto, estas funciones serán de utilidad para futuros usos del VBOX 3i, con visión a que sean útiles para programas que controlen vehículos móviles y necesiten de las coordenadas GPS para orientarse o manejarse.

La librería es de código abierto así que se puede realizar cualquier modificación que se desee o se necesite.

4. Pruebas y comparativas

Una vez tenemos elegido y configurado el sistema GPS que vamos a utilizar, debemos realizar pruebas para comprobar que efectivamente nos será útil para posteriormente construir un vehículo móvil autónomo.

Las pruebas intentarán analizar y comparar, sobretodo, la precisión de la latitud y de la longitud, cruciales para nuestro objetivo.

4.1 VBOX 3i en reposo

Esta prueba se ha realizado para ver cómo se comporta el VBOX 3i en un entorno controlado y estático. Se quiere estudiar la precisión de los datos por lo que en esta prueba se dejara el GPS completamente quiero. Así, la diferencia de las diferentes latitudes y longitudes nos indicarán el margen de error que podemos esperar del aparato. También se podrá ver que datos que en teoría deberían dar 0, como la velocidad, obtienen ciertos valores distintos a 0. La opción de optimización en el menú Setup del VBOXTools se ha seleccionado “Normal”, considerando que el vehículo podrá estar parándose y moviéndose continuamente. La optimización ideal para este tipo de prueba sería “Low Dynamics”, pero no es la que se utilizará en caso de que estas pruebas sean un éxito y se use este GPS para construir el vehículo móvil autónomo.

Se estudiarán 3 casos: utilizar el VBOX 3i solo, con IMU y con DGPS.

Para realizar esta prueba se ha situado el equipo en el tercer piso de la Ciudad Politécnica de la Innovación, entre los edificios M y L. Es una posición con mucho cielo abierto, pero también con bloques de edificios alrededor que pueden bloquear la señal de algún satélite, lo que podría simular una ciudad.

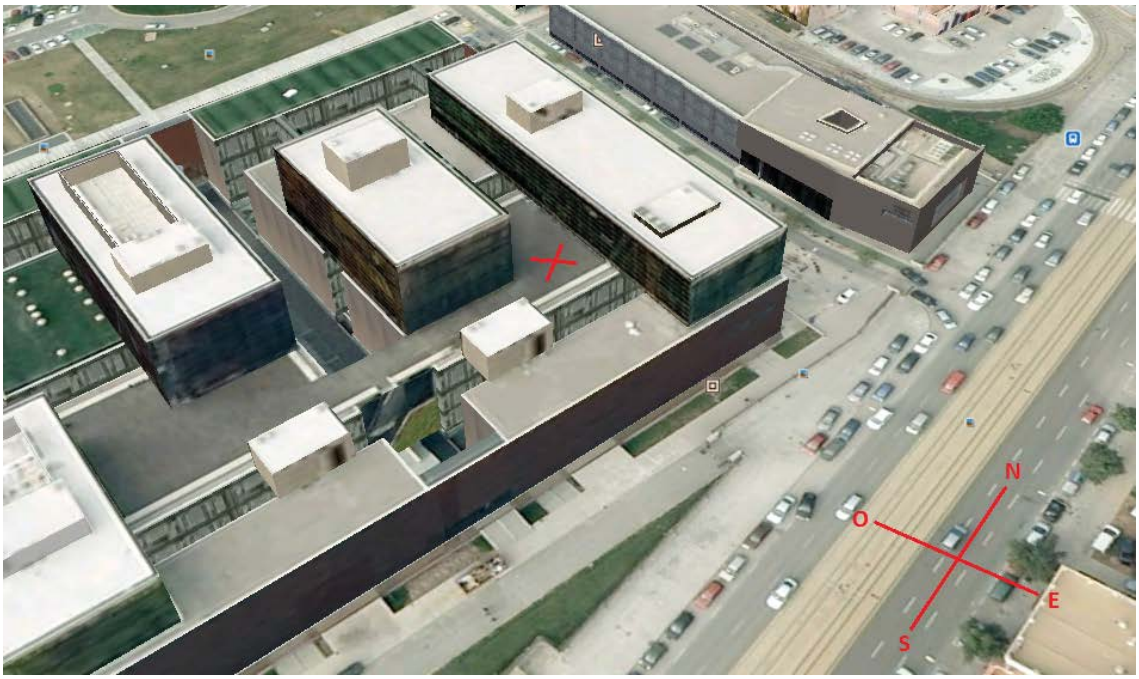

Figura 68: Modelo 3D de la Ciudad Politécnica de la Innovación

El modelo 3D tenía el fallo de que la parte donde se realizó la prueba no estaba hecha. Al ser simétrico el edificio, he colocado la marca en el lado opuesto de éste, para mostrar la disposición de los edificios.

La realización de estas fue durante 3 días distintos, manteniendo la antena en el mismo punto durante 6 horas y 35 minutos cada uno de los días.

Figura 69: Área de pruebas con el VBOX 3i + DGPS montados

4.1.1 Caso del VBOX 3i solo:

Este caso debería ser el más impreciso de todos, ya que no contamos ni con la IMU ni con el DGPS para corregir el error.

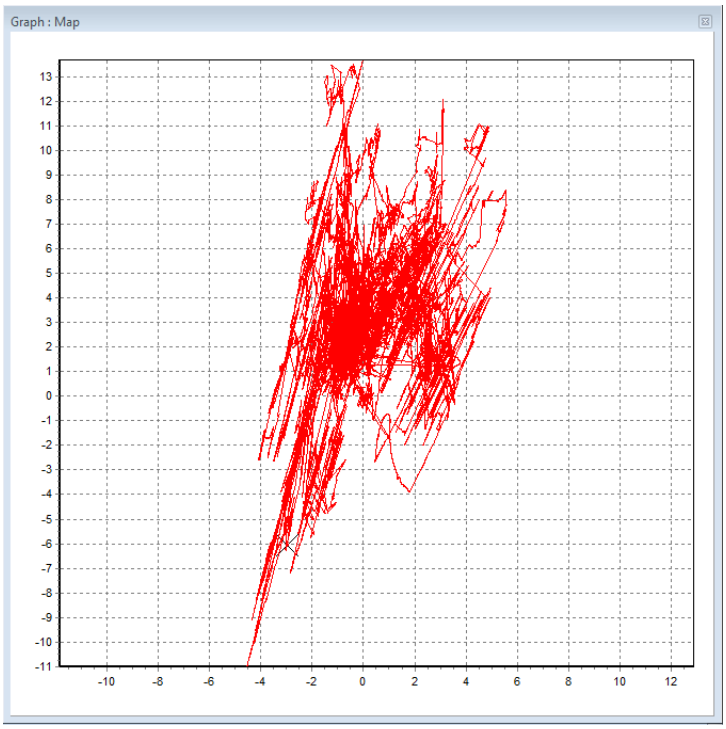

Figura 70: Recorrido VBOX 3i parado

Lo visto en la figura 70 es el recorrido que, en teoría, ha realizado el GPS basándonos en la latitud y la longitud.

Éste ha estado quieto en todo momento y durante 6 horas ha realizado todo este recorrido. La escala es en metros. Las gráficas de la latitud y la longitud nos aclaran bastante este asunto:

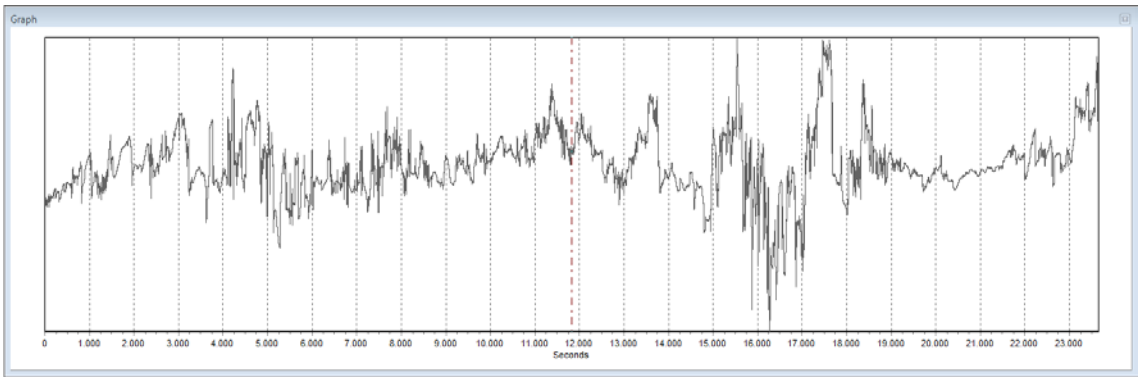

Figura 71: Latitud VBOX 3i parado

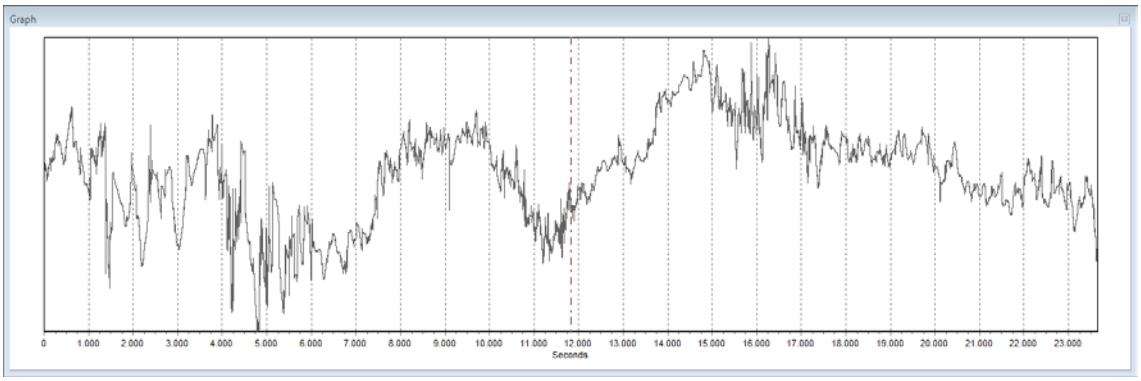

Figura 72: Longitud VBOX 3i parado

Se ve claramente como ambos valores fluctúan a lo largo de las gráficas, la longitud algo más que la latitud. En ambas se podría trazar fácilmente una línea horizontal que cruzase la gráfica, encontrando la media.

Contamos con una herramienta explicada en el apartado “3.2.1 VBOXTools – Post-procesamiento” que nos permite conocer la media de estos dos valores. Está expresada en minutos, pero dividiendo este valor por 60 obtenemos los grados y, posteriormente multiplicando por 60 la diferencia de la división anterior obtenemos los minutos.

Latitud media: 2368.638’ -> 39□ 28.638’

Longitud media: 20.050’ -> 0□ 20.050’

Compraremos estas coordenadas con las siguientes pruebas que hagamos para ver si esta media coincide con las de las demás.

Sabiendo el punto donde coinciden ambos datos y teniendo en cuenta que debería tener un error menor de 3 metros CEP el 95% del tiempo, contaríamos con una gráfica del recorrido como esta, con la media en el centro de la cruz verde y con un radio de error de 3 metros:

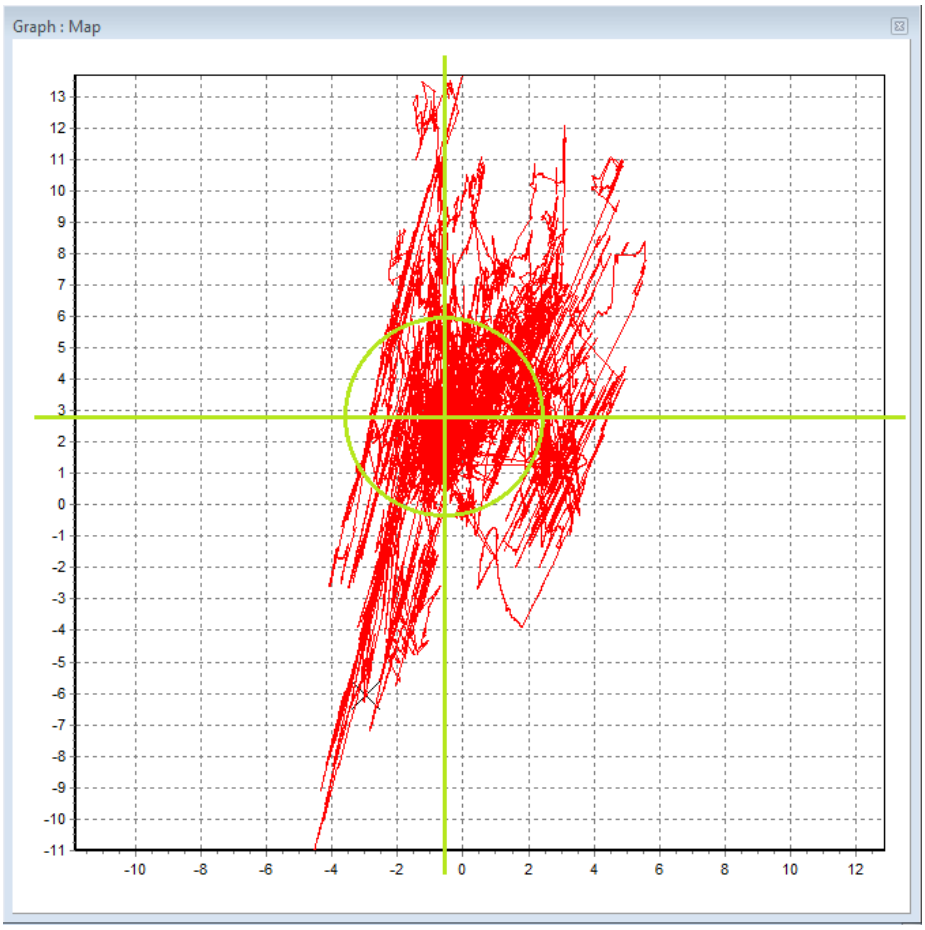

Figura 73: Punto medio del recorrido VBOX 3i parado

El 95% de los datos deberían encontrarse dentro del círculo verde.

Ciertamente sí que se ve que esa zona es más densa que la de fuera, pero aun así no parece que abarque el 95% de los datos, ya que fuera del círculo se ven bastantes. Seguramente si lo hubiésemos dejado más tiempo, se habría acabado por completar la parte de la izquierda del círculo.

Para comprobarlo, usando el Excel se ordenan de mayor a menor las columnas de latitud y longitud y buscamos que coordenada está en el puesto 2.5% de la columna y cual en la 97.5%. Esto nos dará un valor aproximado de donde están contenidos el 95% de los datos.

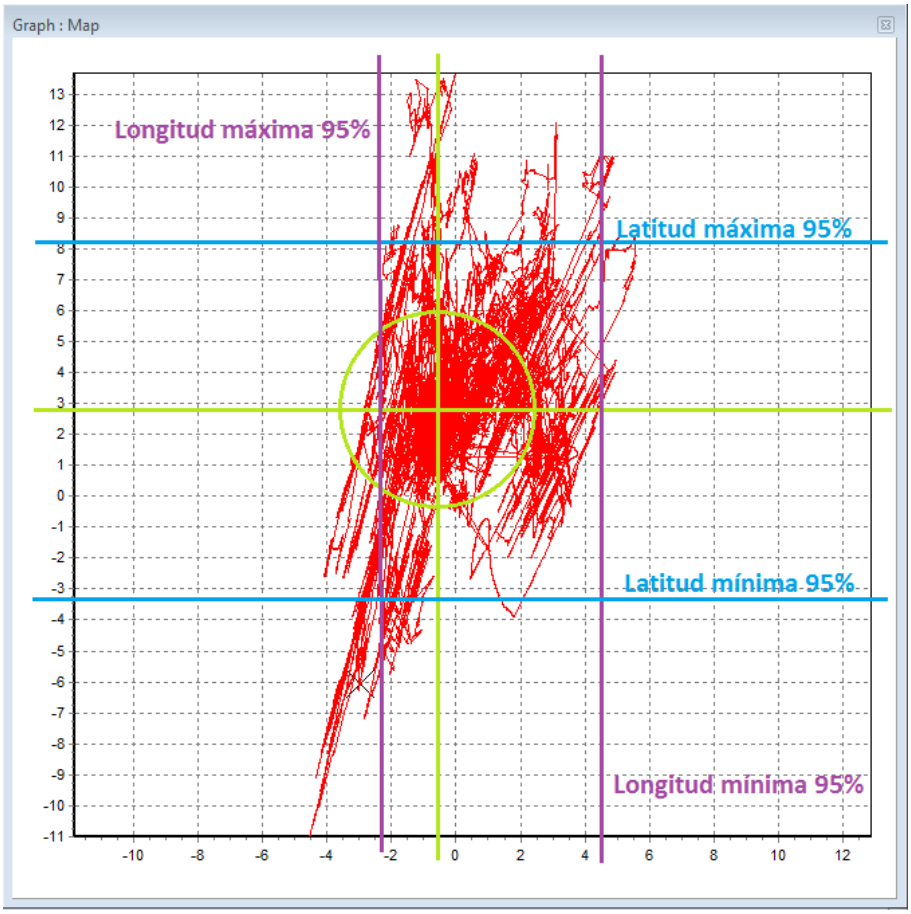

Figura 74: 95% de los datos VBOX 3i parado

El 95% de los datos de longitud y latitud se encuentran en el recuadro formado por las cuatro líneas dibujadas. Si moviésemos las líneas moradas verticales que representan la longitud un poco hacia la izquierda, quedarían del mismo tamaño más o menos que el círculo verde. Podemos mover las líneas debido a que al realizar la mediana no contamos con los valores más extremos que si se toman en cuenta al calcular la media, pudiendo cambiar la posición de las líneas si así se cree oportuno, siempre y cuando el 95% de los datos se encuentren entre éstas. Con la latitud sin embargo, nos encontramos con que hay muchos datos delimitados por las líneas azules horizontales que se encuentran fuera del círculo. Esto indica que el error es mayor de 3 metros.

