Proyecto Final de Carrera

Ingeniería técnica de telecomunicaciones

Especialidad en sistemas electrónicos

"CARACTERIZACION DE UNA UNIDAD DE MEDIDA INERCIAL PARA LA OBTENCION DE ENTORNOS DE SIMULACION DE UN SISTEMA DE NAVEGACION INERCIAL DE BAJO COSTE MEDIANTE ALLAN VARIANCE"

Carlos Preckler Clemente

Director: **D. Carles Ferrer Ramis**

Departamento de microelectrónica y sistemas electrónicos.

Asesor: **Alex García Quinchia**

Departamento de microelectrónica y sistemas electrónicos.

Escola Tècnica Superior d'Enginyeria (ETSE)
Universitat Autònoma de Barcelona (UAB)

Febrero 2012

El tribunal de evaluación de este Trabajo Final de Carrera, reunido el día 13 de Febrero de 2012, ha acordado conceder la siguiente cualificación:

Presidente:

Vocal:

Secretario:

El sotasignant, **D. Carles Ferrer Ramis**, Professor de l'Escola Tècnica Superior d'Enginyeria (ETSE) de la Universitat Autònoma de Barcelona (UAB),

CERTIFICA:

Que el projecte presentat en aquesta memòria de Projecte Fi de Carrera ha estat realitzat sota la seva direcció per l'alumne **Carlos Preckler Clemente**.

I, perquè consti a tots els efectes, signa el present certificat.

Bellaterra, 12 / Gener / 2012

Signatura: **D. Carles Ferrer Ramis**

INDICE

1. INTRODUCCION

Desde que el hombre se lanzó a la exploración y conquista de nuevas tierras siempre ha desarrollado útiles y métodos que le permitieran orientarse y desplazarse de la forma más segura y precisa posible.

En la actualidad es habitual disponer de sistemas o dispositivos de ayuda a la navegación en vehículos terrestres (*coches, camiones y motocicletas de gama alta*), aéreos (*aviones y helicópteros*), marítimos y espaciales (exploración extraterrestre).

Estos sistemas han proliferado gracias a la introducción militar de vehículos no tripulados destinados a tareas de vigilancia y exploración, evitando poner en peligro vidas humanas de forma innecesaria.

También la industria de manufactura y transporte ha potenciado el uso de estos sistemas. Con su introducción en los vehículos de transporte y en los almacenes de material, ha automatizado los procesos rutinarios de almacenamiento y recogida de la mercancía. Reduciendo así la probabilidad de errores y accidentes durante los procesos de carga y descarga y la posibilidad de seguimiento durante el transporte.

Podemos encontrar en el mercado una amplia gama de estos sistemas, con un amplio abanico de precios, prestaciones y tamaños. En ocasiones específicos (integrados en el vehículo al que están destinados) y en ocasiones genéricos (dispositivos móviles GSM y Smartphones[1]).

A pesar de la amplia variedad de estos sistemas, la composición de los mismos es básicamente la misma. Incorporan unos sensores determinados (receptor GPS[2], brújula y unidad de medida inercial) y un software más o menos específico conformado por la interfaz con el usuario y los algoritmos pertinentes para el correcto funcionamiento del sistema de navegación asistida.

[1] Teléfonos inteligentes
[2] GPS: Global Positioning System (Sistema de Posicionamiento Global).

a) Resumen.

En este documento vamos a desarrollar una plataforma hardware e implementar un modelo de los errores estocásticos de una unidad de medida inercial, o IMU[3], mediante Varianza de Allan.

En el Capítulo I se presentan los objetivos de este documento.

En el Capítulo II introducimos el uso de las unidades de medida inercial en los sistemas de navegación asistidos y en otros sistemas de uso cotidiano. Se ubica este estudio como parte de un sistema de navegación asistida de mayor envergadura. Introducimos el algoritmo de Varianza de Allan. Por último, justificamos el uso de la Unidad de Medida Inercial SPARKFUN SEN-09184 ATOMIC IMU 6 DOF.

En el Capítulo III realizamos una descripción detallada de la IMU, funcionamiento y características. Se presentan las características de los sensores de la unidad y del resto de dispositivos utilizados durante el estudio realizado.

En el Capítulo IV se muestra el sistema hardware utilizado. Se describe el proceso de obtención y procesado de los datos adquiridos. Por último, se presentan los modelos obtenidos tras el estudio.

En el Capítulo V se presentan los resultados del estudio realizado y las conclusiones extraídas.

En el Capítulo VI se indican mejoras y ampliaciones sobre el estudio realizado.

[3] IMU: Inertial Measurement Unit (Unidad de Medida Inercial).

b) Objetivos.

En este trabajo, basado en los documentos:

- "Analysis and Modeling of MEMS based Inertial Sensors" (Ref. [1])
- "Stochastic Modeling of MEMS Inertial sensors" (Ref. [2])

Se pretende:

- Desarrollar e implementar una plataforma hardware para una unidad de medida inercial de bajo coste (SPARKFUN SEN-09184 ATOMIC IMU 6 DOF) que permita la adaptación de dispositivos tales como GPS y sensores de temperatura. Además de dotar de comunicación inalámbrica bluetooth a la plataforma, para la adquisición de datos y configuración de la IMU.

- Estudiar e implementar el método de Allan Variance para la identificación de los errores que afectan a los sensores inerciales de la Atomic IMU.

- Implementar un modelo del error estocástico a partir de los parámetros obtenidos del análisis de Allan Variance.

Se ha escogido para esta aplicación una IMU de muy bajo coste del fabricante SPARKFUN, el modelo seleccionado es: SEN-09184 ATOMIC IMU 6 DOF.

Para llevar a cabo la tarea propuesta analizaremos los sensores que incorpora esta unidad de bajo coste. Identificaremos los errores que afectan a cada uno de los sensores e implementaremos el modelado de errores aleatorios mediante el método de modelado de Allan Variance para los acelerómetros y giroscopios que incorpora la IMU escogida.

Finalizado el análisis de la varianza, obtendremos unos parámetros que nos permitirán generar cada modelo para cada uno de los errores que afectan a de cada uno de los

sensores que incorpora esta IMU. Así, elaboraremos un modelo de error para cada sensor.

Con los modelos de error obtenidos se podrá, a posteriori, corregir dichos errores mediante los algoritmos correspondientes en el sistema de ayuda a la navegación.

De este modo incrementamos la eficiencia inicial de la IMU y por extensión la del sistema de navegación en el que se integre.

2. ESTADO DEL ARTE

Originalmente los sistemas de asistencia a la navegación solo contaban con un módulo de posicionamiento global (GPS). El GPS es un instrumento fundamental estos sistemas de navegación, sin embargo se encuentra afectado por diversos factores:

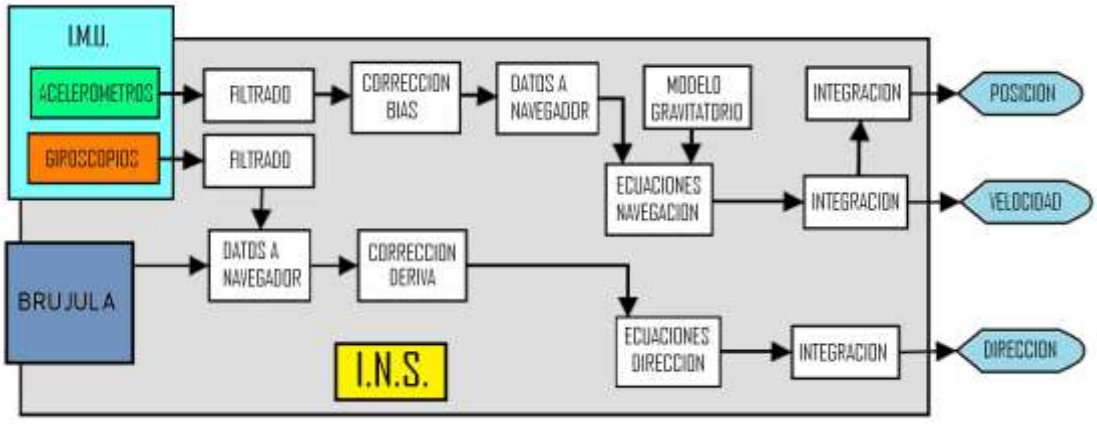
- Factores ambientales: *cielo nublado o lluvia*.
- Factores geográficos: *áreas urbanas y túneles*.
- Factores funcionales: *número satélites disponibles para posicionamiento*.

Por ello, hoy en día se están integrando otros dispositivos sensores en estos sistemas de navegación asistida, que complementan las debilidades de los receptores GPS.

Uno de estos instrumentos son las Unidades de Medida Inercial, las cuales no se encuentran afectadas por el entorno como es el caso del GPS.

Los dispositivos de medida inercial suelen utilizarse en todo tipo de sistemas electromecánicos complejos para controlar sus desplazamientos.

En los sistemas de navegación asistida, se incorporan para complementar a otros dispositivos de medida como pueden ser módulos GPS y brújulas digitales. De esta manera se incrementan la eficiencia de estos sistemas. Suelen aparecer emparejados con brújulas, conformando Sistemas de Navegación Inercial (INS[4])



[4] INS: Inertial Navigation System (Sistema Navegación Inercial).

En general, los INS actuales incorporan IMU. El corazón del sistema se encarga de gestionar los datos recibidos por dichos sensores y de aplicar los algoritmos necesarios para corregir los errores inherentes de cada tipo de sensor y proporcionar una navegación precisa.

Las IMU integran acelerómetros y giroscopios, sensores que miden aceleración lineal y velocidad angular respectivamente. De los datos entregados por estos sensores podemos integrar velocidad y posición (acelerómetros) y ángulo o trayectoria (giroscopios). Así obtenemos datos relativos de desplazamiento del sistema en el que se integra este dispositivo. Estos datos no se encuentran tan afectados por los factores ambientales como en el caso de los receptores GPS.

Actualmente las IMU se incorporan en todo tipo de sistemas electrónicos de carácter portátil:

- Terminales GSM y Smartphones.
- Tablet PC's
- Dispositivos de ocio domésticos (mandos de Nintendo Wii)
- Cámaras de video y foto (estabilizadores la imagen)
- Discos duros de ordenadores portátiles.

No obstante, las IMU presentan errores a corto, medio y largo plazo temporal, tanto mayores cuanto mayor es el tiempo de funcionamiento debido a su carácter acumulativo, que alteran las medidas detectadas y por tanto afectan de forma negativa al rendimiento de los sistemas de navegación INS.

Por este motivo, se hace necesario establecer unos modelos de error que permitan corregir estos comportamientos no deseados que afectan negativamente al conjunto de la aplicación.

a) Ubicación del proyecto.

En el ámbito de la navegación de vehículos tripulados y no tripulados, es común el uso de sistemas que incorporan varios tipos de sensores que permiten realizar tareas de guiado y posicionamiento. Los sensores utilizados en estos incorporan receptores GPS, módulos IMU y brújulas digitales.

Ejemplos de estos sistemas los tenemos a continuación.

MARS POLAR LANDER[5] (vehículo de exploración extraterrestre de la NASA)

RAVEN RQ-11[6] (vuelos no tripulados de reconocimiento para entornos bélicos)

[5] http://mars.jpl.nasa.gov/msp98/lander/index.html
[6] http://www.avinc.com/uas/small_uas/raven/

Nuestra IMU se integra en un sistema de posicionamiento y guiado, complementando a otros dos módulos en un sistema de navegación autónomo.

Dicho sistema, está controlado por una plataforma de desarrollo basada en FPGA[7] de Xilinx (***XUPV5-LX110T***) que se encarga de recoger los datos recibidos por los sensores que incorpora, IMU, receptor GPS y Magnetómetro (brújula digital).

Con los datos obtenidos, la FPGA realiza los cálculos correspondientes a trayectorias y velocidades. Con ello, realiza las correcciones necesarias para llevar a cabo un guiado preciso.

Módulo GPS LS40MM

Brújula Digital CMPS03

XILINX XUPV5-LX110T

[7] FPGA: Field Programmable Gate Array.

b) Identificación de errores aleatorios (Varianza de Allan).[8]

Todos los dispositivos electrónicos por el simple hecho de ser dispositivos reales, no ideales, muestran en su funcionamiento unas desviaciones respecto a las especificaciones diseñadas. Dichas desviaciones son debidas a factores tales como la propia fabricación del dispositivo, impurezas de los materiales utilizados en la fabricación, la calibración, las condiciones de funcionamiento o incluso la alimentación que el propio dispositivo requiera para funcionar.

Estas desviaciones, conocidas como ruido, tienen orígenes diversos. Se catalogan en dos familias:

Errores determinísticos: *son conocidos a priori y es sencillo corregir su efecto*.

Errores estocásticos: *son aleatorios y ello implica que se deben estudiar y modelar para poder corregir su efecto.*

Puesto que los sensores utilizados en el módulo introducen un error aleatorio que falsea las medidas debemos modelar dicho error para poder corregir sus desviaciones.

Por tal motivo haremos uso de un método de modelado estocástico de errores denominado Varianza de Allan. Este método es computacionalmente sencillo de implementar y se obtienen resultados muy precisos, 95%-100%, respecto a los ruidos modelados.

Como en el laboratorio podemos controlar las entradas del sistema y conocemos las salidas teóricas del mismo, según las hojas características del fabricante y las configuraciones aplicadas, toda respuesta que difiera de la teórica es considerada como error introducido por dicho sistema.

[8] Conceptos teóricos y expresiones matemáticas referencia [1]. Véase Anexo "***a***"

Así pues, para modelar dicho error lanzamos la premisa:

Puesto que nuestro módulo se encuentra en reposo y en condiciones controladas de inclinación asumimos como fuente de excitación del sistema un ruido blanco.

Como el ruido blanco es un proceso estocástico estacionario podemos aplicar el método escogido.

El modelado estocástico permite estimar las distribuciones de probabilidad de las posibles salidas del sistema. Dichas distribuciones de probabilidad se obtienen tomando un gran número de muestras, que reflejan la variación aleatoria de las entradas.

Si modelamos las salidas de los sensores como procesos estocásticos podremos caracterizar el ruido de las medidas mediante la densidad espectral de potencia o cualquier otra técnica de estimación de ruido.

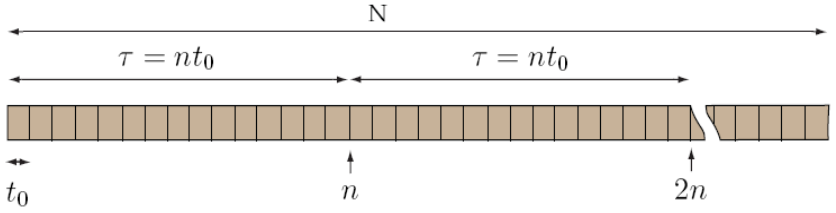
Para determinar dichos modelos emplearemos el método de modelización estocástica de Allan Variance (stochastic modeling).

Este método es matemáticamente sencillo de desarrollar e implementar, aunque computacionalmente es tedioso pues requiere de un elevado número de iteraciones en el proceso de cálculo.

El algoritmo de cálculo consiste en el siguiente proceso:

- *Sea un grupo de muestras consecutivas de longitud* $\boldsymbol{N}$ *("N" muestras) tomadas a una frecuencia de muestreo* $\boldsymbol{F_s}$ *y con un tiempo de muestra* $\boldsymbol{\tau_0}$.