Se ha obtenido un error de unos 3 metros 95% CEP en la longitud y 6.5 metros 95% CEP en la latitud, un resultado que no se adecua bastante a lo prometido y que no nos sirve para nuestro objetivo final, ya que contiene demasiado error.

4.1.2 Caso del VBOX 3i con la IMU:

En esta ocasión el error obtenido debería ser menor, ya que la IMU ayuda a corregirlo y suavizarlo. Seguiremos contando con los 3 metros 95% CEP ya que el fabricante no ha dejado constancia de que el error sea menor con la IMU.

Volveremos a analizar los mismos datos y los compararemos con los anteriores.

Las gráficas de la latitud y la longitud se aprecian muchísimo más suaves, con menos subidas y bajadas repentinas, aunque siguen fluctuado.

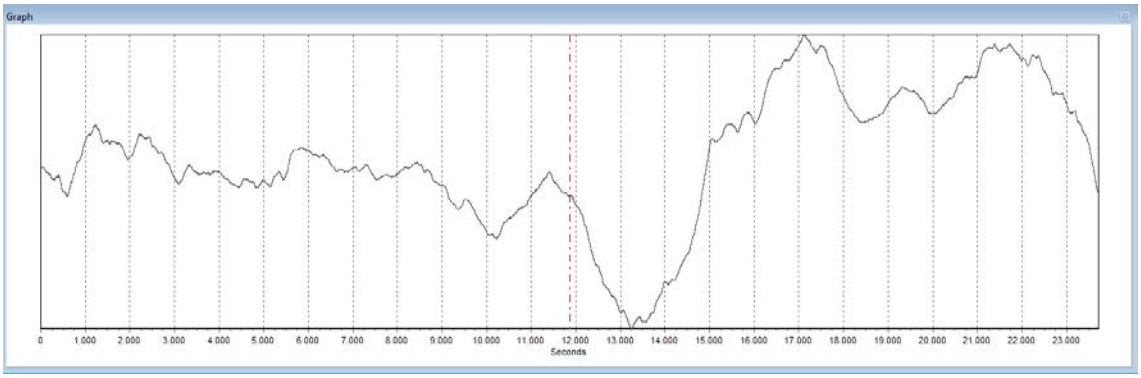

Figura 75: Latitud VBOX 3i + IMU parado

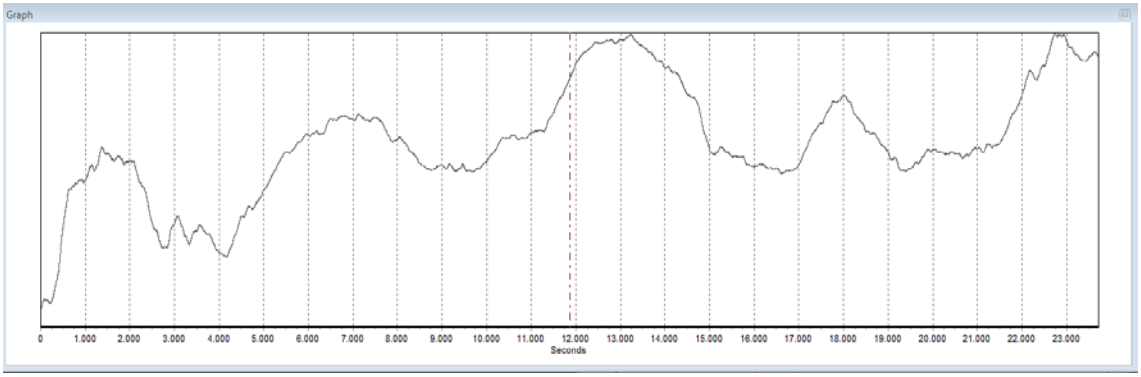

Figura 76: Longitud VBOX 3i + IMU parado

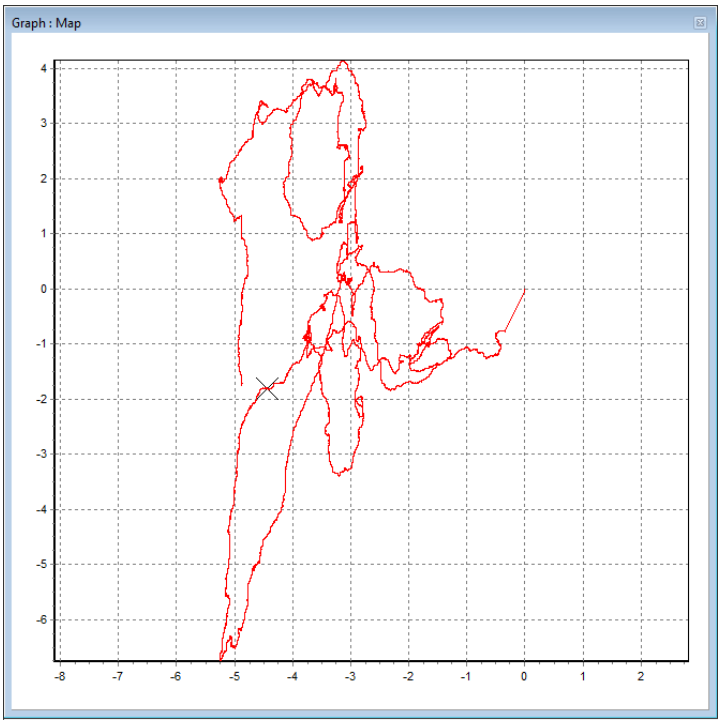

Figura 77: Recorrido VBOX 3i + IMU parado

En esta ocasión la media de la longitud no se ve muy clara, así que calcularemos la media como antes y la situaremos en la gráfica del recorrido, la cual es mucho más clara y limpia esta vez. Esto y la suavidad de las gráficas es debido a que el IMU estaba indicando en todo momento que el GPS se encontraba parado, lo cual ha hecho que hubiese cambios mucho más lentos en la latitud y la longitud.

Latitud media: 2368.639’ -> 39□ 28.639’

Longitud media: 20.051’ -> 0□ 20.051 ’

Comparado con la media de antes la diferencia es de tan solo una milésima, por lo que ambos resultados se pueden dar por buenos.

Es probable que, de haber dejado el VBOX 3i con la IMU mas tiempo obteniendo datos, la grafica del recorrido hubiera acabado siendo más parecida a la anterior en cuanto a densidad y podría haber rellenado más zonas.

En esta ocasión también nos salimos un poco de los 3 metros 95% CEP de error prometidos en la latitud. Calculamos el centro y el radio en el que se tendría que tener el 95% de los datos:

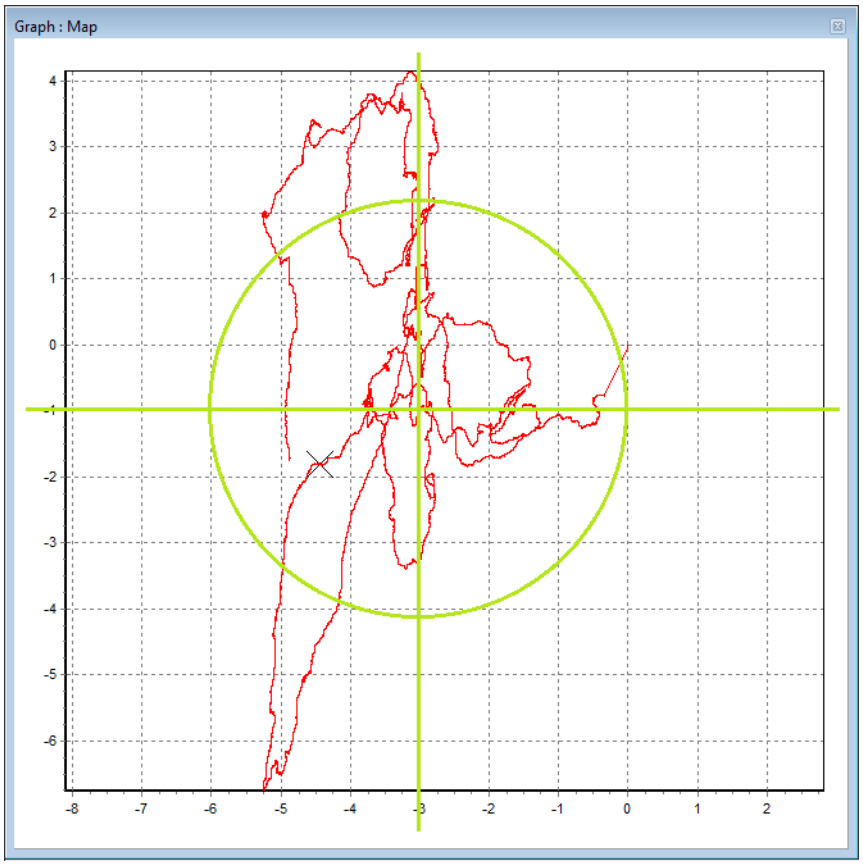

Figura 78: Punto medio del recorrido VBOX 3i + IMU parado

Hacemos como antes y quitamos el 2.5% de coordenadas más pequeñas y más grandes del conjunto de coordenadas y dibujamos las líneas que delimitaran la zona en que se contiene el 95% de los datos.

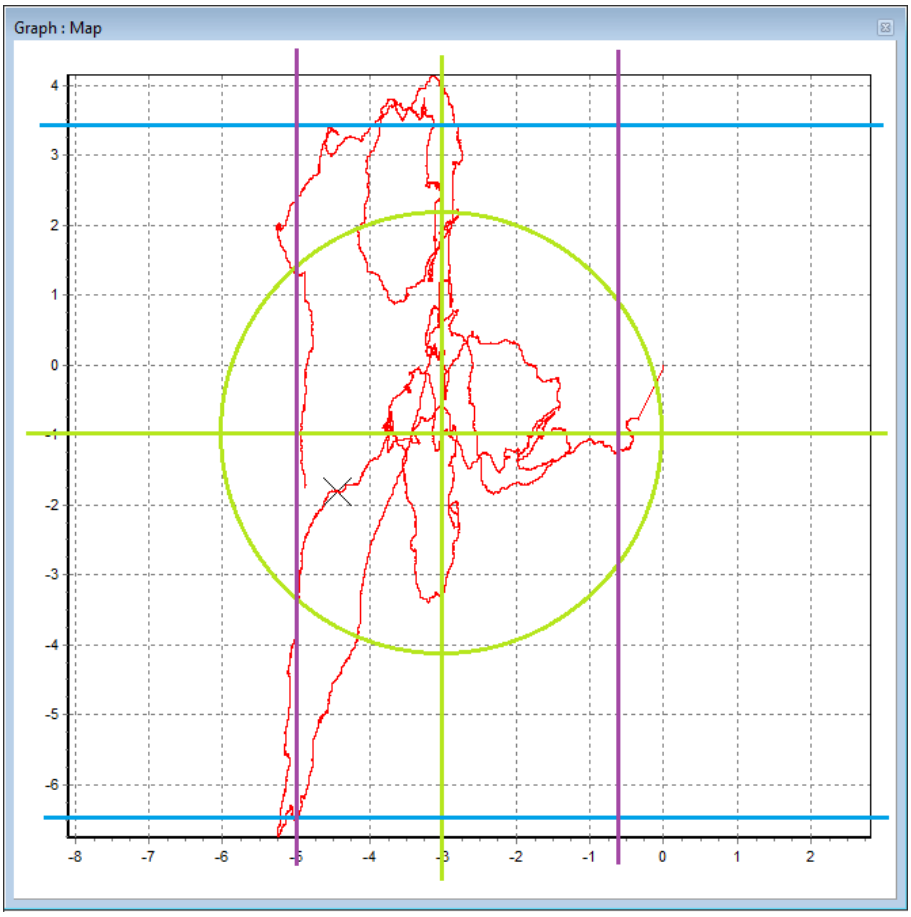

Figura 79: 95% de los datos VBOX 3i + IMU parado

Al final ha acabado ocurriendo lo mismo que antes, una parte del 95% de los datos de la latitud que debería encontrarse dentro del circulo esta fuera de este, mientras que los datos de la longitud si se encuentran dentro.

La latitud ha obtenido un error de casi 5 metros 95% CEP mientras que en la longitud se ha obtenido 2.25 metros 95% CEP de error.

Estos datos son mucho mejores que los recogidos en la primera prueba y ya se acercan más al umbral de error que nos había dicho el fabricante.

Sin embargo, ¿Por qué falla más la latitud que la longitud? Ambas gráficas de recorrido se parecen bastante. ¿Podría esto deberse a la disposición de los edificios?

Durante las pruebas se veía más trozo de cielo de este a oeste que de norte a sur. Esto indica que el GPS podía conectarse con más satélites cercanos al horizonte en el este y el oeste y menos al norte y al sur.

La latitud es un valor que nos indica cuanto al norte o al sur estamos con respecto al ecuador, mientras que la longitud nos informa de cuanto al este o al oeste estamos del meridiano de Greenwich.

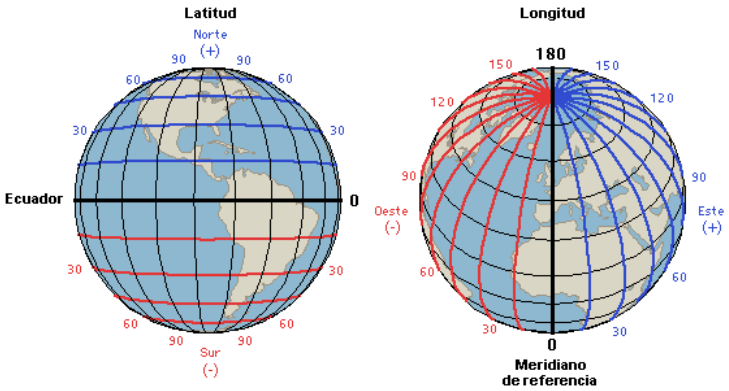

Figura 80: Latitud y longitud

Si tuviésemos dos satélites colocados en latitudes similares pero distintas longitudes, y nosotros nos encontrásemos en un punto intermedio entre ambos, sería más fácil calcular nuestra posición en la coordenada de la longitud que de la latitud.

Si los satélites colocados al este y al oeste de nuestra posición calculan mejor nuestra longitud, mientras que los situados de norte a sur calculan mejor nuestra latitud, es fácil llegar a la conclusión de que si hubiésemos tenido el mismo campo de visión en todas las direcciones, el error en la latitud y la longitud sería el mismo, por lo que en un entorno sin edificios ni obstáculos, el VBOX 3i alcanzaría sin problemas un error de 3 metros 95% CEP.

4.1.3 Caso del VBOX 3i con el DGPS:

Esta debería ser la prueba más precisa de todas, aunque también la más exigente, ya que se va a intentar alcanzar los 2 cm de error.

No obstante viendo las gráficas de la latitud, longitud y recorrido se puede apreciar que algo no marcha bien:

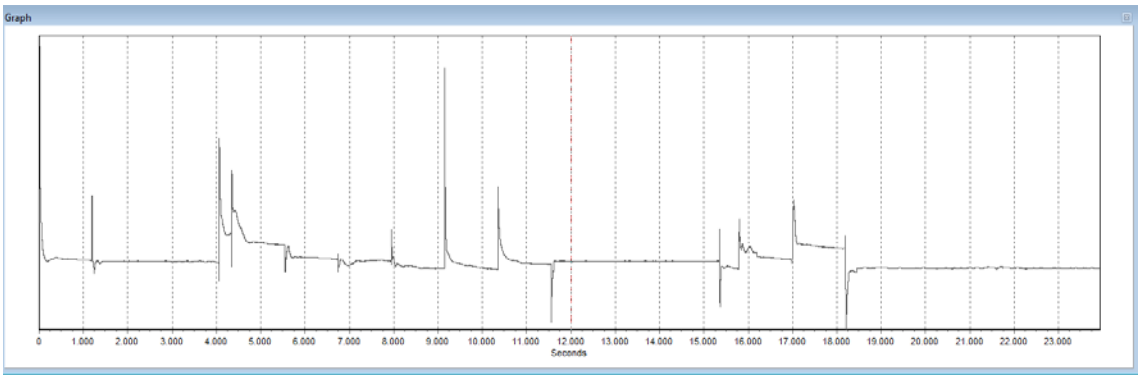

Figura 81: Latitud VBOX 3i + DGPS parado

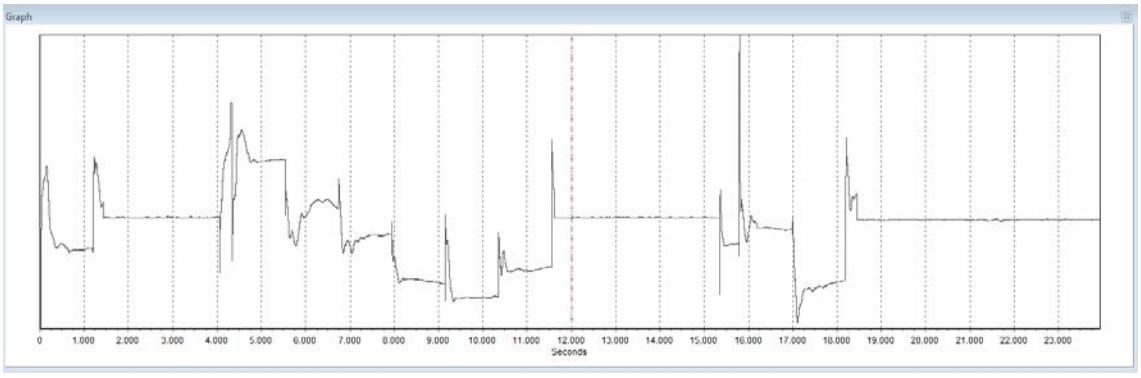

Figura 82: Longitud VBOX 3i + DGPS parado

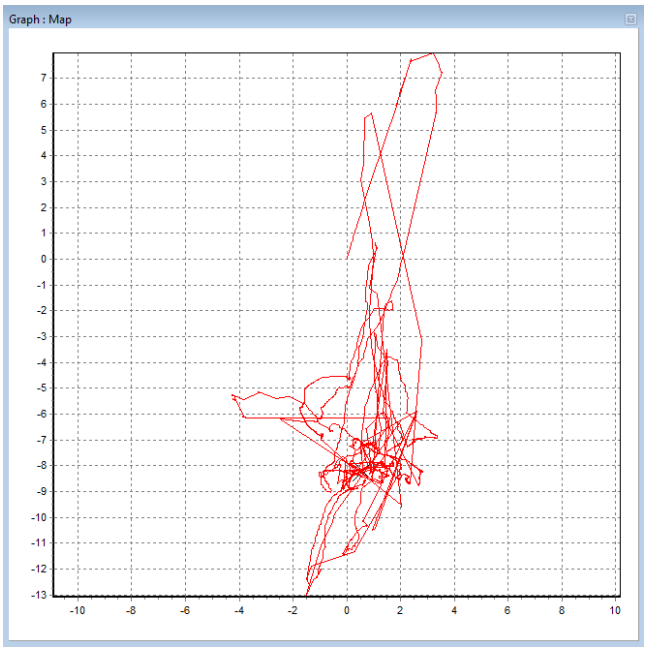

Figura 83: Recorrido VBOX 3i + DGPS parado

En principio, la gráfica del recorrido debería haber sido un pequeño punto pero en cambio vemos una figura que llega a alcanzar una diferencia de 20 metros en la latitud.