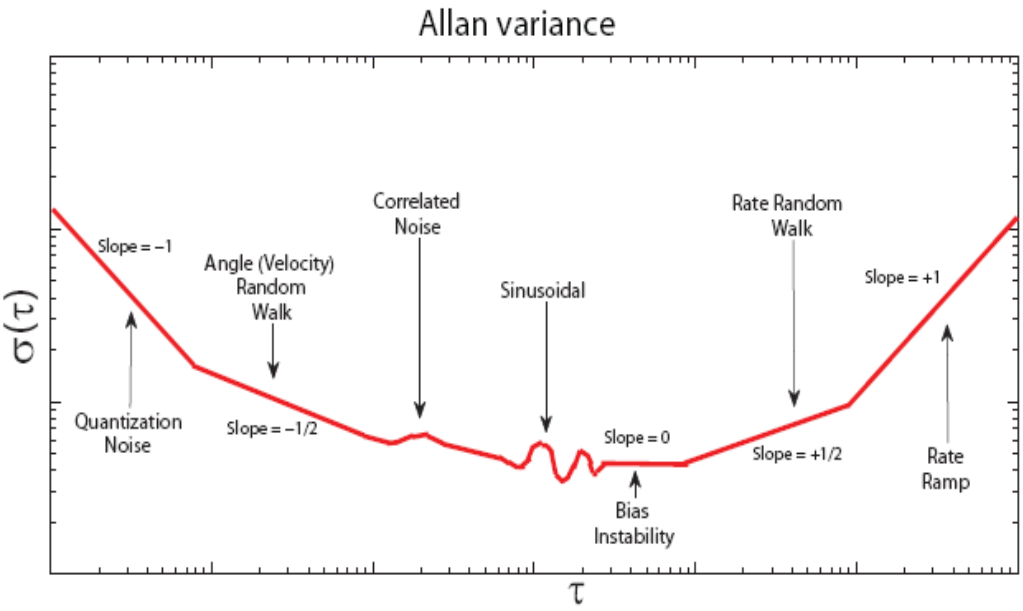
- *Se forman grupos de "n" datos consecutivos, con duraciones* $\boldsymbol{\tau_0, 2\tau_0, 3\tau_0, \ldots n\tau_0}$ *con* $1 \leq n < N/2$



- *Cada grupo, llamado CLUSTER, tiene asociado un tiempo* ***"τ"*** *que tiene una duración* $n\tau_0$ *dependiendo del número de muestras del CLUSTER.* ***(n=1,2,… <(N/2)-1)***

- *A continuación se obtiene la media de cada CLUSTER y se comparan todas las medias.*

En nuestra aplicación trabajaremos con clústeres de potencias de 2, esto permite reducir ampliamente el número de iteraciones del algoritmo, y como consecuencia de ello el tiempo de cálculo. Los resultados obtenidos utilizando potencias de dos en lugar de incrementos uno a uno en el tamaño de los clústeres difieren solo a frecuencias altas, donde se observan unas variaciones abruptas en las pendientes de las gráficas de Varianza Allan, relacionadas con la perdida de resolución debido a la diferencia de tamaños entre clústeres de una iteración a la siguiente.



Gráfica 1: resultado genérico análisis Varianza Allan[9]

Tras el análisis de Varianza Allan, procederemos a identificar las constantes de error de cada una de las contribuciones. Esto lo hacemos para cada uno de los 6 sensores de la IMU.

Con las constantes determinadas, podemos proceder a modelar cada uno de los errores que afectan a cada uno de los sensores. Para ello aplicamos los modelos establecidos para cada tipo de error.

[9] Gráfica 1: de referencia [1]

- Quantization Noise (Q): corresponde con un tramo de pendiente -1.

  Su varianza asociada es:

$$\sigma^2(\tau) = 3\frac{Q^2}{\tau^2} \Leftrightarrow \sigma(\tau) = \sqrt{3}\frac{Q}{\tau}$$

- Angular/Velocity Randon Walk (N): corresponde con un tramo de pendiente -1/2.

  Su varianza asociada es:

$$\sigma^2(\tau) = \frac{N^2}{\tau} \Leftrightarrow \sigma(\tau) = \frac{N}{\sqrt{\tau}}$$

- Bias Instability (B): corresponde con el mínimo de la curva de la Varianza Allan.

  Asíntota de pendiente 0.

  Su varianza asociada es:

$$\sigma^2(\tau) = 2\frac{B^2}{\pi}\ln 2 \Leftrightarrow \sigma(\tau) = \sqrt{\frac{2\ln(2)}{\pi}}B \approx 0.664B$$

- Rate Random Walk (K): corresponde con un tramo de pendiente +1/2.

  Su varianza asociada es:

$$\sigma^2(\tau) = \frac{\tau K^2}{3} \Leftrightarrow \sigma(\tau) = \sqrt{\frac{\tau}{3}}K$$

- Rate Ramp (R): corresponde con un tramo de pendiente +1.

  Su varianza asociada es:

$$\sigma^2(\tau) = \frac{\tau^2 R^2}{2} \Leftrightarrow \sigma(\tau) = \frac{\tau}{\sqrt{2}}R$$

Con los modelos de cada uno de los errores podemos determinar el modelo completo del error de cada sensor. Este es la suma de cada uno de los modelos de error previamente obtenidos:

$$\sigma_{TOT}^2 = \sigma_{Quant}^2 + \sigma_{AVRW}^2 + \sigma_{Bias}^2 + \sigma_{RRW}^2 + \sigma_{RateRamp}^2$$

Una vez obtenidos los modelos completos, comparamos cada uno con la señal obtenida del sensor correspondiente. De esta manera podemos determinar cuál es la precisión, coincidencia (FIT), del modelo generado. Usaremos la definición de FIT[10]:

$$FIT = \left(1 - \frac{\sqrt{\sum_{k=1}^{L} (X_{medida}(k) - X_{Modelo}(k))^2}}{\sqrt{\sum_{k=1}^{L} (X_{medida}(k))^2}}\right) * 100$$

c) <u>Justificación de uso de esta IMU.</u>

Se ha escogido esta IMU, frente al resto de IMUs del segmento de ultra bajo coste, por aunar las características de bajo consumo de corriente (24mA), el amplio rango de tensión de alimentación (3,4V_{DC} – 10 V_{DC}), la posibilidad de dotarla directamente de conectividad inalámbrica mediante módulos XBee (consumo: 75mA), la posibilidad de modificación del software mediante la programación serie del ATMega168, el hardware (sustitución de giroscopios) y el coste del módulo (120$).

[10] Bibliografía Ref.[2]

3. DESCRIPCION GENERAL DEL HARDWARE.

a) Descripción general de la unidad de medida inercial (IMU).

Un módulo de navegación inercial, IMU, es un dispositivo que incorpora al menos un acelerómetro y un giroscopio orientados axialmente. De esta forma proporciona los datos de aceleración lineal y velocidad angular sobre el eje en el que se encuentran ubicados.

En nuestro caso la IMU incorpora un acelerómetro triaxial y tres giroscopios colocados sobre tres ejes ortogonales, coincidiendo cada giroscopio con una de las componentes axiales del acelerómetro triaxial. Esto es, en cada eje ortogonal disponemos de una pareja acelerómetro/giroscopio.

Habitualmente estos módulos incorporan un microcontrolador, encargado de recoger, convertir y transmitir los datos entregados por los sensores, en forma de niveles de voltaje, a otro sistema que se encargará de trabajar los datos para su uso.

Los acelerómetros detectan variaciones de velocidad perpendicularmente a la superficie del propio sensor, proporcionando una tensión proporcional a la aceleración que sufre el sensor.

Los giroscopios detectan variaciones de ángulo en un eje determinado, entregando una tensión proporcional al ángulo de giro, o rotación, que sufre el sensor.

b) Características de la IMU.

Hemos escogido la IMU de Sparkfun Electronics[11] SEN-09184 Atomic IMU 6 DOF.

Se trata de un módulo multicomponente basado en microcontrolador, al que se complementa con un acelerómetro triaxial y tres giroscopios axiales.

El cerebro del módulo es un microcontrolador ATMEL ATmega168. Este trabaja a una frecuencia de reloj de 10MHz y dedica 6 conversores ADC[12] de 10bits de resolución para las medidas de los sensores integrados.

Los sensores que incorpora el módulo son:

- *Acelerómetro triaxial FREESCALE MMA7260Q.*
- *Tres giroscopios axiales ST MICROELECTRONICS LISY300AL.*

CARACTERISTICAS:

- Alimentación: *[3.4-10]V_{DC}*
- Consumo: *24mA (75mA con XBee[13])*
- Resolución giroscopios: *0.977º/s*
- Resolución acelerómetros: *0.00403g @ 1.5g (FS)*
  0.00537g @ 2g (FS)
  0.0107g @ 4g (FS)
  0.0161g @ 6g (FS)
- UART[14]

[11] www.sparkfun.com
[12] ADC: Conversor analógico a digital (Digital to Analog Conversor).
[13] XBee: protocolo comunicaciones RF IEEE 802.15.4.
[14] UART: Receptor-Transmisor asíncrono universal.

PROS:

- *Versatilidad proporcionada por el ATmega168, permite editar y personalizar el código fuente para ajustar su uso a las necesidades de la aplicación.*
- *Tensión de entrada regulada a 3.3V.*
- *Bajo precio del dispositivo, teniendo en cuenta las prestaciones que ofrece.*
- *Opciones de configuración de los rangos de medida de gravedad y ángulo.*
- *Posibilidad de conexión de un módulo XBee, que proporciona conectividad inalámbrica.*
- *Posibilidad de uso con baterías, debido a su bajo consumo.*

CONTRAS:

- *Al ser un dispositivo de ultra bajo coste, los sensores son ruidosos y requieren de un proceso elaborado de corrección de errores.*
- *Tamaño relativamente grande del módulo.*
- *No viene en ningún tipo de carcasa, dejando expuestos todos los componentes.*

c) Características de los acelerómetros.

La IMU incorpora el acelerómetro triaxial MMA7260Q de Freescale[15].

Acelerómetros micro mecánicos de efecto capacitivo. Disponen de acondicionamiento de la señal entregada: *filtro paso bajo orden 1, compensación efecto térmico y selección de sensibilidad en 'g'. (1.5g, 2g, 4g o 6g).*

[15] www.freescale.com

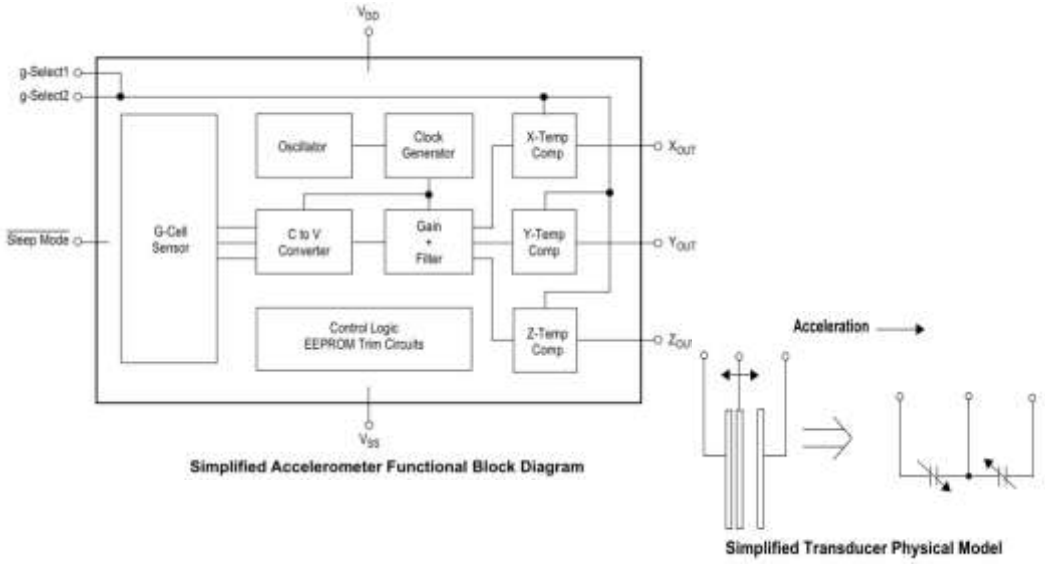
Características:

- Bajo coste.
- Diseño robusto, altamente resistente a golpes.
- Tiempo de encendido corto.
- Triaxial
- Alimentación: *2,2V-3,6V*
- Consumo: *500uA (Typ)*
- Sensibilidad seleccionable: *800mV/g @ 1,5g*
  600mV/g @ 2g
  300mV/g @ 4g
  200mV/g @ 6g
- Deriva térmica Sensibilidad: *±0,03%/ºC (Typ)*
- Termo compensación de las medidas.
- Rango térmico: *[-20,+85]ºC*

Usos típicos:

- Sistemas de navegación.
- Detección de movimiento: plataformas robóticas, podómetros, sistemas de control de juegos (mandos Nintendo Wii)
- Detección de caída en ordenadores portátiles y reproductores multimedia portables con disco duro.
- Sistemas de estabilización de imagen de cámaras de foto y video.

Las figuras a continuación muestran, de forma simple, la estructura y el funcionamiento del MMA7260Q:

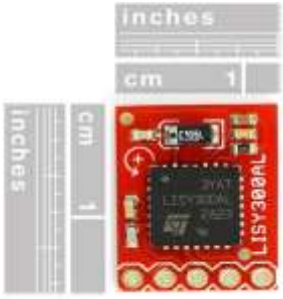


Simplified Accelerometer Functional Block Diagram

Simplified Transducer Physical Model

d) Características de los giroscopios.

La IMU incorpora tres giroscopios axiales LISY300AL de ST Microelectronics[16].



Sensor micro electromecánico dedicado a la medida de ratios de giro en un eje, de bajo consumo energético.

Características:

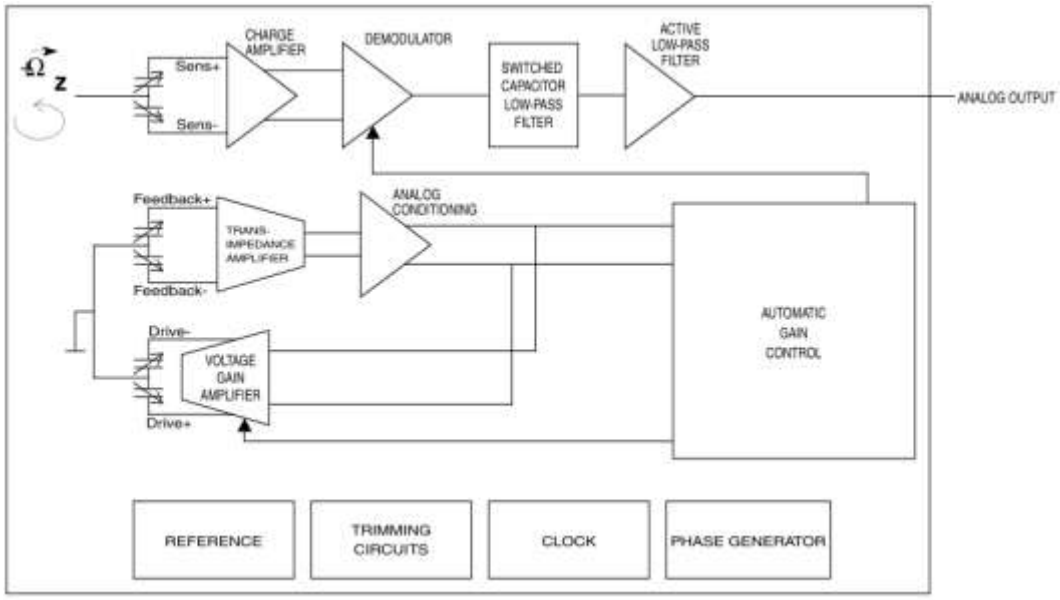
- Bajo consumo energético.
- Sistema integrado de filtrado paso bajo.
- Sistema integrado de auto-test.
- Auto apagado integrado.
- Niveles de salida analógicos.
- Alta resistencia a impactos.
- Alimentación: *2,7V - 3,6V*.
- Consumo: *4.8mA (Typ)*
- Escala medida: ±*300º/s*
- Sensibilidad: *3.3mV/(º/s)*
- Deriva térmica Sensibilidad: *4% (Typ)*
- No linealidad: ±*0.8% (FS)*
- Termo compensación de las medidas.
- Rango térmico: *[-40,+85]ºC*

Usos típicos:

- Sistemas de navegación GPS.
- Sistemas de estabilización de imagen en cámaras digitales de foto y video.
- Robótica.
- Sistemas de control de movimiento con interacción hombre-máquina.
- Dispositivos de entrada de sistemas de realidad virtual y sistemas de juego.

[16] www.st.com

La figura a continuación muestra, de forma simple, la estructura y el funcionamiento de los giroscopios LISY300AL:



e) Otros dispositivos.

EIKON Bluemore200 (http://www.eikonsite.it)

Módulo bluetooth de comunicaciones serie inalámbricas.

Usos típicos:

- Sensores de redes.
- Sistemas de automatización doméstica e industrial.
- Telemetrías.
- Comunicaciones inalámbricas punto a punto bidireccionales.