Otro aspecto raro son los extraños saltos que se aprecian en las dos gráficas de latitud y longitud. Podríamos decir que los valores de latitud y longitud mantienen las coordenadas constantes, pudiendo llegar de esta forma a un error real de tan solo 2 cm, hasta que sucede el salto.

Comprobando el resto de datos obtenidos del GPS, notamos que la señal de que el DGPS está activado desaparece varias veces durante la prueba, para volver a aparecer un segundo después:

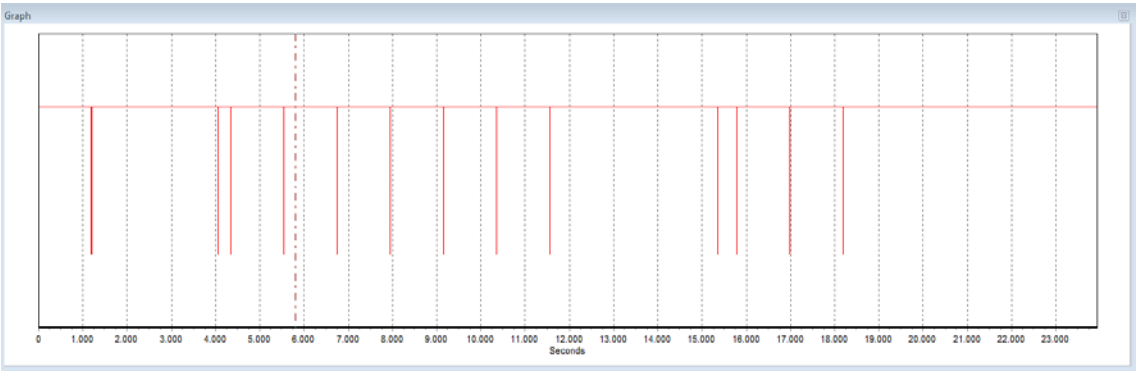

Figura 84: DGPS

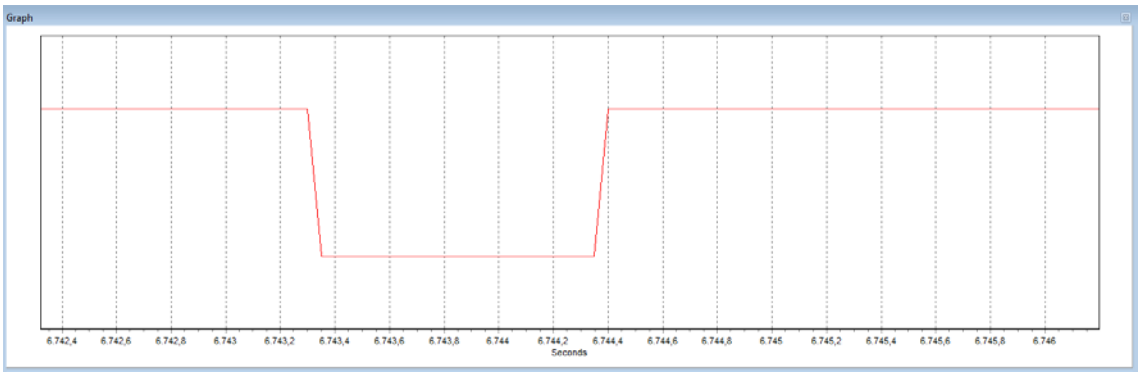

Figura 85: Un segundo de caída de la señal DGPS

Esto causa que cada vez que hay una caída, la latitud y la longitud pierden su posición inmediatamente, necesitando de varios minutos para volver a estabilizarse, pudiendo no volver a la misma posición algunas veces.

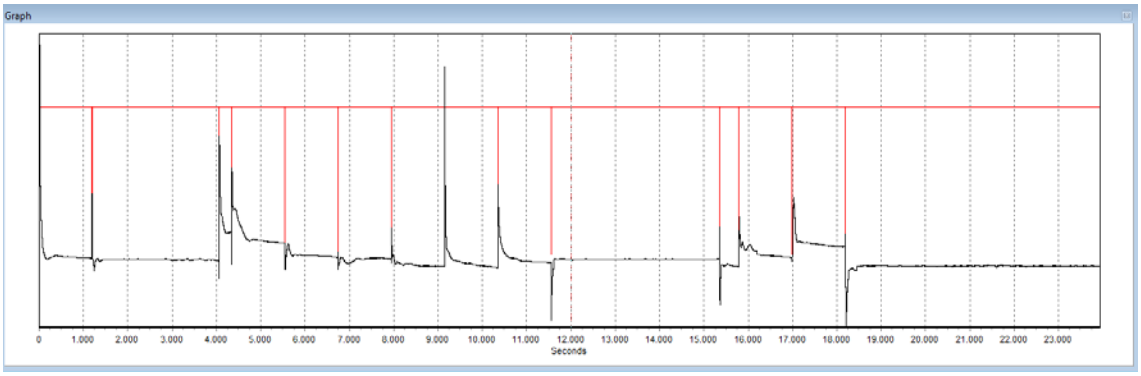

Figura 86: DGPS + Latitud

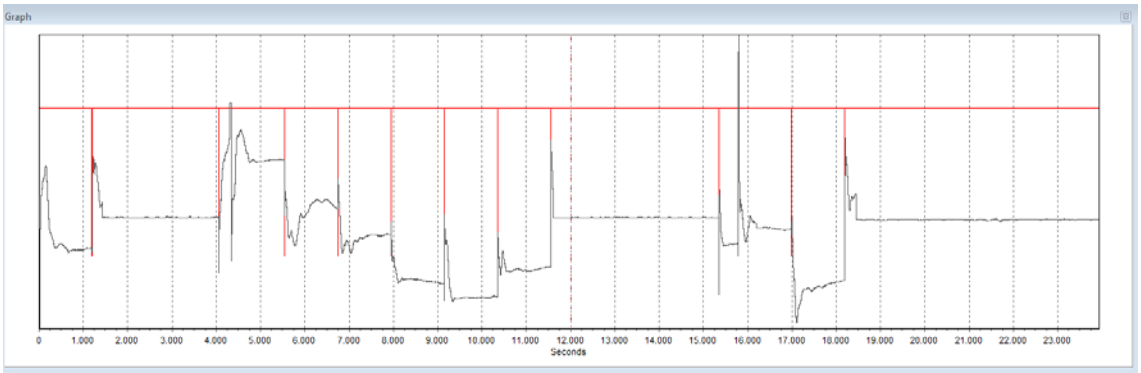

Figura 87: DGPS + Longitud

Además se aprecia un cambio en las coordenadas a las que se vuelve después de cada salto, más pronunciado en la longitud. Posiblemente sea debido a la posición de la antena del DGPS, la cual tiene más obstáculos que la antena del VBOX 3i.

Esta vez no hace falta calcular la media pues en los momentos en que las coordenadas son estables se puede apreciar sin problemas el punto en el que las coordenadas son realmente las correctas.

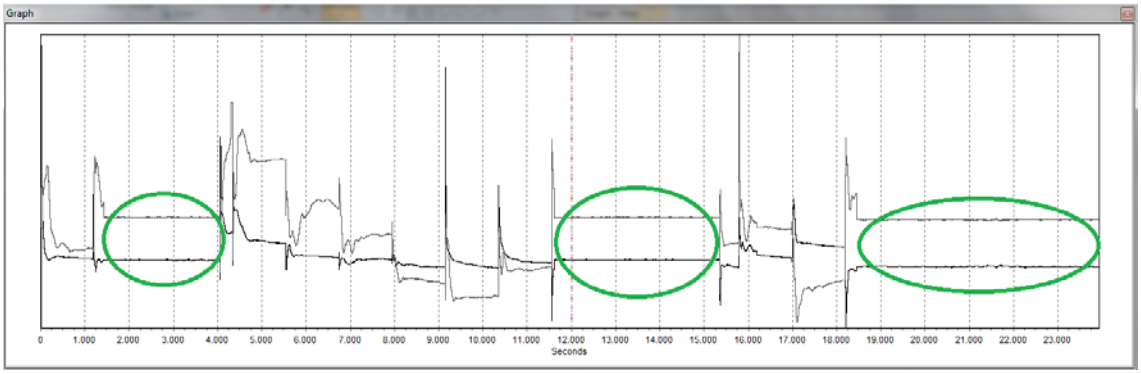

Figura 88: Comparación de la latitud y la longitud

Latitud: 2368.637′ -> 39 28.637′

Longitud: 20.051′ -> 0 20.051′

La latitud es menor a las anteriormente calculadas en una y dos decimas respectivamente, mientras que la longitud es la misma que la calculada para la IMU.

También se vuelve a apreciar la forma que han tomado el resto de gráficas.

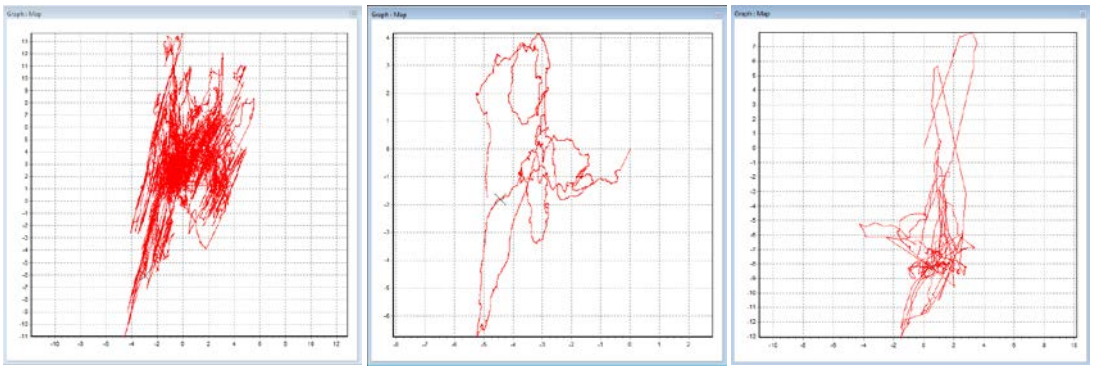

Figura 89: Comparativa de recorridos

La última adquiere esa forma cada vez que ocurre un salto, ya que al perder el DGPS parece que actúa como si no lo hubiese estado usando, volviendo al estado de precisión de 2cm poco a poco.

Esto se podría solucionar un poco aplicando a la vez que el DGPS un filtro Kalman (Explicado en el apartado “3.2.1 VBOXTools – Configuración”) que suavizaría los saltos y haciendo que se vuelva antes a la normalidad.

Sin embargo, ese fallo en el envío de la señal del DGPS no es normal por lo que habría que pedir ayuda técnica para solucionar el problema, ya que es posible que haya que cambiar el VBOX 3i o el DGPS.

En caso de que no ocurriese la perdida de señal, sería cuestión de ver si cumple la precisión de 2cm 95% CEP de error para poder utilizarlo para crear un vehículo móvil autónomo, aunque las gráficas de latitud y longitud parecen indicar que sí lo lograría.

Se puede eliminar de las gráficas los datos que no nos gustan debido a los saltos, dejando solo los rodeados de la figura 88. De esta forma podemos estudiar si alcanza los 2 cm 95% CEP de error en esos casos.

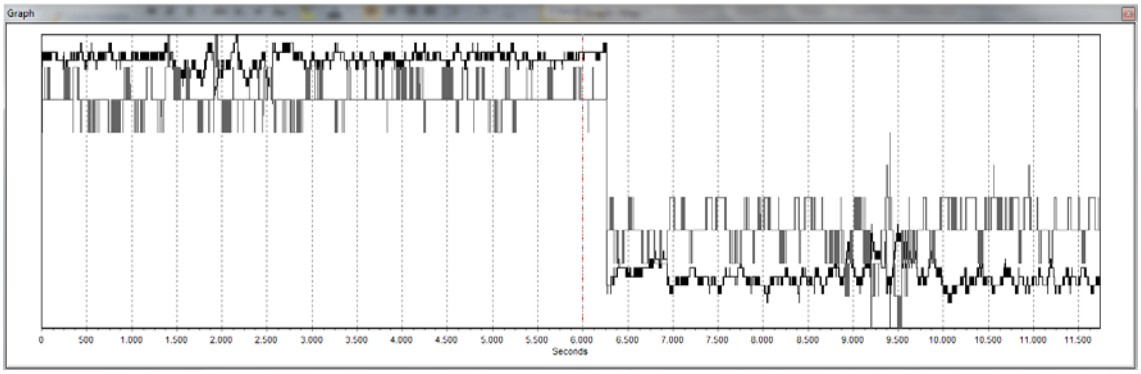

Figura 90: Latitud y longitud sin los datos anómalos

Se aprecian dos grupos de coordenadas distintas. Esto es causado por un fallo en la recepción del DGPS que al volver a corregirse ha vuelto a unas coordenadas diferentes. La gráfica del recorrido quedaría así:

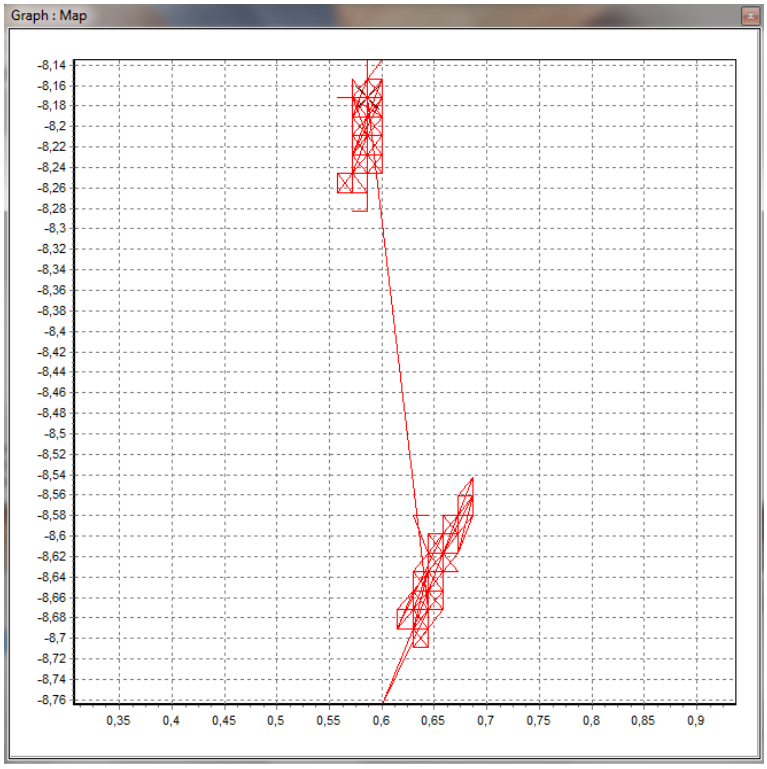

Figura 91: Recorrido recortado

Ambos grupos de coordenadas vuelven a tener una forma alargada en la latitud debido a la posición de los edificios.