Características:

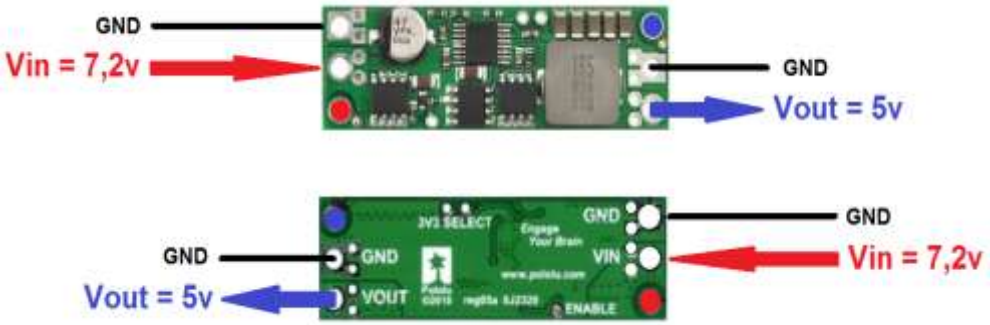
- Bluetooth v1.2 compatible, hasta 100m alcance.
- Comunicación serie RS232 (1200baud – 230400baud).
- UART serie 3,3V y 5V.
- Hasta 3 señales I/O.
- Firmware estándar o personalizado.
- No requiere drivers para su funcionamiento.

BATERIA

Acumulador de Ni-Cd a 7,2V y capacidad de 1500mAh del fabricante BYCMO[17].

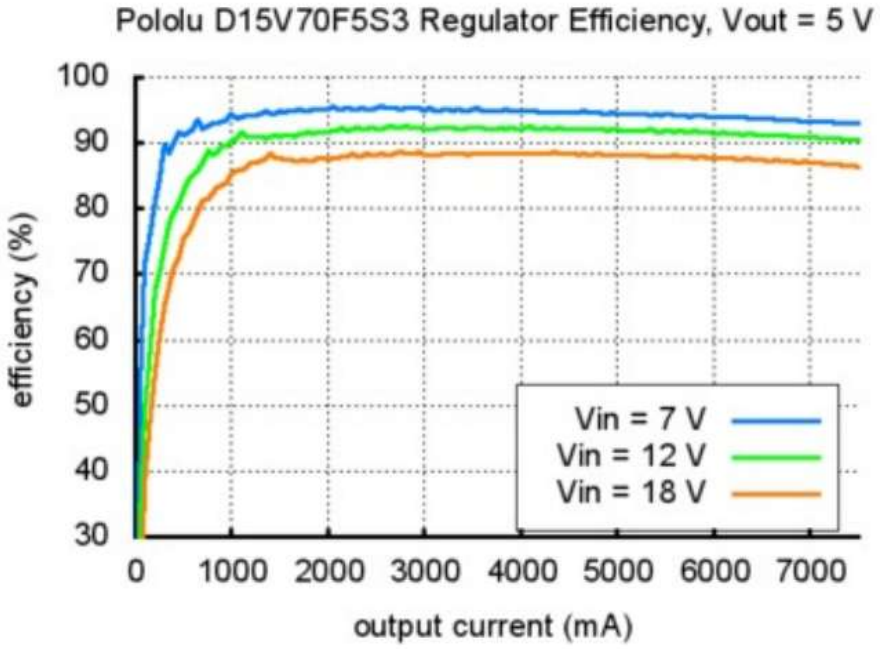
[17] www.bycmo.com

POLOLU[18] STEP-DOWN VOLTAGE REGULADOR D15V70F5S3



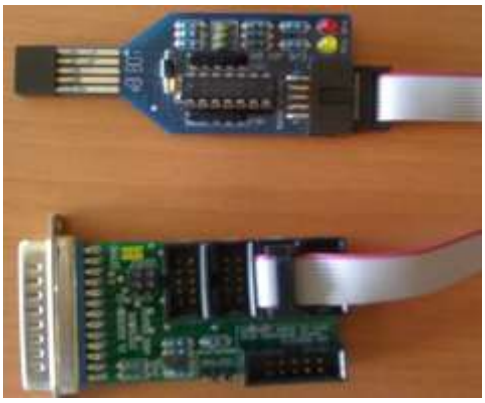
Regulador DC-DC de [4.5v-24v] de entrada a [3.3v, 5v] de salida seleccionable por JUMPER. Puede entregar hasta 7 amperios de corriente continua.

Este regulador dispone de protección ante polarización inversa, cortocircuitos y auto apagado por sobrecalentamiento.



[18] www.pololu.com

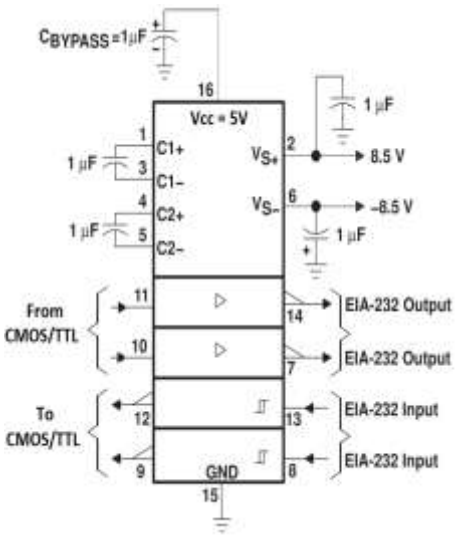
PROGRAMADOR IMU.

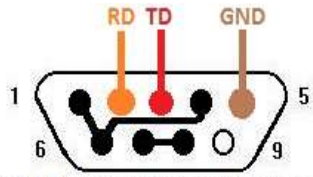


Se trata de un programador serie, por puerto paralelo, para micro controladores. Lo utilizamos conjuntamente con el software genérico de programación de circuitos integrados PonyProg.

INTERFICIE RS232

Utilizamos el driver MAX232 para montar el circuito de interconexión entre la IMU y el PC. El circuito es el recomendado por el fabricante en sus hojas de características. Además utilizaremos un adaptador RS232 a USB, confeccionando el conector.





DB9 MACHO (lado soldaduras)

Pin 1 - Data Carrier Detect	(DCD)
Pin 2 - Received Data	(RD)
Pin 3 - Transmit Data	(TD)
Pin 4 - Data Terminal Ready	(DTR)
Pin 5 - Signal Ground	(SG)
Pin 6 - Data Set Ready	(DSR)
Pin 7 - Request To Send	(RTS)
Pin 8 - Clear To Send	(CTS)
Pin 9 - Ring Indicator	(RI)

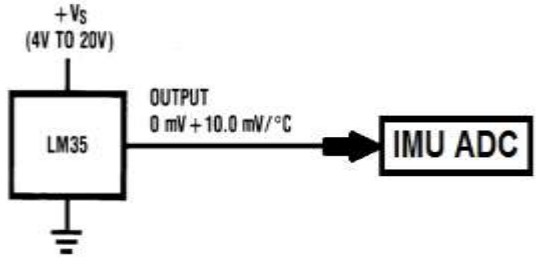
SENSOR TERMICO LM35 (NATIONAL INSTRUMENTS[19])

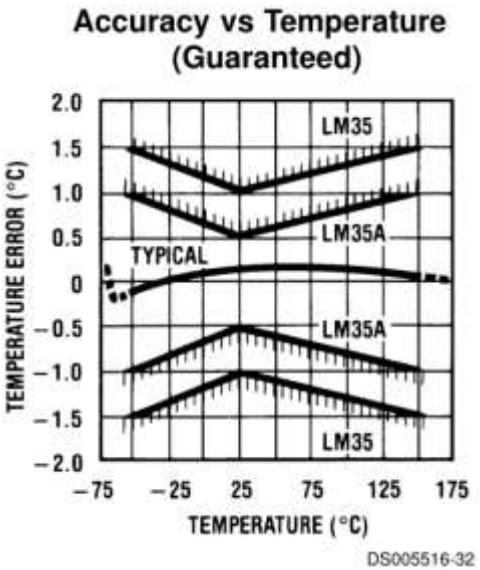
Incorporamos el sensor térmico LM35 (TO-92) de National Instruments al sistema, para poder tener referencias térmicas con las lecturas de los sensores de la IMU.

Características:

- *Calibrado en º Celsius.*
- *Factor de escala lineal: +10mV/ºC.*
- *Precisión (@25ºC): 0.5ºC.*
- *Rango térmico: -55ºC a +150ºC.*
- *Alimentación: 4V-30V.*
- *Consumo (@5V): 91uA (Typ).*
- *No linealidad: ±0,25ºC (Typ).*
- *Baja impedancia de salida.*
- *Bajo coste.*
- *Baja tasa de auto calentamiento: 0.08ºC (aire)*

Se implementa el circuito básico de sensado, puesto que para la aplicación que nos ocupa es suficiente.





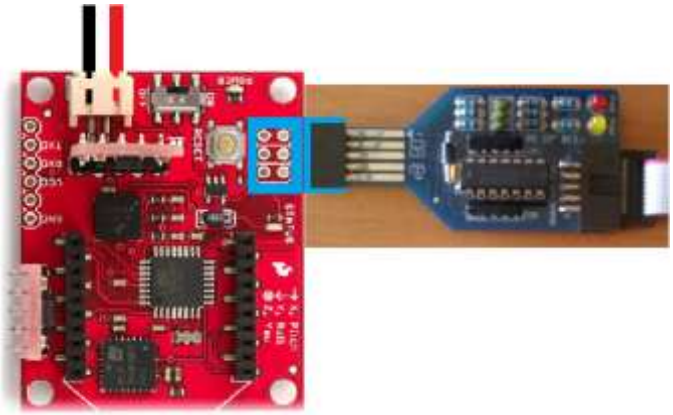
[19] www.ni.com

4. MODELADO DE ERRORES

a) Sistema de test.

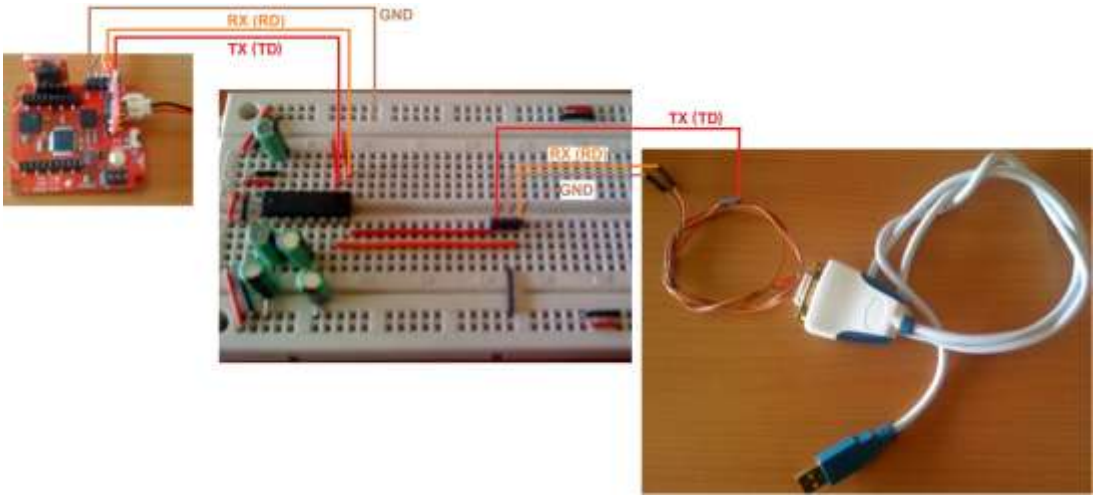
En primer lugar preparamos el módulo SPARKFUN SEN-09184 Atomic IMU 6 DOF para nuestra aplicación. Para ello, descargamos el código fuente básico que ofrece el fabricante para una configuración rápida inicial.

Tras la revisión de dicho código y su modificación para nuestras necesidades, procedemos a programar la IMU mediante el programador paralelo a serie y el software PonyProg.



Finalizado el proceso de programación procedemos a testear el correcto funcionamiento del módulo con la programación realizada.

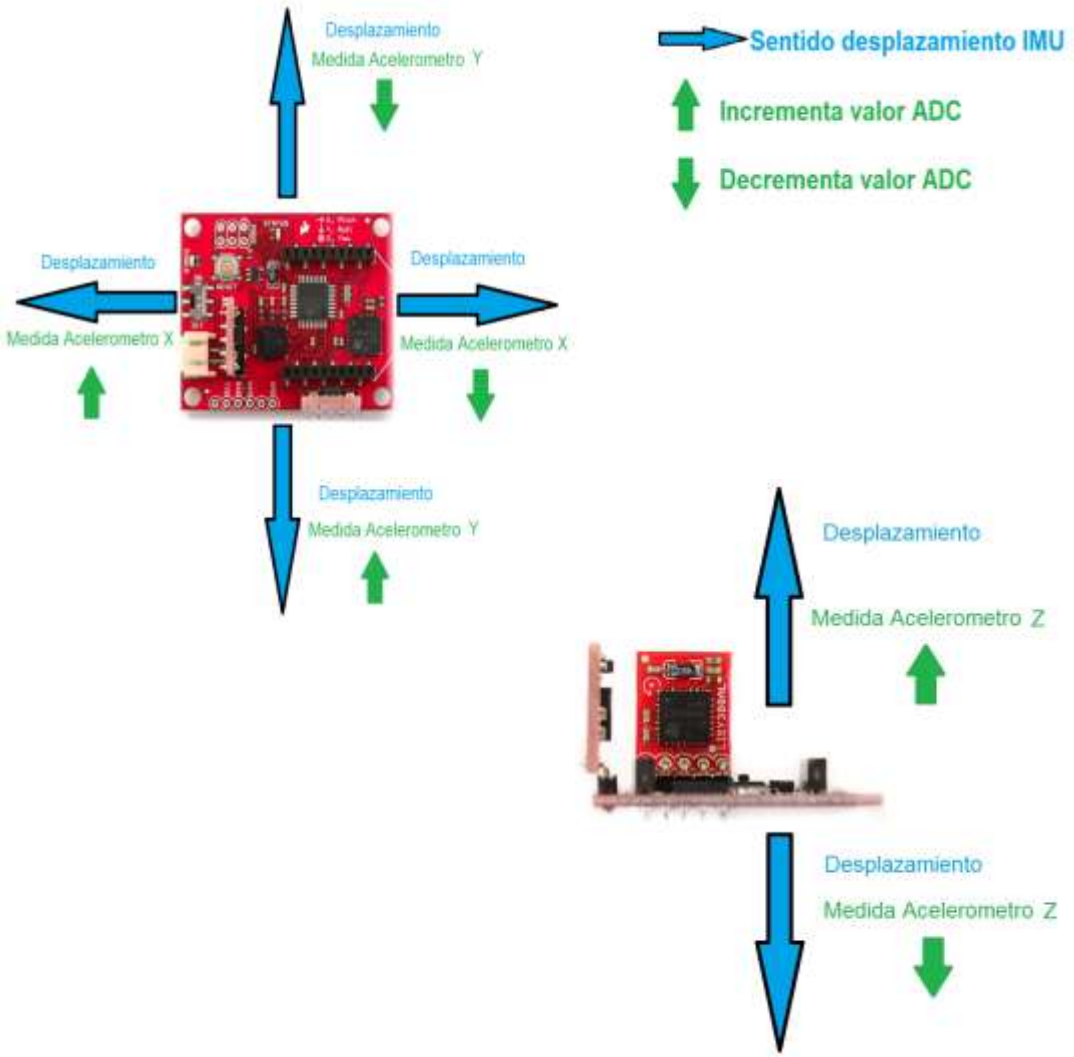
Conectamos la IMU al interface RS232 y éste al puerto USB del ordenador, mediante el adaptador RS232-USB