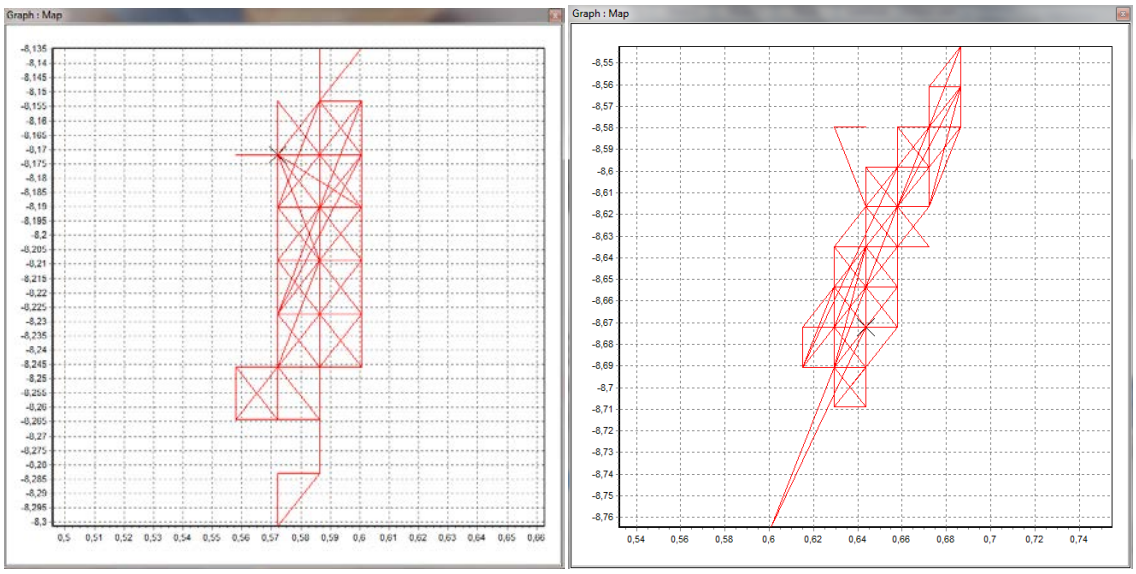

Figura 92: Grupos de coordenadas

En los dos se aprecia que tienen un error superior a 2 cm en la latitud pero bastante aproximado en la longitud.

El error en la latitud es superior a los 5 cm mientras que en la longitud alcanza los 2 cm en el primer grupo y 2.5 en el segundo.

No es la precisión de 2 cm que esperábamos pero casi, siendo aceptable si obtuviésemos siempre esta calidad en las coordenadas. Parte del error en la latitud proviene del edificio que no deja ver los satélites al norte y al sur de la posición de la antena, por lo que probablemente habríamos alcanzado los 2 cm 95% CEP de error con ambas coordenadas. No obstante, debido a las pérdidas de la señal DGPS que se producen y de que no siempre vuelve al mismo sitio, no se recomendaría usar este sistema para instalarlo en un vehículo móvil autónomo a menos que se solventen estos problemas.

4.2 VBOX 3i sobre un coche

En esta ocasión situaremos el GPS en un vehículo y se realizarán algunas vueltas a un circuito cerrado. Será lo que más se asemeje al uso real que se le quiere dar al VBOX 3i.

En este caso no podemos saber la precisión con la que contaremos exactamente ya que el vehículo estará en constante movimiento y no conocemos las coordenadas de los puntos por los que éste pasará. Lo que si se verá será la diferencia entre el punto inicial y el punto final, los cuales deberían ser prácticamente los mismos, debido a que acabamos donde empezamos. Se estudiará que recorra bien el circuito y el posible ruido que veamos, aparte de estudiar los datos anómalos que vayamos encontrando.

El circuito en que realizaremos la prueba será en el parking cercano a la Ciudad Politécnica de la Innovación. Se podrá ver el recorrido trazado en Google Earth.

Figura 93: Circuito de pruebas

Se estudiaran los tres mismos casos que antes, utilizando el VBOX 3i solo, con la IMU y con el DGPS.

4.2.1 Caso del VBOX 3i solo:

El montaje es sencillo: se coloca la antena en el techo del coche y el VBOX 3i en el interior. El coche se encuentra en una plaza de aparcamiento en la penúltima hilera de la derecha del parking. Saldremos de allí y volveremos al mismo punto tras dar una vuelta completa.

En la gráfica del recorrido obtenemos esto:

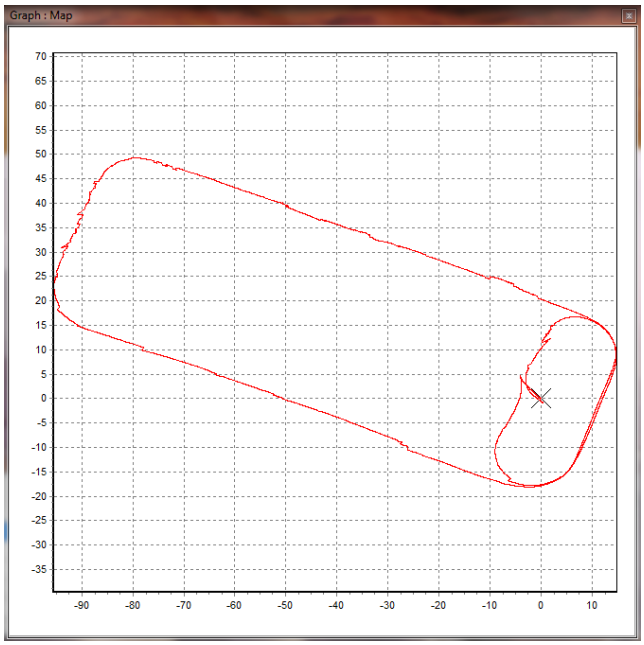

Figura 94: Recorrido en coche del VBOX 3i

Se puede ver a simple vista que el recorrido no es todo lo recto que debiera. Se nota más en las rectas, en las que se aprecian pequeños tambaleos.

Sin embargo, acercándonos al punto de inicio y fin, la diferencia en las coordenadas de ambos puntos es de poco más de un metro.

Aquí vemos el recorrido exportado a Google Earth:

Figura 95: Recorrido del VBOX 3i sobre Google Earth

Este resultado no está mal del todo, aunque siempre cabe la posibilidad de que se presente un error de tres metros.

Si construyésemos un vehículo autónomo con este grado de error, solo nos serviría para espacios en los que hubiese pocos obstáculos y con buen margen de maniobra para sortearlos, a menos que cuente con otro tipo de sensores que le proporcionen más precisión al vehículo. No nos serviría para que circulara él solo.

Como dato trivial, aquí están las gráficas de la latitud, longitud y velocidad, para ver como son los datos que se toman en movimiento.

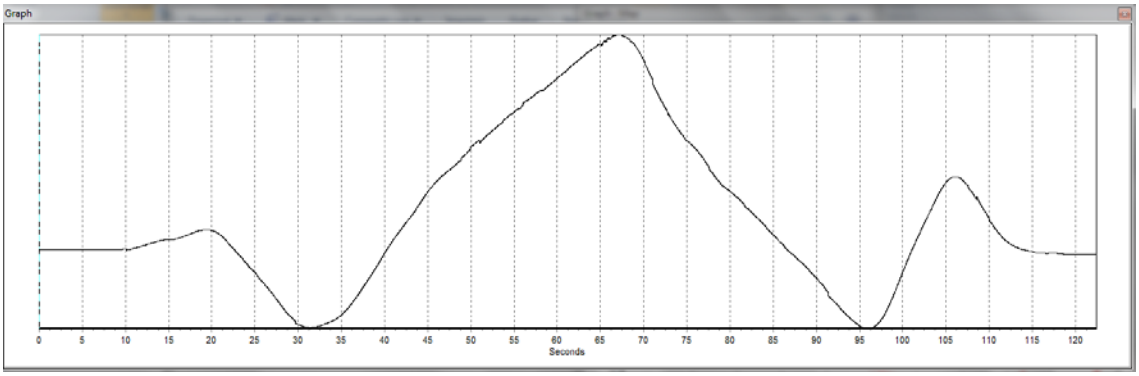

Figura 96: Latitud en coche del VBOX 3i

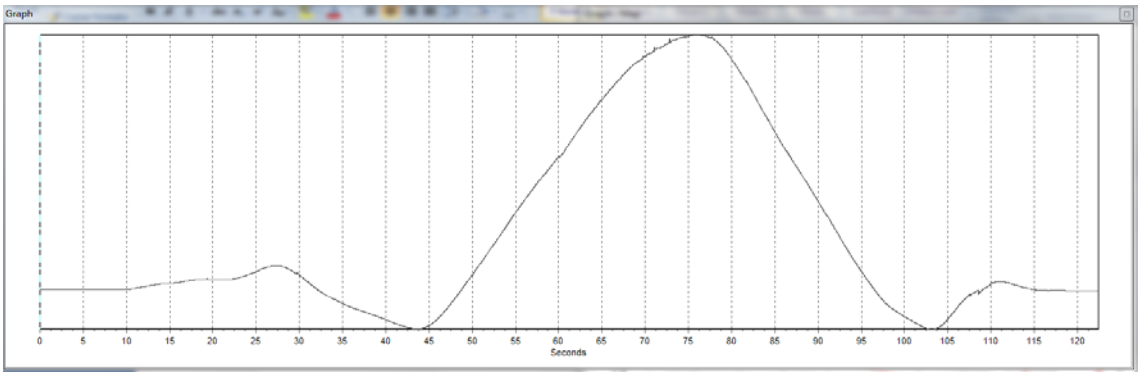

Figura 97: Longitud en coche del VBOX 3i

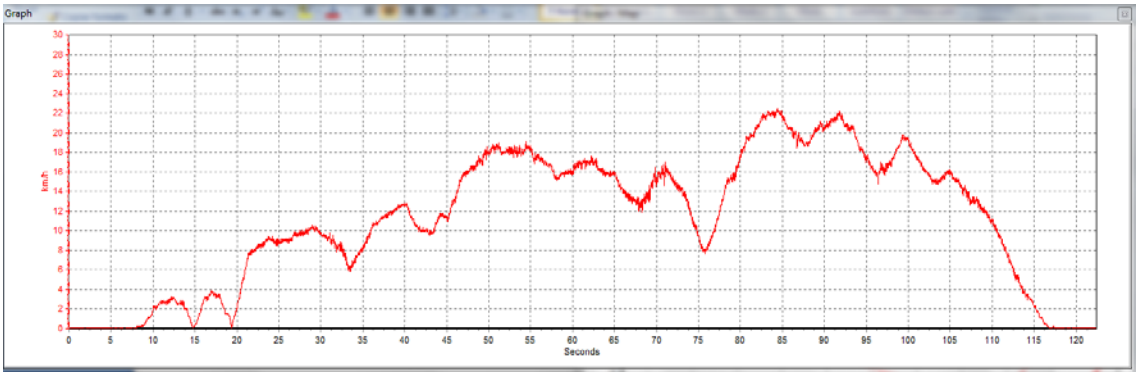

Figura 98: Velocidad en coche del VBOX 3i

Latitud y longitud parecen mucho menos ruidosas y más suavizadas. Esto se debe al movimiento y a que la escala en la que se mueven es mucho mayor.

4.2.2 Caso del VBOX 3i con la IMU:

Con la IMU el resultado debería ser similar pero mucho más suave y con menos error.

Se coloca la IMU en la posición más plana y céntrica posible y enseguida comprobamos que la gráfica del recorrido obtenida difiere bastante de la anterior:

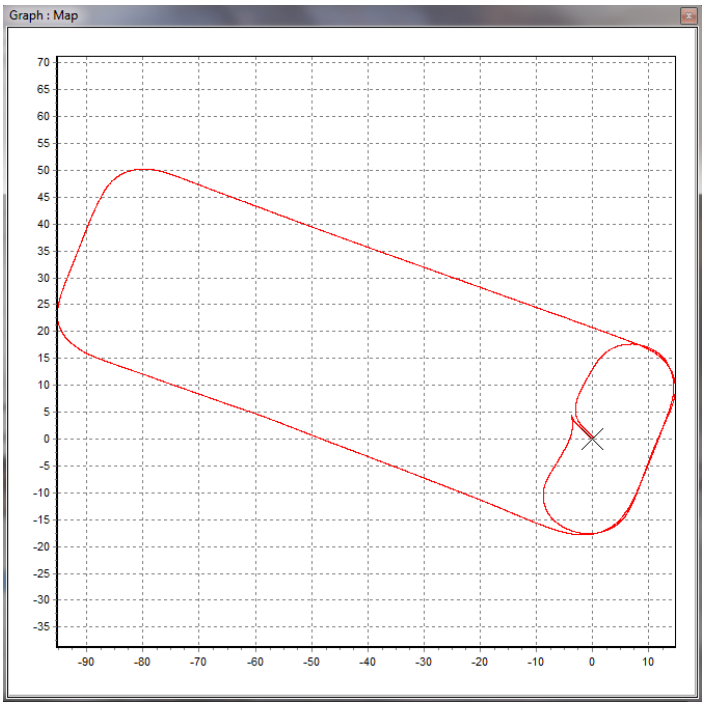

Figura 99: Recorrido en coche del VBOX 3i + IMU

Este nuevo recorrido es mucho más suave y recto que el anterior. No se aprecia que se haya producido algún tipo de error visible, llegando casi a coincidir el punto de inicio con el de fin por 20 centímetros.

Sobre Google Earth también parece que la traza se haya dibujado realmente bien:

Figura 100: Recorrido del VBOX 3i + IMU sobre Google Earth

El cambio en los datos más significativo lo encontramos en la gráfica de la velocidad, siendo mucho más suave que utilizando el VBOX 3i solo.

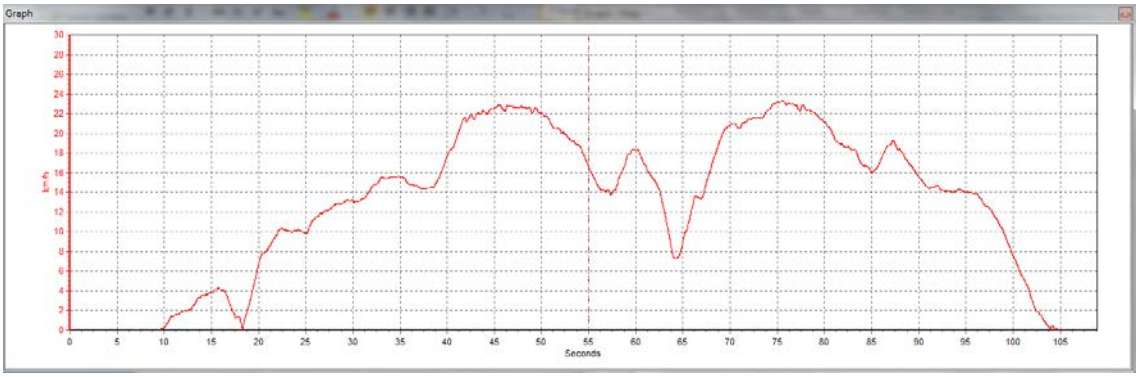

Figura 101: Velocidad del VBOX 3i + IMU

4.2.3 Caso del VBOX 3i con el DGPS:

Se montó la estación base y las antenas a la derecha de la zona de pruebas, atando la antena emisora a una farola para que se aguantase levantada.

En esta prueba se empezó desde otro lugar, casi justo debajo de la antena.

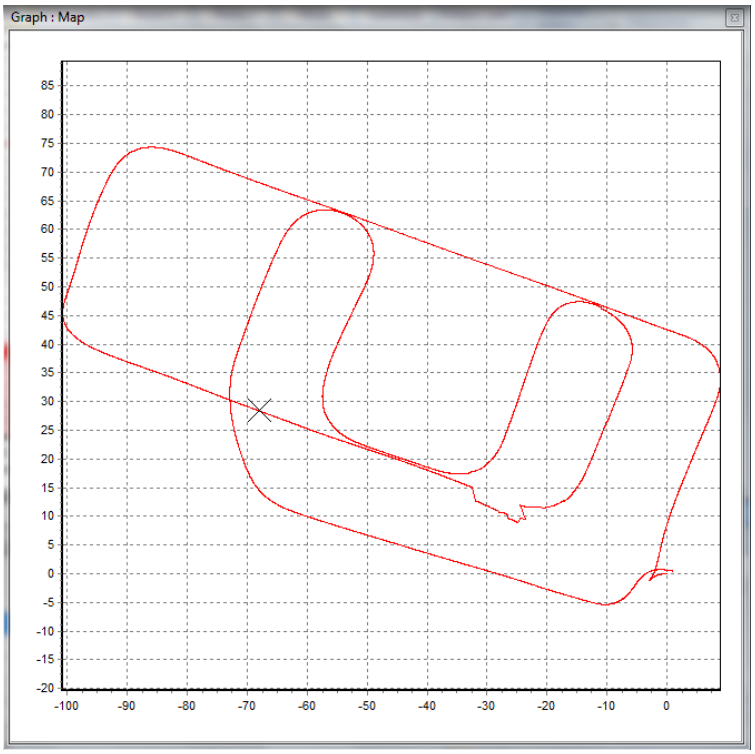

Figura 102: Recorrido del VBOX 3i con DGPS

El recorrido se ve muy suave y sin ruido. No se diferenciaría mucho de la IMU si no supiésemos que aquí contamos con un error de 2cm 95% CEP y por esa irregularidad a mitad del recorrido.

Con un vistazo rápido a los datos observamos que la señal del DGPS se cortó durante dos segundos.