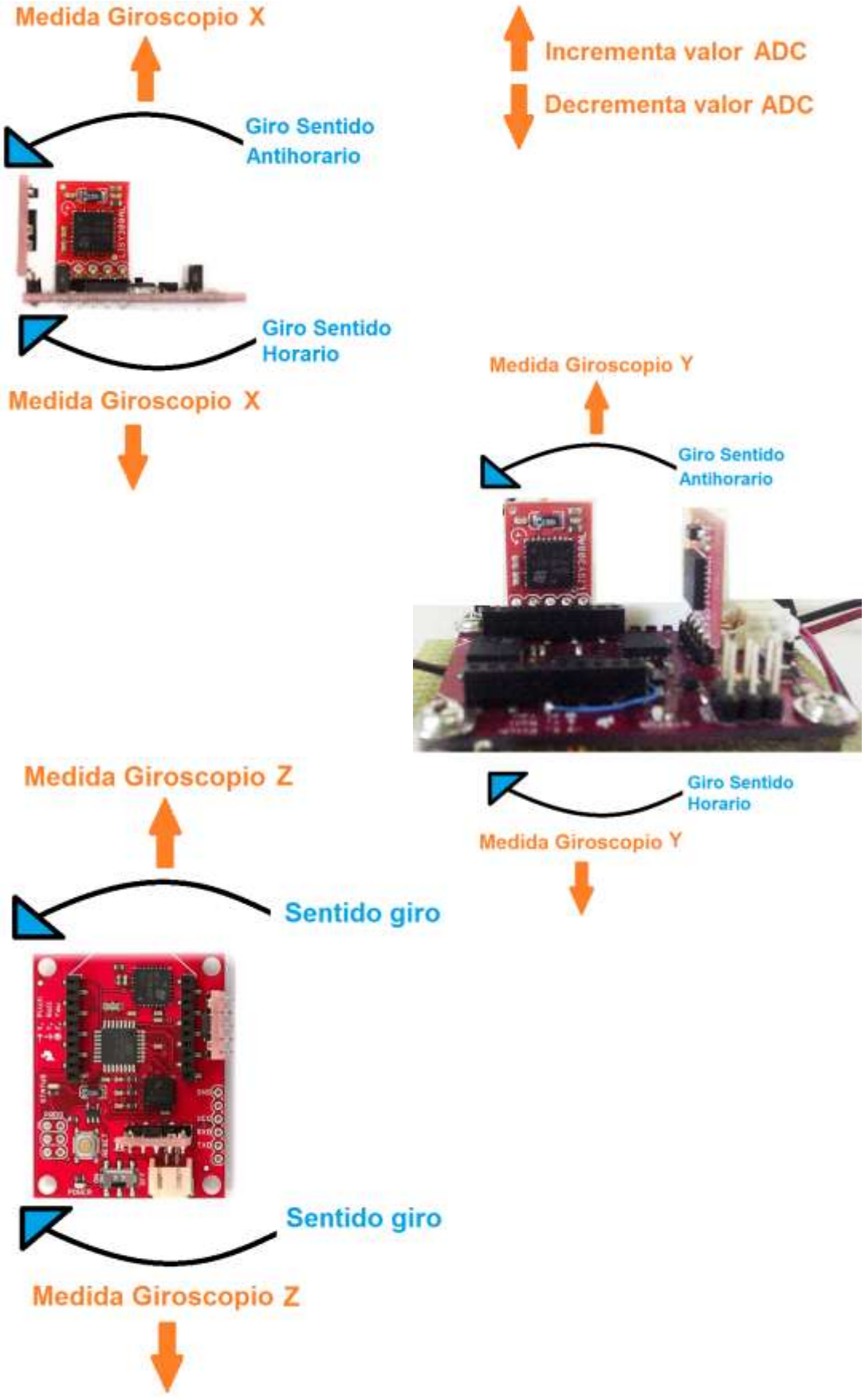
Alimentamos la IMU y el circuito RS232 con una fuente de laboratorio regulada a 5V. Se conectan todos los puntos de masa de cada circuito a la masa de la fuente, evitando así que puedan aparecer diferentes puntos de masa y se varíen las condiciones de alimentación y funcionamiento entre etapas. (IMU, interface RS232, adaptador USB y PC)

Una vez realizadas las pruebas de conexión y verificadas la conexión y la recepción de datos, procedemos a verificar las condiciones de funcionamiento del módulo. Esto es, el sentido de las variaciones en los datos entregados por el ADC de la IMU.

Para el caso de los acelerómetros, se muestran los resultados obtenidos en los siguientes diagramas:



Para los giroscopios se obtienen los resultados mostrados a continuación:



Una vez obtenidas las relaciones entre los movimientos y sus respectivas variaciones en las medias sustituimos la interficie RS232-USB por el módulo bluetooth BLUEMORE200, conectado a la IMU, y un interfaz genérico USB Bluetooth, conectado al PC. De esta manera dotamos al sistema de cierta libertad de movimiento.

El paso siguiente es disponer de un sistema de alimentación a baterías, para así disponer de libertad y autonomía suficientes para poder trabajar con la IMU de forma adecuada. Pudiendo moverla a nuestro antojo para obtener datos variantes y poder llevar a cabo un estudio dinámico del sistema completo (FPGA+GPS+IMU+ MAGNETOMETRO).

En nuestro caso solo se realiza el estudio en condiciones estáticas conocidas y controladas.

Situamos la IMU en una superficie lisa y lo mejor nivelada posible. Con ayuda de los soportes de la plataforma y de un nivel, ajustamos al máximo la horizontalidad del módulo sobre el plano de la superficie de apoyo (mesa).

Para llevar a cabo dicha tarea se propone una estructura modular por pisos.

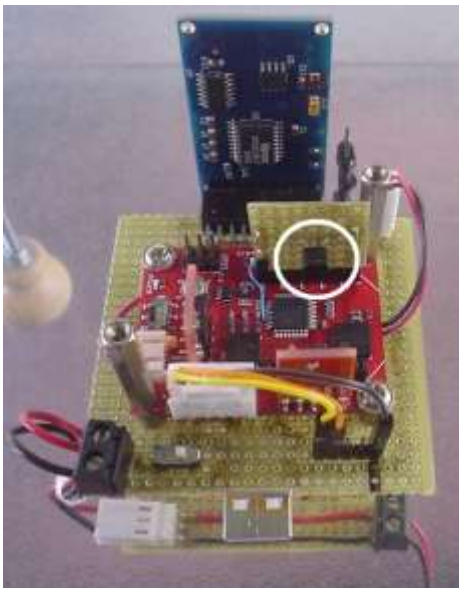
En el piso superior se monta la IMU sobre una placa de desarrollo, en la que se incorporan:

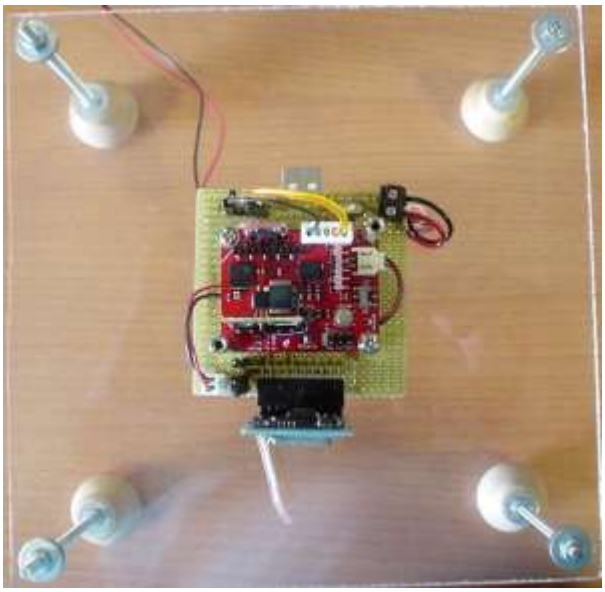
- Zócalo 2x10 pines hembra, para conectar Bluemore200 (o la FPGA).
- Conector USB macho PCB, para alimentar el sistema desde un PC.
- Conector placa a cable 2 polos con tornillos, para cables de la batería (piso inferior).
- Mini switch 2 circuitos, para seleccionar alimentación USB o batería.

El piso central es una plancha de metacrilato de 20x20cm que incorpora en sus esquinas unas patas con sinfín de rosca métrico 4, arandelas y tuercas. Con esto podemos ajustar de forma bastante precisa las inclinaciones y así controlar en estado de reposo nivelado para los test de adquisición estática.