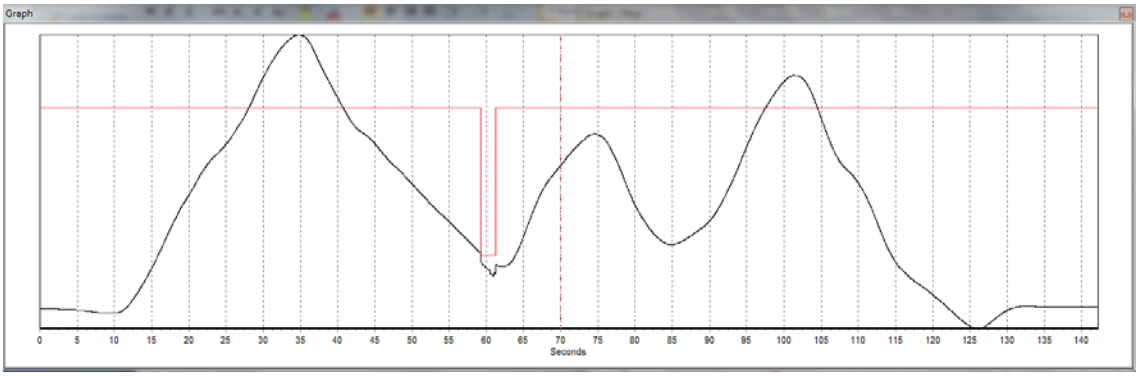

Figura 103: Señal DGPS y latitud del VBOX 3i + DGPS

Se aprecia como en el pequeño intervalo en que se pierde la señal DGPS la latitud da un salto situándose lejos de donde estaba unos instantes antes. Por suerte, nada mas vuelve el DGPS, la latitud vuelve a su estado normal.

Esto no parece que sea lo mismo que pasaba con el DGPS en la anterior prueba, cuando lo probábamos sin moverlo del sitio. En aquel caso el DGPS desaparecía solo un segundo y al volver, la latitud y la longitud tardaban en recuperar el estado en que se encontraban. Lo más probable es que algún árbol o furgoneta se haya interpuesto entre las antenas y se haya perdido la señal.

Esto puede ocurrir en cualquier momento debido a que las antenas siempre deben estar a la vista la una de la otra. Estos casos se podrían evitar utilizando la IMU a la vez que el DGPS, por lo que sería recomendable comprar el hardware que permite conectar ambos al mismo tiempo.

También se puede utilizar un filtro Kalman que evitará que la latitud y la longitud cambien mucho de valor durante el corto periodo de tiempo en que se ha perdido la señal, pero no será tan efectivo como una IMU ya que esta puede detectar incluso los giros.

Aparte, en la gráfica del tipo de solución podemos apreciar que después de que la señal DGPS se pierda ésta sufre un bajón.

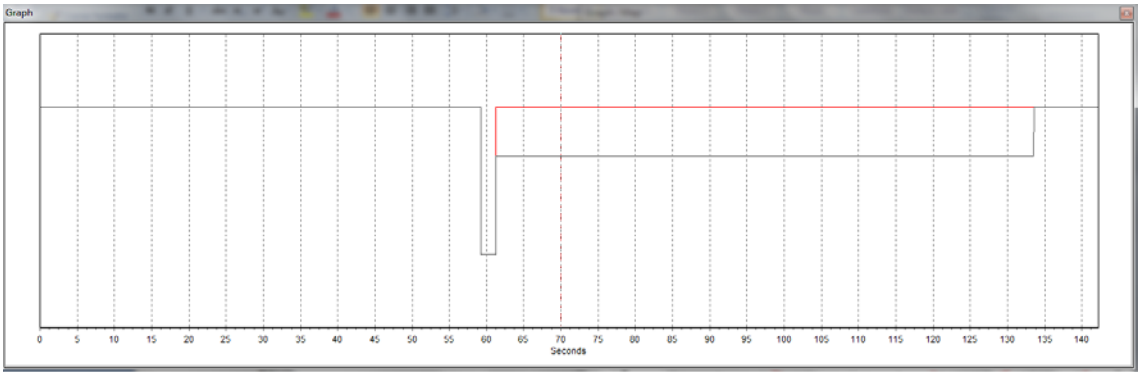

Figura 104: Señal DGPS y Tipo de solución del VBOX 3i + DGPS

Ambas señales van parejas hasta que ocurre la perdida de señal. Es entonces cuando el tipo de solución tiene valor 1, que significa que no se está utilizando el DGPS, solo los datos que llegan de la antena del VBOX 3i. Cuando se recupera, el tipo de solución toma valor 3 y después de un tiempo toma valor 4. Esto indica que cuando tipo de solución vale 3, la precisión que se está obteniendo es inferior a la de 2 cm 95% CEP.

El resto parece todo correcto excepto cuando llegamos a la velocidad, que nos encontramos con esto:

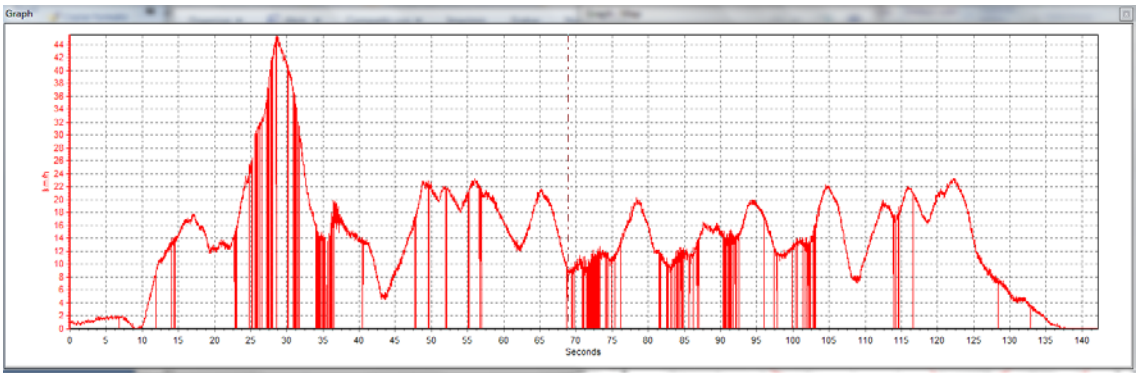

Figura 105: Velocidad del VBOX 3i + DGPS

Hay picos extraños que se reducen hasta el valor cero y luego vuelven a recuperar el valor original. Parece que haya cortes en la información que se recibe y solo ocurre cuando esta el DGPS encendido.

Por suerte esto no parece afectar a la latitud ni a la longitud por lo que el recorrido no se ve afectado por este error. No obstante, la aceleración de la longitud y la aceleración de la latitud si se ven afectadas por esto, bajando alguna de ellas a valor 0 a la vez que lo hace la velocidad.

Solo se me ocurren dos posibles causas para esto: la primera consiste en que cuando el DGPS está activado, el VBOX 3i debe hacer cálculos adicionales y el nivel de procesamiento no le llega para calcular y devolver todos los datos. Esto no debería ocurrir ya que en ningún lugar de la documentación advierten de que esto pueda suceder.

La otra razón es que, al igual que el DGPS fallaba cuando el GPS estaba quieto, aquí puede que también falle por lo que sería recomendable cambiar el material para ver si sigue fallando.

Considerando lo visto, consideraría utilizar el VBOX 3i con la opción del DGPS para un vehículo móvil autónomo solo en entornos controlados, ya que si pierde la señal podría resultar en un accidente. Con el uso de la IMU a la vez que el DGPS, este problema se solventa en gran parte, al igual que se solventa perder la señal de los satélites.

5. Conclusiones

Con el objetivo de mejorar la forma de viajar de las personas hemos acabado realizando un estudio exhaustivo de que tipos de GPS existen, surgiendo el VBOX 3i como un posible candidato para ser el sistema de posicionamiento de un vehículo móvil autónomo.

Los datos que este GPS devuelve pueden llegar a ser muy útiles, ya que además de la posición también podemos obtener otro tipo de datos como la velocidad o la aceleración. El vehículo debería poseer sensores que calculasen estos parámetros por su cuenta, por lo que contando con el VBOX 3i, poseeremos dos fuentes de datos que usaremos para comparar los valores obtenidos y corregir el posible error que se vaya acumulando en los sensores.

El VBOX 3i cuenta además con la IMU, un aparato que solventa durante un periodo de tiempo el problema más común de los GPS, que es el de encontrarse debajo de algo y no poder posicionar correctamente los satélites. No obstante esto solo lo hace durante un corto periodo de tiempo, por lo que habrá que utilizar otros sensores que tenga el vehículo para los casos en que no contemos con los satélites durante un periodo largo de tiempo.

Si bien las pruebas con el DGPS no han salido tan bien como se esperaba, si se consiguiese obtener siempre un error de 2cm, el VBOX sería una potente herramienta para que el vehículo pueda realizar maniobras de todo tipo, circular correctamente y coordinarse con otros vehículos con los que podamos comunicarnos y cuenten con sistemas similares.

Por ello propongo seguir con el VBOX 3i para intentar conseguir los resultados deseados. Para alcanzar estos objetivos podría sugerir cambiar el equipo, por si el nuestro es defectuoso, y en caso contrario pedir ayuda a expertos que nos indiquen que estamos haciendo mal.

Con las pruebas ya se ha visto que necesitamos un cielo despejado sin edificios para conseguir una buena precisión, aunque utilizando el DGPS solo pasamos de 2cm a 5cm de error en estos casos, lo cual considero que sigue siendo un margen aceptable.

En caso de no contar con una IMU, es aconsejable aplicar un filtro de Kalman para no obtener de repente datos incongruentes. En caso de contar con más sensores en el vehículo, podríamos utilizar los valores que estos proporcionen para crear nuestro propio filtro.

En cuanto a la librería de Linux, se ha creado en vistas a su uso futuro y cumple perfectamente con su misión de proporcionar la información que transmite el VBOX 3i en tiempo real. Lo único que habrá que ser cuidadoso con los parámetros que se introduzcan ya que de ellos depende el buen funcionamiento de la librería.

En general creo que se ha cumplido con todos los objetivos y que se debería seguir trabajando con el VBOX 3i para la realización de un vehículo móvil autónomo, teniendo en cuenta las recomendaciones que se han dado.

Además se podría seguir investigando para que se puedan utilizar el resto de métodos de transmisión, el USB y el Bluetooth, por si en un futuro resultan necesarios.

6. Bibliografía

Wikipedia: http://es.wikipedia.org/wiki/Wikipedia:Portada

Página y foros de Racelogic: http://www.racelogic.co.uk/

Información general del VBOX 3i:
http://www.racelogic.co.uk/_downloads/vbox/Datasheets/Data_Loggers/RLVB3i_Data.pdf

Manual de usuario del VBOX 3i:
http://www.racelogic.co.uk/_downloads/vbox/Manuals/Data_Loggers/RLVB3i_Manual%20-%20English.pdf

Manual de usuario de la IMU:
http://www.racelogic.co.uk/_downloads/vbox/Manuals/Input_Modules/RLVBIMU03_RLVBYAW03_Manual.pdf

Manual de usuario de la estación base:
http://www.racelogic.co.uk/_downloads/vbox/Manuals/Base_Stations/BaseStation_UserGuide.pdf

Manual de VBOXTools:
http://www.racelogic.co.uk/_downloads/vbox/Manuals/Software/VBOXTools%20Software%20Manual%20-%20English.pdf

Página web “Así funciona: GPS”:
http://www.asifunciona.com/electronica/af_gps/af_gps_13.htm

Página web “Mancuentro” sobre posicionamiento geográfico:
http://www.mancuentro.com/info/tipos_de_gps.html

7. Anexo

7.1. Anexo A – Guía de usuario del VBOX 3i

Manual del VBOX 3i adjunto al final del documento.

También se puede conseguir de esta dirección web:

http://www.racelogic.co.uk/_downloads/vbox/Manuals/Data_Loggers/RLVB3i_Manual%20-%20English.pdf

7.2. Anexo B – Guía de usuario de la IMU

Manual de la IMU (Unidad de Medida Inercial) adjunto al final del documento.

También se puede conseguir de esta dirección web:

http://www.racelogic.co.uk/_downloads/vbox/Manuals/Input_Modules/RLVBIMU03_RLVBYAW03_Manual.pdf

7.3. Anexo C – Guía de usuario del DGPS

Manual de la estación base del DGPS (GPS diferencial) adjunto al final del documento.

También se puede conseguir de esta dirección web:

http://www.racelogic.co.uk/_downloads/vbox/Manuals/Base_Stations/BaseStation_UserGuide.pdf

7.4. Anexo D – Manual de usuario del VBOXTools

Manual de la aplicación para Windows VBOXTools adjunto al final del documento.

También se puede conseguir de esta dirección web:

http://www.racelogic.co.uk/_downloads/vbox/Manuals/Software/VBOXTools%20Software%20Manual%20-%20English.pdf

7.5. Anexo E – Montaje y guía rápida

Montaje y configuración de los distintos componentes para su correcto funcionamiento.

Montaje del Vbox3i

-**Esto debe hacerse lo primero**: usando el cable con bordes dorados (figura 1), conectar la antena (figura 2) (quitando la funda roja protectora) al Vbox3i en la toma de arriba a la izquierda (figura 3). Es aconsejable situarla sobre una superficie metálica que imante (mejor calidad de datos).

Figura 1

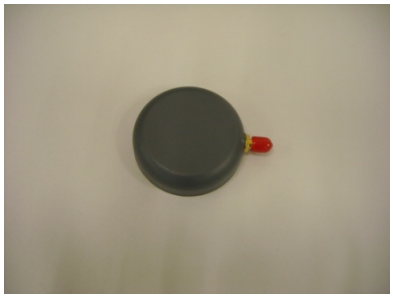

Figura 2

Figura 3

-Conectar el cable de alimentación (figura 4) al puerto etiquetado como PWR (figura 5) y el otro extremo a una toma de luz o batería. También se puede conectar al mechero de un coche usando otro cable (figura 6).

Figura 4

Figura 5

Figura 6

-Si se quiere grabar la información usaremos la Tarjeta Flash (figura 7). Se inserta en la ranura destinada a ello (figura 8). Según la configuración del GPS, se pondrá a grabar en ella inmediatamente o solo cuando haya movimiento. Si se aprieta la tecla LOG (figura 9) dejara de grabar cerrando el archivo, y si se pulsa LOG otra vez empezara a grabar en un archivo nuevo.

Figura 7

Figura 8

Figura 9

-Para monitorizar la información que está recogiendo el GPS en el PC y modificar su configuración hay tres posibles casos:

1. Serie: Conectar el cable serie (figura 10) a la ranura que pone SER (figura 11) y el otro extremo al ordenador. Se puede utilizar un conversor serie-USB si no se cuenta con puerto serie en el equipo.

Figura 10

Figura 11

2. USB: Conectar el cable miniusb-usb (figura 12) en su ranura (figura 13) y directamente al ordenador.

Figura 12

Figura 13

3. Bluetooth: Poner la antena Bluetooth (figura 14) e iniciar una conexión desde el ordenador.

Figura 14

Para más ayuda, ver las páginas 7 y 8 del manual del VBOX 3i en el Anexo A o las páginas 10 y 11 del manual del VBOXTools en el Anexo D.

Montaje de la IMU

-Conectar el cable (figura 15) a la ranura 1 de la IMU (figura 16) y el otro extremo a la ranura que dice CAN en el VBOX 3i (figura 17).

Figura 15

Figura 16

Figura 17

*Hace falta configurar el GPS mediante VBOXTools para que tenga en cuenta la IMU (explicado en la sección VBOXTools de este mismo anexo).

Para más ayuda, ver la página 7 del manual de la IMU en el Anexo B.

Montaje de la antena receptora del DGPS

-Conectar el cable con la caja azul y la antena (figura 18) en la ranura CAN (figura 17).

Figura 18

No se puede poner la IMU y la antena a la vez, para ello haría falta un hardware del que no disponemos.

*Hace falta configurar el GPS mediante VBOXTools para que tenga en cuenta la antena y use configuración diferencial (explicado en la sección VBOXTools de este mismo anexo).

Montaje de la estacion base

-Situar la antena blanca (figura 19) en el trípode (figura 20).

Figura 19

Figura 20

-Situar la estación base (figura 21) en un sitio estable y fijo.

Figura 21

-Conectar a la estación base la antena con el cable con conexiones plateadas (figura 22) a la ranura donde pone GPS (figura 23) (cuidado no hacerlo en el que pone Radio).

Figura 22

Figura 23

-Situar el mástil con la antena de transmisión (figura24) atada a un lugar estable.

Figura 24

-Conectar el cable con conexiones azuladas (figura 25) al mástil y a la ranura del medio de la estación base.

Figura 25

-Encender con el interruptor (figura 26).

Figura 26

-Esperar 60 segundos.

-Esperar hasta que encuentre satélites.

-Darle al botón verde OK (figura 27).

Figura 27

-Presionar otra vez el botón OK para entrar en el panel de configuración, moverse con las flechas hasta “Set to Average”, darle a OK y seleccionar 5 minutos pulsando OK otra vez. (se puede seleccionar más tiempo si se quiere)

-Pasado el tiempo, se le da a OK y se sale del menú de configuración seleccionando EXIT y pulsando OK.

-Para que funcione correctamente con el VBOX 3i y el hardware comprado, en el menú configuración debe tener esta configuración:

Configuración DGPS
Handshaking: ON
DGPS Baud: 9600
Radio mode: User
Set DGPS Mode: Racelogic 2cm

Para más ayuda, mirar la página 12 del manual de la estación base en el Anexo C.

Guía rápida del software VBOXTools

Instalación

-Insertar el CD y clica setup.exe.

-Lo instalara todo pero puede tener problemas con los drivers.

- Si son los drivers del VBOX 3i, buscarlos en el CD cuando diga que no los encuentra
- Si son del cable serie-usb (si es que se está usando), bajarlos de esta página en el caso de utilizar Windows7: http://www.usbfirewire.com/Parts/rr-usb-017.html

Para más ayuda, mirar en la página 9 del manual del VBOXTool en el Anexo D.

Ejecución

-Al pedir que se registre el producto, pulsar “Register Later”, a menos que se cuente con el código para registrarse.

-Pulsar F11 o el botón Offline.

-Si sale una ventana diciendo que no encuentra el puerto COM, ir al menú Options -> COM Port y selecciona el que esté conectado el VBOX 3i.

-Botones de interés (una vez online):

- Real time plot: Cuarto botón por la izquierda, muestra un mapa del camino recorrido por el GPS.
- New data display: Quinto botón por la izquierda, para añadir una nueva ventana de información. Clicando con el botón derecho sobre la ventana se puede cambiar el tipo de información en Main Data -> Standard Channels o Main Data -> Can Channels.
- Terminal: Séptimo botón por la izquierda, te muestra la traza en ASCII o hexadecimal.
- Vbox Setup: Sexto botón por la izquierda, abre el menú de configuración del GPS.

Para más ayuda, mirar el apartado “3.2.1 VBOXTools – Monitorización”, o en la página 9 del manual del VBOXTool en el Anexo D.

Configuración (añadir/quitar IMU o DGPS)

-Pulsando el sexto botón por la derecha, el de Vbox Setup, se abre la ventana de configuración.

-En la pestaña Standard del menú Channels seleccionamos la información que devuelve el GPS.

-Caso de querer poner la **IMU** (si se quiere quitar, realizar los pasos de forma inversa):

- Conectar la IMU según se ha visto en las instrucciones de montaje de este mismo anexo.

- En el menú Channels, aparecerá la pestaña 3 axis modules (entrando en el menú de configuración después de haberlo conectado). Y seleccionar los datos que queremos.
- En el menú GPS, seleccionar la casilla que pone “Use IMU”.
- Clicar en Close Set-Up para que se aplique la configuración.

-Caso de querer activar el **DGPS** (si se quiere quitar, seguir los pasos de forma inversa):

- Conectar la antena e instalar la estación base según los instrucciones de montaje de este mismo anexo.
- Ir al menú GPS y en el cuadrado DGPS, en Mode, seleccionar “Racelogic (2cm RTK)”.
- Darle a Close Set-Up para que se aplique la configuración.

Para más ayuda en cuanto al IMU, consultar las páginas 86 y 87 del manual de VBOXTools en el Anexo D.

Para más ayuda en cuanto al DGPS, consultar la página 24 del manual de VBOXTools en el Anexo D.

7.6. Anexo F – Código de la librería gps.h y gps.c

Códigos de la librería realizada para este proyecto gps.h y gps.c.

Incluimos aquí el código de la librería, con comentarios que explican el funcionamiento de esta.

gps.h

```
#include <stdio.h>
#include <fcntl.h>
#include <termios.h>
#include <errno.h>
#include <string.h>
#include <stdbool.h>
#include <unistd.h>
#include <time.h>

#define VS     0x00000001     //El GPS devuelve la Vertical Speed
#define LONGACC       0x00000010     //El GPS devuelve la GPS LongAcc
#define LATACC        0x00000100     //El GPS devuelve la GPS LatAcc
#define GLSAT  0x00001000     //El GPS devuelve el numero de Glonass Satellites
#define GPSSAT 0x00010000     //El GPS devuelve el numero de GPS Satellites
#define SQ     0x00100000     //El GPS devuelve la Speed Quality
#define ST     0x01000000     //El GPS devuelve la Solution Type
#define IKFS   0x10000000     //El GPS devuelve el IMU Kalman Filter Status

#define TAM_TRAZA_BASICA 60   //Tamaño de la traza basica
#define TAM_TRAZA_MAXIMA 104  //Tamaño maximo que puede tener la traza
#define TAM_VS 2              //Tamaño en la traza de la Vertical Speed
#define TAM_LONGACC 2         //Tamaño en la traza del GPS LongAcc
#define TAM_LATACC 2          //Tamaño en la traza del GPS LatAcc
#define TAM_GLSAT 1           //Tamaño en la traza del numero de Glonass Satellites
#define TAM_GPSSAT 1          //Tamaño en la traza del numero de GPS Satellites
#define TAM_SQ 4              //Tamaño en la traza de la Speed Quality
#define TAM_ST 2              //Tamaño en la traza del Solution Type
#define TAM_IKFS 2            //Tamaño en la tarza del IMU Kalman Filter Status

#define YAW    0x00000001     //El GPS devuelve el giro en el eje Z obtenida con la IMU
#define XACEL  0x00000010     //El GPS devuelve la aceleracion en X obtenida con la IMU
#define YACEL  0x00000100     //El GPS devuelve la aceleracion en Y obtenida con la IMU
#define TEMP   0x00001000     //El GPS devuelve la temperatura obtenida con la IMU
#define PITCH  0x00010000     //El GPS devuelve el giro en el eje Y obtenida con la IMU
#define ROLL   0x00100000     //El GPS devuelve el giro en el eje X obtenida con la IMU
#define ZACEL  0x01000000     //El GPS devuelve la aceleracion en Z obtenida con la IMU

#define TAM_YAW 4             //Tamaño en la traza del giro en el eje Z
#define TAM_XACEL 4           //Tamaño en la traza de la aceleracion en X
#define TAM_YACEL 4           //Tamaño en la traza de la aceleracion en Y
#define TAM_TEMP 4            //Tamaño en la traza de la temperatura
#define TAM_PITCH 4           //Tamaño en la traza del giro en el eje Y
#define TAM_ROLL 4            //Tamaño en la traza del giro en el eje X
#define TAM_ZACEL 4           //Tamaño en la traza de la aceleracion en Z


static int tam_traza;         //Tamaño en bytes de la traza, aumentara dependiendo de
configuracion y configuracion_imu
static double fr;                           //1/Frecuencia de muestreo del VBOX 3i
static int configuracion;                   //Define que campos opcionales transmite el
VBOX 3i por el puerto serie
static int configuracion_imu;               //Define que campos opcionales transmite el
VBOX 3i por el puerto serie procedentes de la IMU

typedef struct
{
       unsigned int traza[TAM_TRAZA_MAXIMA];       //Informacion que transmite el GPS
       int tiempo;            //Esta en decimas de segundo
       int satelites;         //El numero de satelites
       double latitud;               //Esta en minutos*100000
       double longitud;       //Esta en minutos*100000
       double velocidad;      //Esta en km/h*100
       double direccion;      //Esta en grados*100
       double altura;         //Esta en metros*100
       double vvertical;      //Velocidad vertical en km/h*100
       double plongitud;      //Precision de la longitud en fuerza g * 10
       double platitud;       //Precision de la latitud en fuerza g * 10
       int glsatelites;       //El numero de satelites glonass
       int gpssatelites;      //El numero de satelites gps
       int estadokalman;      //Numero que indica el estado del filtro
       int tiposolucion;      //No Data = -1, No solution = 0, Stand Alone >= 1 | -1 = No
se reciben datos, 0 = sin satelites, 1 = todo bien, >1 = DGPS)
       double vcalidad;       //Esta en km/h*100
```

```
        double yawrate;              //Esta en grados/s
        double acelx;         //Esta en fuerza g
        double acely;         //Esta en fuerza g
        double temperatura;   //Esta en grados Centigrados
        double pitchrate;     //Esta en grados/s
        double rollrate;      //Esta en grados/s
        double acelz;         //Esta en fuerza g
} datos;

int getTime(unsigned int t[tam_traza]);                    //Coge el tiempo de la traza
y lo inserta en la estructura datos
int getSats(unsigned int t[tam_traza]);                    //Coge el numero de satelites
de lo traza y la inserta en la estructura datos
double getLat(unsigned int t[tam_traza]);          //Coge la latitud de la traza y la
inserta en la estructura datos
double getLon(unsigned int t[tam_traza]);          //Coge la longitud de la traza y la
inserta en la estructura datos
double getVel(unsigned int t[tam_traza]);          //Coge la velocidad de la traza y la
inserta en la estructura datos
double getHead(unsigned int t[tam_traza]);         //Coge la direccion de la traza y la
inserta en la estructura datos
double getAlt(unsigned int t[tam_traza]);          //Coge la altura de la traza y la
inserta en la estructura datos
double getVVel(unsigned int t[tam_traza]);         //Coge la velocidad vertical de la
traza y la inserta en la estructura datos
double getLongAcc(unsigned int t[tam_traza]);              //Coge la precision de la
longitud de la traza y la inserta en la estructura datos
double getLatAcc(unsigned int t[tam_traza]);       //Coge la precision de la latidud de
la traza y la inserta en la estructura datos
int getGlSat(unsigned int t[tam_traza]);           //Coge el numero de satelites
glonass de la traza y lo inserta en la estructura datos
int getGpsSat(unsigned int t[tam_traza]);          //Coge el numero de satelites GPS de
la traza y lo inserta en la estructura datos
int getIKFS(unsigned int t[tam_traza]);                    //Coge el estado del filtro
Kalman de la traza y lo inserta en la estructura datos
int getST(unsigned int t[tam_traza]);                      //Coge el tipo de solucion de
la traza y lo inserta en la estructura datos
double getSQ(unsigned int t[tam_traza]);           //Coge la calidad de la velocidad de
la traza y la inserta en la estructura datos
double exponent(double n, double e);               //Devuelve el valor n elevado a e
double comaflotante2double(unsigned int t0, unsigned int t1, unsigned int t2, unsigned
int t3);       //Pasa de coma flotante a decimal
double getYAW(unsigned int t[tam_traza]);          //Coge el yawrace y lo inserta en la
estructura de datos
double getROLL(unsigned int t[tam_traza]);         //Coge el rollrace y lo inserta en
la estructura de datos
double getPITCH(unsigned int t[tam_traza]);        //Coge el pitchrace y lo inserta en
la estructura de datos
double getXACEL(unsigned int t[tam_traza]);        //Coge la aceleracion en X y la
inserta en la estructura de datos
double getYACEL(unsigned int t[tam_traza]);        //Coge la aceleracion en Y y la
inserta en la estructura de datos
double getZACEL(unsigned int t[tam_traza]);        //Coge la aceleracion en Z y la
inserta en la estructura de datos
double getTEMP(unsigned int t[tam_traza]);         //Coge la temperatura y la inserta
en la estructura de datos
void printTrace(unsigned int t[tam_traza]);        //Imprime la traza por pantalla
void printTime (int tiempo);                       //Imprime el tiempo por pantalla
void printSats (int satelites);                            //Imprime el numero de
satelites por pantalla
void printLat (double latitud);                            //Imprime la latitud por
pantalla
void printLon (double longitud);                   //Imprime la longitud por pantalla
void printVel (double velocidad);                  //Imprime la velocidad por pantalla
void printHead (double direccion);                 //Imprime la direccion por pantalla
void printAlt (double altura);                             //Imprime la altura por
pantalla
void printVVel (double vvertical);                 //Imprime la velocidad vertical por
pantalla
void printLongAcc(double plongitud);               //Imprime la precision en la
longitud por pantalla
void printLatAcc (double platitud);                //Imprime la precision en la latitud
por pantalla
void printGlSat (int glsatelites);                 //Imprime el numero de satelites
glonass por pantalla
void printGpsSat (int gpssatelites);               //Imprime el numero de satelites GPS
por pantalla
```

```
void printIKFS (int estadokalman);                  //Imprime el estado del filtro
Kalman por pantalla
void printST (int tiposolucion);                    //Imprime el tipo de solucion por
pantalla
void printSQ (double vcalidad);                             //Imprime la calidad de la
velocidad por pantalla
void printYAW (double yawrate);                             //Imprime la yawrate por
pantalla
void printPITCH (double pitchrate);                 //Imprime la pitchrace por pantalla
void printROLL (double rollrate);                   //Imprime la rollrate por pantalla
void printXACEL (double acelx);                             //Imprime la aceleracion en X
por pantalla
void printYACEL (double acely);                             //Imprime la aceleracion en Y
por pantalla
void printZACEL (double acelz);                             //Imprime la aceleracion en Z
por pantalla
void printTEMP (double temperatura);        //Imprime la temperatura por pantalla
int EncontrarAbrirPuerto(int conf, int conf_imu, double hz);      //Encuentra el puerto
al que esta conectado el GPS, lo abre y lo sincroniza
void EsperarSiguienteTraza(int fd, double MUESTREO);              //Espera un tiempo
MUESTREO para volver a coger la siguiente traza
datos CogerTraza(int fd);                                   //Devuelve una estructura de
datos con la informacion de la ultima traza transmitida por el GPS
void ImprimirDatos(datos d);                                //Imprime los datos de la
estructura
```

gps.c

```
#include "gps.h"

int getTime(unsigned int t[tam_traza])
{	//El tiempo esta en 3 bytes de la traza
	return t[18]*256*256+t[19]*256+t[20];
}

int getSats(unsigned int t[tam_traza])
{	//El numero de satelites esta en 1 byte de la traza
	return t[17];
}

double getLat(unsigned int t[tam_traza])
{	//La latitud esta en 4 bytes de la traza
	return  t[21]*256*256*256+t[22]*256*256+t[23]*256+t[24];
}

double getLon(unsigned int t[tam_traza])
{	//La longitud esta en 4 bytes de la traza
	return t[25]*256*256*256+t[26]*256*256+t[27]*256+t[28];
}

double getVel(unsigned int t[tam_traza])
{	//La velocidad esta en 2 bytes de la traza
	if (fr >= 0.02)
		return t[29]*256+t[30];
	else
		return t[21]*256+t[22];
}

double getHead(unsigned int t[tam_traza])
{	//La direccion esta en 2 bytes de la traza
	return t[31]*256+t[32];
}

double getAlt(unsigned int t[tam_traza])
{	//La altura esta en 3 bytes de la traza
	return t[33]*256*256+t[34]*256+t[35];
}

double getVVel(unsigned int t[tam_traza])
{	//La velocidad vertical esta en 2 bytes de la traza
	if (t[36] < 128)	//Si el bit de mayor peso es 0, el valor es positivo
		return t[36]*256+t[37];
	else			//Si el bit de mayor peso es 1, el valor es negativo,
teniendo que invertir los bits
	{
		double aux = t[36]*256+t[37];
		aux = aux - 65536;
		return aux;
	}
}

double getLongAcc(unsigned int t[tam_traza])
{	//La precision en la longitud esta en 2 bytes de la traza
	int pos = 36;
	if (configuracion & VS)		//Si VS esta activado, los bytes estaran
corridos TAM_VS posiciones a la derecha
		pos += TAM_VS;
	if (t[pos] < 128)	//Si el bit de mayor peso es 0, el valor es positivo
		return t[pos]*256+t[pos+1];
	else			//Si el bit de mayor peso es 1, el valor es negativo,
teniendo que invertir los bits
	{
		double aux = t[pos]*256+t[pos+1];
		aux = aux - 65536;
		return aux;
	}
}

double getLatAcc(unsigned int t[tam_traza])
{	//La precision de la latitud esta en 2 bytes de la traza
	int pos = 36;
```

```
        if (configuracion & VS)             //Si VS esta activado, los bytes estaran
corridos TAM_VS posiciones a la derecha
                pos += TAM_VS;
        if (configuracion & LONGACC)  //Si LONGACC esta activado, los bytes estaran
corridos TAM_LONGACC posiciones a la derecha
                pos += TAM_LONGACC;
        if (t[pos] < 128)     //Si el bit de mayor peso es 0, el valor es positivo
                return t[pos]*256+t[pos+1];
        else                  //Si el bit de mayor peso es 1, el valor es negativo,
teniendo que invertir los bits
        {
                double aux = t[pos]*256+t[pos+1];
                aux = aux - 65536;
                return aux;
        }
}

int getGlSat(unsigned int t[tam_traza])
{       //El numero de satelites glonass esta en 1 byte de la traza
        int pos = 36;
        if (configuracion & VS)             //Si VS esta activado, los bytes estaran
corridos TAM_VS posiciones a la derecha
                pos += TAM_VS;
        if (configuracion & LONGACC)  //Si LONGACC esta activado, los bytes estaran
corridos TAM_LONGACC posiciones a la derecha
                pos += TAM_LONGACC;
        if (configuracion & LATACC)   //Si LATACC esta activado, los bytes estaran
corridos TAM_LATACC posiciones a la derecha
                pos += TAM_LATACC;
        return t[pos];
}

int getGpsSat(unsigned int t[tam_traza])
{       //El numero de satelites GPS esta en 1 byte de la traza
        int pos = 36;
        if (configuracion & VS)             //Si VS esta activado, los bytes estaran
corridos TAM_VS posiciones a la derecha
                pos += TAM_VS;
        if (configuracion & LONGACC)  //Si LONGACC esta activado, los bytes estaran
corridos TAM_LONGACC posiciones a la derecha
                pos += TAM_LONGACC;
        if (configuracion & LATACC)   //Si LATACC esta activado, los bytes estaran
corridos TAM_LATACC posiciones a la derecha
                pos += TAM_LATACC;
        if (configuracion & GLSAT)    //Si GLSAT esta activado, los bytes estaran
corridos TAM_GLSAT posiciones a la derecha
                pos += TAM_GLSAT;
        return t[pos];
}

int getIKFS(unsigned int t[tam_traza])
{       //El estado del filtro Kalman esta en 2 bytes de la traza
        int pos = 36;
        if (configuracion & VS)             //Si VS esta activado, los bytes estaran
corridos TAM_VS posiciones a la derecha
                pos += TAM_VS;
        if (configuracion & LONGACC)  //Si LONGACC esta activado, los bytes estaran
corridos TAM_LONGACC posiciones a la derecha
                pos += TAM_LONGACC;
        if (configuracion & LATACC)   //Si LATACC esta activado, los bytes estaran
corridos TAM_LATACC posiciones a la derecha
                pos += TAM_LATACC;
        if (configuracion & GLSAT)    //Si GLSAT esta activado, los bytes estaran
corridos TAM_GLSAT posiciones a la derecha
                pos += TAM_GLSAT;
        if (configuracion & GPSSAT)   //Si GPSSAT esta activado, los bytes estaran
corridos TAM_GPSSAT posiciones a la derecha
                pos += TAM_GPSSAT;
        return t[pos]*256+t[pos+1];
}

int getST(unsigned int t[tam_traza])
{       //El tipo de solucion esta en 2 bytes de la traza
        int pos = 36;
        if (configuracion & VS)             //Si VS esta activado, los bytes estaran
corridos TAM_VS posiciones a la derecha
                pos += TAM_VS;
```

```
	if (configuracion & LONGACC)  //Si LONGACC esta activado, los bytes estaran
corridos TAM_LONGACC posiciones a la derecha
		pos += TAM_LONGACC;
	if (configuracion & LATACC)   //Si LATACC esta activado, los bytes estaran
corridos TAM_LATACC posiciones a la derecha
		pos += TAM_LATACC;
	if (configuracion & GLSAT)    //Si GLSAT esta activado, los bytes estaran
corridos TAM_GLSAT posiciones a la derecha
		pos += TAM_GLSAT;
	if (configuracion & GPSSAT)   //Si GPSSAT esta activado, los bytes estaran
corridos TAM_GPSSAT posiciones a la derecha
		pos += TAM_GPSSAT;
	if (configuracion & IKFS)     //Si IKFS esta activado, los bytes estaran corridos
TAM_IKFS posiciones a la derecha
		pos += TAM_IKFS;
	return t[pos]*256+t[pos+1];
}

double getSQ(unsigned int t[tam_traza])
{	//La calidad de la velocidad esta en 4 bytes de la traza
	int pos = 36;
	if (configuracion & VS)              //Si VS esta activado, los bytes estaran
corridos TAM_VS posiciones a la derecha
		pos += TAM_VS;
	if (configuracion & LONGACC)  //Si LONGACC esta activado, los bytes estaran
corridos TAM_LONGACC posiciones a la derecha
		pos += TAM_LONGACC;
	if (configuracion & LATACC)   //Si LATACC esta activado, los bytes estaran
corridos TAM_LATACC posiciones a la derecha
		pos += TAM_LATACC;
	if (configuracion & GLSAT)    //Si GLSAT esta activado, los bytes estaran
corridos TAM_GLSAT posiciones a la derecha
		pos += TAM_GLSAT;
	if (configuracion & GPSSAT)   //Si GPSSAT esta activado, los bytes estaran
corridos TAM_GPSSAT posiciones a la derecha
		pos += TAM_GPSSAT;
	if (configuracion & IKFS)     //Si IKFS esta activado, los bytes estaran corridos
TAM_IKFS posiciones a la derecha
		pos += TAM_IKFS;
	if (configuracion & ST)              //Si ST esta activado, los bytes estaran
corridos TAM_ST posiciones a la derecha
		pos += TAM_ST;
	return t[pos]*256*256*256+t[pos+1]*256*256+t[pos+2]*256+t[pos+3];
}

double exponent(double n, double e)
{ //devuelve n elevado a e
	int i, res = 1;
	for (i = 0; i < e; i++)
		res = res * n;
	return res;
}

double comaflotante2double (unsigned int t0, unsigned int t1, unsigned int t2, unsigned
int t3)
{
	double resultado = 0;
	int negativo = 0;
	int exponente = 0;
	int significante = 0;
	int vector[24];
	int i, contador;
	unsigned int aux = t1;
	if (t0 >= 128)
		negativo = 1;                //Si el primer bit es 1, significa que el
resultado sera negativo
	if (negativo == 1)
		exponente = (t0 - 128) * 2;  //Quitamos el primer bit y movemos el resto
una posicion a la izquierda
	else
		exponente = t0 * 2;
	if (t1 >= 128)
	{
		exponente = exponente + 1;   //El bit mas a la derecha de t1 es el bit
mas a la derecha de exponente
		aux = t1 - 128;                      //Eliminamos el bit mas a la
izquierda
```

```
	}
	significante = t1*256*256 + t2*256 + t3;
	if (exponente == 0)
		return significante;		//Caso para el 0 o numeros desnormalizados
	exponente = exponente - 127;
	vector[0] = 1;
	for (i = 23; i > 0; i--) //se pasa de decimal a binario -> rellenamos un vector
con unos y ceros
	{
		if (significante % 2 == 0)
			vector[i] = 0;
		else
			vector[i] = 1;
		significante = significante / 2;
	}
	contador = 0;
	for (;exponente >= 0; exponente--)
	{
		resultado = resultado + (exponent(2,exponente)*vector[contador]);
		contador++;
	}
	for (;contador < 24; contador++)
	{
		resultado = resultado + (vector[contador]/exponent(2,0-exponente));
		exponente--;
	}
	if (negativo == 1)
		resultado = 0 - resultado;
	return resultado;
}

double getYAW(unsigned int t[tam_traza])
{	//La inclinacion Yaw esta en 4 bytes de la traza en formato coma flotante
	int pos = 36;
	if (configuracion & VS)			//Si VS esta activado, los bytes estaran
corridos TAM_VS posiciones a la derecha
		pos += TAM_VS;
	if (configuracion & LONGACC)  //Si LONGACC esta activado, los bytes estaran
corridos TAM_LONGACC posiciones a la derecha
		pos += TAM_LONGACC;
	if (configuracion & LATACC)   //Si LATACC esta activado, los bytes estaran
corridos TAM_LATACC posiciones a la derecha
		pos += TAM_LATACC;
	if (configuracion & GLSAT)    //Si GLSAT esta activado, los bytes estaran
corridos TAM_GLSAT posiciones a la derecha
		pos += TAM_GLSAT;
	if (configuracion & GPSSAT)   //Si GPSSAT esta activado, los bytes estaran
corridos TAM_GPSSAT posiciones a la derecha
		pos += TAM_GPSSAT;
	if (configuracion & IKFS)     //Si IKFS esta activado, los bytes estaran corridos
TAM_IKFS posiciones a la derecha
		pos += TAM_IKFS;
	if (configuracion & ST)			//Si ST esta activado, los bytes estaran
corridos TAM_ST posiciones a la derecha
		pos += TAM_ST;
	if (configuracion & SQ)			//Si SQ esta activado, los bytes estaran
corridos TAM_SQ posiciones a la derecha
		pos += TAM_SQ;
	pos += 22;
	return comaflotante2double(t[pos], t[pos+1], t[pos+2], t[pos+3]);
}

double getXACEL(unsigned int t[tam_traza])
{	//La aceleracion en X esta en 4 bytes de la traza en formato coma flotante
	int pos = 36;
	if (configuracion & VS)			//Si VS esta activado, los bytes estaran
corridos TAM_VS posiciones a la derecha
		pos += TAM_VS;
	if (configuracion & LONGACC)  //Si LONGACC esta activado, los bytes estaran
corridos TAM_LONGACC posiciones a la derecha
		pos += TAM_LONGACC;
	if (configuracion & LATACC)   //Si LATACC esta activado, los bytes estaran
corridos TAM_LATACC posiciones a la derecha
		pos += TAM_LATACC;
	if (configuracion & GLSAT)    //Si GLSAT esta activado, los bytes estaran
corridos TAM_GLSAT posiciones a la derecha
		pos += TAM_GLSAT;
```

```
        if (configuracion & GPSSAT)   //Si GPSSAT esta activado, los bytes estaran
corridos TAM_GPSSAT posiciones a la derecha
                pos += TAM_GPSSAT;
        if (configuracion & IKFS)     //Si IKFS esta activado, los bytes estaran corridos
TAM_IKFS posiciones a la derecha
                pos += TAM_IKFS;
        if (configuracion & ST)               //Si ST esta activado, los bytes estaran
corridos TAM_ST posiciones a la derecha
                pos += TAM_ST;
        if (configuracion & SQ)               //Si SQ esta activado, los bytes estaran
corridos TAM_SQ posiciones a la derecha
                pos += TAM_SQ;
        pos += 22;
        if (configuracion_imu & YAW)//Si YAW esta activado, los bytes estaran corridos
TAM_YAW posiciones a la derecha
                pos += TAM_YAW;
        return comaflotante2double(t[pos], t[pos+1], t[pos+2], t[pos+3]);
}

double getYACEL(unsigned int t[tam_traza])
{       //La aceleracion en Y esta en 4 bytes de la traza en formato coma flotante
        int pos = 36;
        if (configuracion & VS)               //Si VS esta activado, los bytes estaran
corridos TAM_VS posiciones a la derecha
                pos += TAM_VS;
        if (configuracion & LONGACC)  //Si LONGACC esta activado, los bytes estaran
corridos TAM_LONGACC posiciones a la derecha
                pos += TAM_LONGACC;
        if (configuracion & LATACC)   //Si LATACC esta activado, los bytes estaran
corridos TAM_LATACC posiciones a la derecha
                pos += TAM_LATACC;
        if (configuracion & GLSAT)    //Si GLSAT esta activado, los bytes estaran
corridos TAM_GLSAT posiciones a la derecha
                pos += TAM_GLSAT;
        if (configuracion & GPSSAT)   //Si GPSSAT esta activado, los bytes estaran
corridos TAM_GPSSAT posiciones a la derecha
                pos += TAM_GPSSAT;
        if (configuracion & IKFS)     //Si IKFS esta activado, los bytes estaran corridos
TAM_IKFS posiciones a la derecha
                pos += TAM_IKFS;
        if (configuracion & ST)               //Si ST esta activado, los bytes estaran
corridos TAM_ST posiciones a la derecha
                pos += TAM_ST;
        if (configuracion & SQ)               //Si SQ esta activado, los bytes estaran
corridos TAM_SQ posiciones a la derecha
                pos += TAM_SQ;
        pos += 22;
        if (configuracion_imu & YAW)//Si YAW esta activado, los bytes estaran corridos
TAM_YAW posiciones a la derecha
                pos += TAM_YAW;
        if (configuracion_imu & XACEL)//Si XACEL esta activado, los bytes estaran
corridos TAM_XACEL posiciones a la derecha
                pos += TAM_XACEL;
        return comaflotante2double(t[pos], t[pos+1], t[pos+2], t[pos+3]);
}

double getTEMP(unsigned int t[tam_traza])
{       //La temperatura esta en 4 bytes de la traza en formato coma flotante
        int pos = 36;
        if (configuracion & VS)               //Si VS esta activado, los bytes estaran
corridos TAM_VS posiciones a la derecha
                pos += TAM_VS;
        if (configuracion & LONGACC)  //Si LONGACC esta activado, los bytes estaran
corridos TAM_LONGACC posiciones a la derecha
                pos += TAM_LONGACC;
        if (configuracion & LATACC)   //Si LATACC esta activado, los bytes estaran
corridos TAM_LATACC posiciones a la derecha
                pos += TAM_LATACC;
        if (configuracion & GLSAT)    //Si GLSAT esta activado, los bytes estaran
corridos TAM_GLSAT posiciones a la derecha
                pos += TAM_GLSAT;
        if (configuracion & GPSSAT)   //Si GPSSAT esta activado, los bytes estaran
corridos TAM_GPSSAT posiciones a la derecha
                pos += TAM_GPSSAT;
        if (configuracion & IKFS)     //Si IKFS esta activado, los bytes estaran corridos
TAM_IKFS posiciones a la derecha
                pos += TAM_IKFS;
```

```
        if (configuracion & ST)             //Si ST esta activado, los bytes estaran
corridos TAM_ST posiciones a la derecha
                pos += TAM_ST;
        if (configuracion & SQ)             //Si SQ esta activado, los bytes estaran
corridos TAM_SQ posiciones a la derecha
                pos += TAM_SQ;
        pos += 22;
        if (configuracion_imu & YAW)//Si YAW esta activado, los bytes estaran corridos
TAM_YAW posiciones a la derecha
                pos += TAM_YAW;
        if (configuracion_imu & XACEL)//Si XACEL esta activado, los bytes estaran
corridos TAM_XACEL posiciones a la derecha
                pos += TAM_XACEL;
        if (configuracion_imu & YACEL)//Si YACEL esta activado, los bytes estaran
corridos TAM_YACEL posiciones a la derecha
                pos += TAM_YACEL;
        return comaflotante2double(t[pos], t[pos+1], t[pos+2], t[pos+3]);
}

double getPITCH(unsigned int t[tam_traza])
{       //La pitch rate esta en 4 bytes de la traza en formato coma flotante
        int pos = 36;
        if (configuracion & VS)             //Si VS esta activado, los bytes estaran
corridos TAM_VS posiciones a la derecha
                pos += TAM_VS;
        if (configuracion & LONGACC)  //Si LONGACC esta activado, los bytes estaran
corridos TAM_LONGACC posiciones a la derecha
                pos += TAM_LONGACC;
        if (configuracion & LATACC)   //Si LATACC esta activado, los bytes estaran
corridos TAM_LATACC posiciones a la derecha
                pos += TAM_LATACC;
        if (configuracion & GLSAT)    //Si GLSAT esta activado, los bytes estaran
corridos TAM_GLSAT posiciones a la derecha
                pos += TAM_GLSAT;
        if (configuracion & GPSSAT)   //Si GPSSAT esta activado, los bytes estaran
corridos TAM_GPSSAT posiciones a la derecha
                pos += TAM_GPSSAT;
        if (configuracion & IKFS)     //Si IKFS esta activado, los bytes estaran corridos
TAM_IKFS posiciones a la derecha
                pos += TAM_IKFS;
        if (configuracion & ST)             //Si ST esta activado, los bytes estaran
corridos TAM_ST posiciones a la derecha
                pos += TAM_ST;
        if (configuracion & SQ)             //Si SQ esta activado, los bytes estaran
corridos TAM_SQ posiciones a la derecha
                pos += TAM_SQ;
        pos += 22;
        if (configuracion_imu & YAW)//Si YAW esta activado, los bytes estaran corridos
TAM_YAW posiciones a la derecha
                pos += TAM_YAW;
        if (configuracion_imu & XACEL)//Si XACEL esta activado, los bytes estaran
corridos TAM_XACEL posiciones a la derecha
                pos += TAM_XACEL;
        if (configuracion_imu & YACEL)//Si YACEL esta activado, los bytes estaran
corridos TAM_YACEL posiciones a la derecha
                pos += TAM_YACEL;
        if (configuracion_imu & TEMP)//Si TEMP esta activado, los bytes estaran corridos
TAM_TEMP posiciones a la derecha
                pos += TAM_TEMP;
        return comaflotante2double(t[pos], t[pos+1], t[pos+2], t[pos+3]);
}

double getROLL(unsigned int t[tam_traza])
{       //La roll rate esta en 4 bytes de la traza en formato coma flotante
        int pos = 36;
        if (configuracion & VS)             //Si VS esta activado, los bytes estaran
corridos TAM_VS posiciones a la derecha
                pos += TAM_VS;
        if (configuracion & LONGACC)  //Si LONGACC esta activado, los bytes estaran
corridos TAM_LONGACC posiciones a la derecha
                pos += TAM_LONGACC;
        if (configuracion & LATACC)   //Si LATACC esta activado, los bytes estaran
corridos TAM_LATACC posiciones a la derecha
                pos += TAM_LATACC;
        if (configuracion & GLSAT)    //Si GLSAT esta activado, los bytes estaran
corridos TAM_GLSAT posiciones a la derecha
                pos += TAM_GLSAT;
```

```
        if (configuracion & GPSSAT)   //Si GPSSAT esta activado, los bytes estaran
corridos TAM_GPSSAT posiciones a la derecha
                pos += TAM_GPSSAT;
        if (configuracion & IKFS)     //Si IKFS esta activado, los bytes estaran corridos
TAM_IKFS posiciones a la derecha
                pos += TAM_IKFS;
        if (configuracion & ST)              //Si ST esta activado, los bytes estaran
corridos TAM_ST posiciones a la derecha
                pos += TAM_ST;
        if (configuracion & SQ)              //Si SQ esta activado, los bytes estaran
corridos TAM_SQ posiciones a la derecha
                pos += TAM_SQ;
        pos += 22;
        if (configuracion_imu & YAW)//Si YAW esta activado, los bytes estaran corridos
TAM_YAW posiciones a la derecha
                pos += TAM_YAW;
        if (configuracion_imu & XACEL)//Si XACEL esta activado, los bytes estaran
corridos TAM_XACEL posiciones a la derecha
                pos += TAM_XACEL;
        if (configuracion_imu & YACEL)//Si YACEL esta activado, los bytes estaran
corridos TAM_YACEL posiciones a la derecha
                pos += TAM_YACEL;
        if (configuracion_imu & TEMP)//Si TEMP esta activado, los bytes estaran corridos
TAM_TEMP posiciones a la derecha
                pos += TAM_TEMP;
        if (configuracion_imu & PITCH)//Si PITCH esta activado, los bytes estaran
corridos TAM_PITCH posiciones a la derecha
                pos += TAM_PITCH;
        return comaflotante2double(t[pos], t[pos+1], t[pos+2], t[pos+3]);
}

double getZACEL(unsigned int t[tam_traza])
{       //La aceleracion en Z esta en 4 bytes de la traza en formato coma flotante
        int pos = 36;
        if (configuracion & VS)              //Si VS esta activado, los bytes estaran
corridos TAM_VS posiciones a la derecha
                pos += TAM_VS;
        if (configuracion & LONGACC)  //Si LONGACC esta activado, los bytes estaran
corridos TAM_LONGACC posiciones a la derecha
                pos += TAM_LONGACC;
        if (configuracion & LATACC)   //Si LATACC esta activado, los bytes estaran
corridos TAM_LATACC posiciones a la derecha
                pos += TAM_LATACC;
        if (configuracion & GLSAT)    //Si GLSAT esta activado, los bytes estaran
corridos TAM_GLSAT posiciones a la derecha
                pos += TAM_GLSAT;
        if (configuracion & GPSSAT)   //Si GPSSAT esta activado, los bytes estaran
corridos TAM_GPSSAT posiciones a la derecha
                pos += TAM_GPSSAT;
        if (configuracion & IKFS)     //Si IKFS esta activado, los bytes estaran corridos
TAM_IKFS posiciones a la derecha
                pos += TAM_IKFS;
        if (configuracion & ST)              //Si ST esta activado, los bytes estaran
corridos TAM_ST posiciones a la derecha
                pos += TAM_ST;
        if (configuracion & SQ)              //Si SQ esta activado, los bytes estaran
corridos TAM_SQ posiciones a la derecha
                pos += TAM_SQ;
        pos += 22;
        if (configuracion_imu & YAW)//Si YAW esta activado, los bytes estaran corridos
TAM_YAW posiciones a la derecha
                pos += TAM_YAW;
        if (configuracion_imu & XACEL)//Si XACEL esta activado, los bytes estaran
corridos TAM_XACEL posiciones a la derecha
                pos += TAM_XACEL;
        if (configuracion_imu & YACEL)//Si YACEL esta activado, los bytes estaran
corridos TAM_YACEL posiciones a la derecha
                pos += TAM_YACEL;
        if (configuracion_imu & TEMP)//Si TEMP esta activado, los bytes estaran corridos
TAM_TEMP posiciones a la derecha
                pos += TAM_TEMP;
        if (configuracion_imu & PITCH)//Si PITCH esta activado, los bytes estaran
corridos TAM_PITCH posiciones a la derecha
                pos += TAM_PITCH;
        if (configuracion_imu & ROLL)//Si ROLL esta activado, los bytes estaran corridos
TAM_ROLL posiciones a la derecha
                pos += TAM_ROLL;
```

```
        return comaflotante2double(t[pos], t[pos+1], t[pos+2], t[pos+3]);
}

void printTrace(unsigned int t[tam_traza])
{
        int i;
        printf("Traza: ");
        for (i = 0; i < tam_traza; i++)
                printf("%02x ", t[i]);
        printf("\n");
}

void printTime (int tiempo)
{
        int horas, minutos, segundos, decimas;
        horas = ((tiempo / 100) / 60) / 60;
        minutos = ((tiempo / 100) / 60) - (horas * 60);
        segundos = (tiempo / 100) - (horas * 60 * 60) - (minutos * 60);
        decimas = tiempo % 100;
        printf("Tiempo: %02d:%02d:%02d'%02d\n", horas, minutos, segundos, decimas);
}

void printSats (int satelites)
{
        printf("Satelites: %d\n", satelites);
}

void printLat (double latitud)
{
        double grados, minutos;
        grados = (latitud/60)/100000;
        minutos = (grados-(int)grados)*60;
        printf("Latitud: %.0fº%f'\n", grados, minutos);
}

void printLon (double longitud)
{
        double grados, minutos;
        grados = (longitud/60)/100000;
        minutos = (grados-(int)grados)*60;
        printf("Longitud: %.0fº%f'\n", grados, minutos);
}

void printVel (double velocidad)
{
        double adaptada = velocidad/100;
        printf("Velocidad: %.02fkm/h\n", adaptada);
}

void printHead (double direccion)
{
        double adaptada = direccion/100;
        printf("Direccion: %.02fº\n", adaptada);
}

void printAlt (double altura)
{
        double adaptada = altura/100;
        printf("Altura: %.02fm\n", adaptada);
}

void printVVel (double vvertical)
{
        double adaptada = vvertical/100;
        printf("Velocidad vertical: %.02fkm/h\n", adaptada);
}

void printLongAcc(double plongitud)
{
        double adaptada = plongitud/10;
        printf("Aceleracion en la longitud: %.02fg\n", adaptada);
}

void printLatAcc (double platitud)
{
        double adaptada = platitud/10;
        printf("Aceleracion en la latitud: %.02fg\n", adaptada);
```

```
}

void printGlSat (int glsatelites)
{
	printf("Satelites Glonass: %d\n", glsatelites);
}

void printGpsSat (int gpssatelites)
{
	printf("Satelites GPS: %d\n", gpssatelites);
}

void printIKFS (int estadokalman)
{
	printf("Estado del filtro kalman: %d\n", estadokalman);
}

void printST (int tiposolucion)
{
	switch (tiposolucion)
	{
		case -1:
			printf("Tipo de solucion: No Data\n");
			break;
		case 0:
			printf("Tipo de solucion: No solution\n");
			break;
		case 1:
			printf("Tipo de solucion: Stand Alone\n");
			break;
		default:
			printf("Tipo de solucion: %d", tiposolucion);
	}
}

void printSQ (double vcalidad)
{
	printf("Calidad de la velocidad: %gkm/h\n", vcalidad);
}

void printYAW (double yawrate)
{
	printf("Yaw: %.02f°/s\n", yawrate);
}

void printXACEL (double acelx)
{
	printf("Aceleracion en X: %.08fg\n", acelx);
}

void printYACEL (double acely)
{
	printf("Aceleracion en Y: %.08fg\n", acely);
}

void printTEMP (double temperatura)
{
	printf("Temperatura: %.01f°C\n", temperatura);
}

void printPITCH (double pitchrate)
{
	printf("Pitch: %.02f°/s\n", pitchrate);
}

void printROLL (double rollrate)
{
	printf("Roll: %.02f°/s\n", rollrate);
}

void printZACEL (double acelz)
{
	printf("Aceleracion en Z: %.08fg\n", acelz);
}

int EncontrarAbrirPuerto(int conf, int conf_imu, double hz)
{
```

```
        char port[15] = "/dev/ttyUSB0";
        int fd, i = 0, j, k, bandera, intentos, seguro;
        configuracion = conf;
        configuracion_imu = conf_imu;
        fr = 1/hz;
        //Se calcula el tamaño de la traza dependiendo de la configuracion pasada por
parametro
        tam_traza = 47;
        if (fr >= 0.02)
                tam_traza += 13;
        if (fr >= 0.05)
        {
                if (VS & conf)
                        tam_traza += TAM_VS;
                if (LONGACC & conf)
                        tam_traza += TAM_LONGACC;
                if (LATACC & conf)
                        tam_traza += TAM_LATACC;
                if (GLSAT & conf)
                        tam_traza += TAM_GLSAT;
                if (GPSSAT & conf)
                        tam_traza += TAM_GPSSAT;
                if (SQ & conf)
                        tam_traza += TAM_SQ;
                if (ST & conf)
                        tam_traza += TAM_ST;
                if (IKFS & conf)
                        tam_traza += TAM_IKFS;
                if (YAW & conf_imu)
                        tam_traza += TAM_YAW;
                if (XACEL & conf_imu)
                        tam_traza += TAM_XACEL;
                if (YACEL & conf_imu)
                        tam_traza += TAM_YACEL;
                if (TEMP & conf_imu)
                        tam_traza += TAM_TEMP;
                if (PITCH & conf_imu)
                        tam_traza += TAM_PITCH;
                if (ROLL & conf_imu)
                        tam_traza += TAM_ROLL;
                if (ZACEL & conf_imu)
                        tam_traza += TAM_ZACEL;
        }
        char buf[tam_traza];
        while (1)
        {
                if (port[8] == 'U')   //Si port es ttyUSB*, incrementar el numero
                        port[11] = 48 + i;
                else                  //Si port es ttyS*, incrementar el numero
                        port[9] = 48 + i;

                fd = open(port, O_RDWR | O_NOCTTY | O_NONBLOCK );
                //Con O_NONBLOCK, read devuelve valor inmediatamente
                if (fd < 0)    //El puerto esta cerrado
                {
                        i++;
                        if (i == 10)
                        {
                                i = 0;
                                if (port[8] == 'U')
                                        strcpy(port, "/dev/ttyS0");
                                else
                                        strcpy(port, "/dev/ttyUSB0");
                        }
                }
                else           //El puerto esta abierto
                {
                        struct termios newtio;
                        bzero(&newtio, sizeof(newtio));
                        newtio.c_cflag = B115200 | CRTSCTS | CS8 | CLOCAL | CREAD;
                        //Aqui pueden ir mas opciones, estan recogidas al final del
archivo.
                        tcflush(fd, TCIFLUSH);
                        tcsetattr(fd,TCSANOW,&newtio);

                        bandera = tam_traza;
                        intentos = 4;
```

```
                        while (intentos != 0) //A veces no lo coge a la primera ya que le
llega alguna traza a mitad.
                        {
                                usleep(fr*1000000); //Se espera a la siguiente traza
                                if (read(fd, buf, bandera) == -1)
                                        break;
                                if (bandera == tam_traza)
                                        for (j = 0; j < tam_traza; j++)      //Se mira si
el principio de la traza se encuentra en el buffer
                                                if (buf[j] == 0x24 && buf[(j+1)%tam_traza]
== 0x56 && buf[(j+2)%tam_traza] == 0x42 &&
                                                    buf[(j+3)%tam_traza] == 0x4f &&
buf[(j+4)%tam_traza] == 0x58 && buf[(j+5)%tam_traza] == 0x33 &&
                                                    buf[(j+6)%tam_traza] == 0x69 &&
buf[(j+7)%tam_traza] == 0x2c)
                                                {
                                                        bandera = j;   //Si la traza no
empieza en j = 0, bandera sirve para sincronizar.
                                                        break;
                                                }
                                else
                                {
                                        bandera = tam_traza;
                                        if (read(fd, buf, bandera) == -1)
                                        {
                                                printf("Error: %s\n", strerror(errno));
                                                break;
                                        }
                                }
                                if (bandera == 0)             //A veces se desincroniza
nada mas empezar, esto hace algunas mediciones de prueba.
                                {
                                        for (seguro = 10; seguro > 0; seguro--)
                                        {
                                                usleep(fr*1000000);
                                                if (read(fd, buf, tam_traza) == -1)
                                                {
                                                        printf("Error: %s\n",
strerror(errno));
                                                        bandera = tam_traza;
                                                        break;
                                                }
                                                if (buf[0] != 0x24 || buf[1] != 0x56 ||
buf[2] != 0x42 || buf[3] != 0x4f ||
                                                    buf[4] != 0x58 || buf[5] != 0x33 ||
buf[6] != 0x69 || buf[7] != 0x2c)
                                                {
                                                        bandera = tam_traza;
                                                        break;  //Se sale si en alguna de las
pruebas el inicio de la traza no coincide con el inicio del buffer
                                                }
                                        }
                                        if (seguro == 0)
                                        {
                                                usleep(fr*1000000);  //Esperamos para que en
read haya algo que leer en la siguiente funcion.
                                                return fd;     //Sale de la funcion cuando
ya esta sincronizado completamente.
                                        }
                                }
                                intentos--; //Cuando se acaban los intentos, se cambia de
puerto
                        }
                        close(fd);
                        i++;
                        if (i == 10) //Recorremos 10 posibles puertos serie y 10 posibles
puertos USB
                        {
                                i = 0;
                                if (port[8] == 'U')
                                        strcpy(port, "/dev/ttyS0");
                                else
                                        strcpy(port, "/dev/ttyUSB0");
                        }
                }
        }
}
```

```
void EsperarSiguienteTraza(int fd, double MUESTREO)
{
	//Solo hace read cuando detecta que hay dato nuevo. Vacia primero con un read
multiple.
	fd_set rfds;
	int retval, n = 0;
	float i = tam_traza*((MUESTREO/fr)-1);
	char buf[tam_traza];
	FD_ZERO(&rfds);
	while(1) //Va al final del bufer para esperar la nueva traza
	{
		n = read(fd, buf, tam_traza);
		if (n == 0)
			break;
	}
	n = 0;
	while (n < i)
	{
		FD_SET(fd, &rfds);
		retval = select(fd+1, &rfds, NULL, NULL, NULL); //Espera a que llegue un
dato al descriptor de fichero
		if (retval)
			n += read(fd, buf, tam_traza);
	}
}

datos CogerTraza(int fd)
{
	int i, j, n, n2;
	char buf[tam_traza];
	datos d;
	n = 0, n2 = 0;
	EsperarSiguienteTraza(fd, fr); //Va al final del bufer para esperar a la
siguiente traza
	while (n2 != tam_traza)
	{
		n = read(fd, buf, tam_traza-n2);
		if (n != 0)
		{
			for (i = n2; i < n2 + n; i++)
				d.traza[i] = 0x000000ff & buf[i-n2];
			if (n2 == 0 && (0x000000ff & buf[0]) != 0x24) //Quita basura hasta
la siguiente traza
				n = 0;
			if ((n2 + n) > 8 && (d.traza[1] != 0x56 || d.traza[2] != 0x42 ||
d.traza[3] != 0x4f || d.traza[4] != 0x58 ||
					d.traza[5] != 0x33 || d.traza[6] != 0x69 ||
d.traza[7] != 0x2c))
			{
				n = 0;
				n2 = 0;//Hacer que no pille el de antes
			}
		}
		//Hacemos 0x000000ff & buf[] porque cuando el primer bit de buf[] es 1 lo
toma como negativo.
		n2 = n2 + n;
	}
	//TIEMPO
	d.tiempo = getTime(d.traza);
	//SATELITES
	d.satelites = getSats(d.traza);
	//VELOCIDAD
	d.velocidad = getVel(d.traza);
	if (fr >= 0.02)
	{
		//LATITUD
		d.latitud = getLat(d.traza);
		//LONGITUD
		d.longitud = getLon(d.traza);
		//DIRECCION
		d.direccion = getHead(d.traza);
		//ALTURA
		d.altura = getAlt(d.traza);
	}
	if (fr >= 0.05)
	{
```

```
                //VELOCIDAD VERTICAL
                if (configuracion & VS)
                        d.vvertical = getVVel(d.traza);
                //PRECISION LONGITUD
                if (configuracion & LONGACC)
                        d.plongitud = getLongAcc(d.traza);
                //PRECISION LATITUD
                if (configuracion & LATACC)
                        d.platitud = getLatAcc(d.traza);
                //GLONASS SATELITES
                if (configuracion & GLSAT)
                        d.glsatelites = getGlSat(d.traza);
                //GPS SATELITES
                if (configuracion & GPSSAT)
                        d.gpssatelites = getGpsSat(d.traza);
                //ESTADO FILTRO IMU KALMAN
                if (configuracion & IKFS)
                        d.estadokalman = getIKFS(d.traza);
                //TIPO DE SOLUCION
                if (configuracion & ST)
                        d.tiposolucion = getST(d.traza);
                //CALIDAD DE LA VELOCIDAD
                if (configuracion & SQ)
                        d.vcalidad = getSQ(d.traza);
                //YAW RATE
                if (configuracion_imu & YAW)
                        d.yawrate = getYAW(d.traza);
                //ACELERACION EN X
                if (configuracion_imu & XACEL)
                        d.acelx = getXACEL(d.traza);
                //ACELERACION EN Y
                if (configuracion_imu & YACEL)
                        d.acely = getYACEL(d.traza);
                //TEMPERATURA
                if (configuracion_imu & TEMP)
                        d.temperatura = getTEMP(d.traza);
                //PITCH RATE
                if (configuracion_imu & PITCH)
                        d.pitchrate = getPITCH(d.traza);
                //ROLL RATE
                if (configuracion_imu & ROLL)
                        d.rollrate = getROLL(d.traza);
                //ACELERACION EN Z
                if (configuracion_imu & ZACEL)
                        d.acelz = getZACEL(d.traza);
        }
        return d;
}

void ImprimirDatos(datos d)
{
        //TRAZA
        printTrace(d.traza);
        //TIEMPO
        printTime(d.tiempo);
        //SATELITES
        printSats(d.satelites);
        //VELOCIDAD
        printVel(d.velocidad);
        if (fr >= 0.02)
        {
                //LATITUD
                printLat(d.latitud);
                //LONGITUD
                printLon(d.longitud);
                //DIRECCION
                printHead(d.direccion);
                //ALTURA
                printAlt(d.altura);
        }
        if (fr >= 0.05)
        {
                //VELOCIDAD VERTICAL
                if (configuracion & VS)
                        printVVel(d.vvertical);
                //PRECISION LONGITUD
                if (configuracion & LONGACC)
```

```
                        printLongAcc(d.plongitud);
                //PRECISION LATITUD
                if (configuracion & LATACC)
                        printLatAcc(d.platitud);
                //GLONASS SATELITES
                if (configuracion & GLSAT)
                        printGlSat(d.glsatelites);
                //GPS SATELITES
                if (configuracion & GPSSAT)
                        printGpsSat(d.gpssatelites);
                //ESTADO FILTRO IMU KALMAN
                if (configuracion & IKFS)
                        printIKFS(d.estadokalman);
                //TIPO DE SOLUCION
                if (configuracion & ST)
                        printST(d.tiposolucion);
                //CALIDAD DE LA VELOCIDAD
                if (configuracion & SQ)
                        printSQ(d.vcalidad);
                //YAW RATE
                if (configuracion_imu & YAW)
                        printYAW(d.yawrate);
                //PITCH RATE
                if (configuracion_imu & PITCH)
                        printPITCH(d.pitchrate);
                //ROLL RATE
                if (configuracion_imu & ROLL)
                        printROLL(d.rollrate);
                //ACELERACION EN X
                if (configuracion_imu & XACEL)
                        printXACEL(d.acelx);
                //ACELERACION EN Y
                if (configuracion_imu & YACEL)
                        printYACEL(d.acely);
                //ACELERACION EN Z
                if (configuracion_imu & ZACEL)
                        printZACEL(d.acelz);
                //TEMPERATURA
                if (configuracion_imu & TEMP)
                        printTEMP(d.temperatura);
        }
}
```