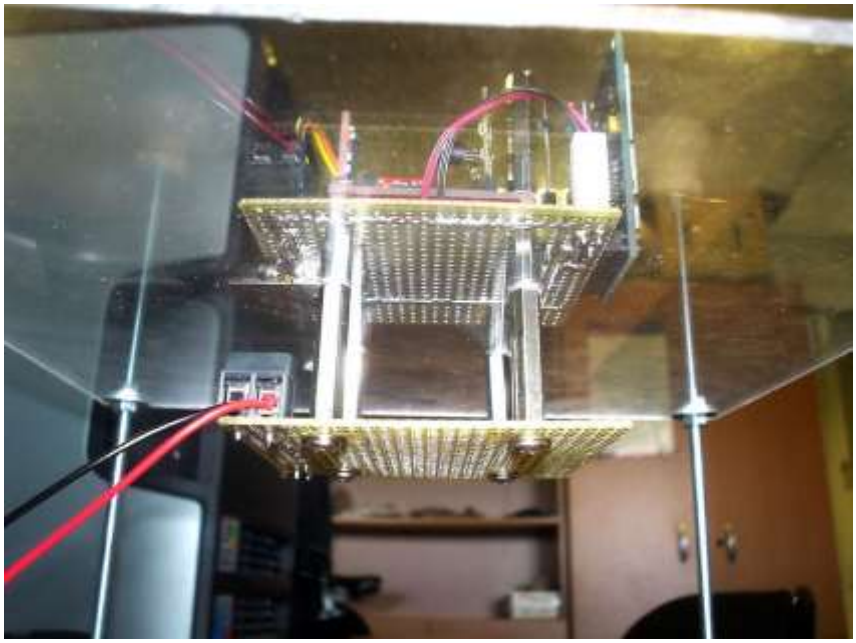
El piso inferior, bajo la estructura de metacrilato, es una pequeña plataforma en la que sujetar la batería de 7,2V y el regulador a 5V.

Así pues el sistema queda de la forma:





Se aprecia en las imágenes anteriores el sensor térmico añadido (circulo blanco), colocado en un zócalo destinado a módulos XBEE (sin uso en esta aplicación)



b) Obtención de datos.

Para la lectura de datos enviados desde la IMU al ordenador utilizamos una aplicación genérica de conexión mediante puertos COM, integrada en el sistema operativo (MS Windows® XP), llamada HIPERTERMINAL.

Con esta herramienta podemos establecer conexiones con otros equipos, gestionando los puertos de comunicaciones del equipo, configurando bitrates, paridad, bits de parada, el puerto COM a utilizar, etc.

En nuestro caso, utilizamos dos módulos bluetooth, uno en el PC y el otro la IMU, con un bitrate de 115.200 bit/s.

Debemos tener en cuenta que los datos son los valores de salida de la conversión del ADC. Por tanto, a la hora de trabajar con ellos debemos cambiar de unidades para que el procesado Allan Variance sea el adecuado. Esto es, debemos convertir a unidades físicas, [m/s] para los acelerómetros y [º/s] para los giroscopios.

Se realizan tres series de tomas de muestras a tres frecuencias de muestreo diferentes, tomando para cada caso 3,24millones de muestras:

- 100Hz (*10 milisegundos/muestra*): 3.240.000 muestras (*9 horas*).
- 200Hz (*5 milisegundos/muestra*): 3.240.000 muestras (*4 horas y 30 minutos*).
- 250Hz (*4 milisegundos/muestra*): 3.240.000 muestras (*3 horas y 36 minutos*).

Se asignan los 10bits disponibles del conversor ADC para poder obtener la mayor resolución de conversión posible.

Se han probado frecuencias de muestreo y tiempos de adquisición diferentes en los tres casos para determinar si el modelo de error obtenido tiene dependencia con la frecuencia de muestreo o con el tiempo de adquisición de las muestras del estudio.

Los datos leídos desde HIPERTERMINAL son almacenados en un fichero de texto plano (TXT) para su posterior procesado.

Puesto en marcha todo el sistema e iniciada la comunicación con la IMU empezamos a recibir los datos desde el módulo.

El formato de los datos recibidos es tipo tabla de siete columnas:

***Aceleración Lineal en* X**	***Aceleración Lineal en* Y**	***Aceleración Lineal en* Z**	***Aceleración Angular en X (*ROLL*)***	***Aceleración Angular en Y (*PITCH*)***	***Aceleración Angular en Z (*YAW*)***	***Temperatura***
496	492	790	519	498	498	63
497	496	791	519	499	498	63
503	490	789	521	497	497	63
498	496	790	519	497	497	68
497	497	790	521	497	497	51
500	494	790	520	498	498	66
499	492	795	519	497	498	66
499	490	793	520	498	498	67
499	492	791	520	498	497	67
500	493	791	519	496	497	52
501	498	790	520	497	498	67
498	492	793	520	497	498	67
501	495	789	519	497	498	67
498	497	793	520	498	497	67
497	494	790	520	498	498	67
503	494	792	519	497	497	67
496	494	791	520	498	498	50
498	495	792	520	497	497	68
498	491	793	520	497	498	65

Desde el menú ***TRANSFERIR*** de la aplicación, seleccionamos la opción ***CAPTURAR TEXTO*** para generar el fichero TXT de muestras. Posteriormente adquiriremos los datos necesarios para el modelado desde este fichero TXT.

c) Procesado de datos.

Desde el entorno de trabajo Matlab importamos los datos desde el archivo TXT y los separamos por variables:

- **Max**: *datos ADC del acelerómetro del eje X*
- **May**: *datos ADC del acelerómetro del eje Y*
- **Maz**: *datos ADC del acelerómetro del eje Z*
- **Mwr**: *datos ADC del giroscopio del eje X (ROLL)*
- **Mwp**: *datos ADC del giroscopio del eje Y (PITCH)*
- **Mwy**: *datos ADC del giroscopio del eje Z (YAW)*
- **Termo**: *datos ADC del sensor térmico (LM35)*

Se establecen las siguientes constantes para todo el procesado:

- Vref = 3.3V	*Tensión Referencia del ADC*
- w_s = 3.3*10$^{(-3)}$[V/(º/s)]	*Sensibilidad giroscopios*
- T_s = 10$^{(-3)}$ [V/º]	*Sensibilidad sensor térmico*
- dt = 1/f_s	*Periodo de muestreo*
- g = 9.80665 [m/s^2]	*Constante gravedad*
- T_Data = (dt)*longitud vectores datos	*Duración temporal vectores de datos*

En primer lugar debemos adaptar las lecturas obtenidas de los sensores a las magnitudes físicas correspondientes.

Las expresiones de conversión del ADC y su inversa son las siguientes:

$$ADC = 1024 * \left(\frac{V_{in}}{V_{ref}} \right) \Leftrightarrow V_{out}^{sensor} = ADC \left(\frac{V_{ref}}{1024} \right)$$

Para obtener la tensión de entrada al conversor ADC, que es la de salida del sensor, nos quedamos con la segunda de las expresiones anteriores. ($V_{in} = V_{out}^{sensor}$)

Obtenemos, con ayuda de la expresión anterior, los datos convertidos a tensiones. A continuación seguimos trabajando los datos para disponer de ellos de forma que nos sea cómodo y sencillo interpretar físicamente las medidas obtenidas.

- Sensor térmico:

$$V_{out}^{sensor} = ADC\left(\frac{V_{ref}}{1024}\right)$$

$$Temp = \frac{V_{out}^{sensor}[V]}{Sensibilidad\left[mV/_{\circ}\right]}$$

En el caso de las medidas de temperatura, aplicamos un filtrado a los datos ya convertidos a grados. Puesto que no es viable, en condiciones ambientales normales, que la temperatura varíe bruscamente de un segundo al siguiente, se promedian los valores térmicos por segundos a lo largo de toda la duración de los datos obtenidos.

Partiendo de la expresión de V_{out}^{sensor}, vamos a desplazar los niveles de tensión de referencia, centrando las medidas en V_{ref}. Para los datos de los sensores inerciales, acelerómetros y giroscopios:

$$V_{OUT}^{ADC} = ADC\left(\frac{V_{ref}}{1024}\right) - \left(\frac{V_{ref}}{2}\right)$$

Hecho esto, centramos las medidas en cero para tomar los valores positivos de tensión como incrementos positivos de la magnitud física y los valores negativos de tensión como incrementos negativos de la magnitud física:

$$V_{ZERO}^{ADC} = V_{OUT}^{ADC} - media\left(V_{OUT}^{ADC}\right)$$

- Acelerómetros:

$$a_{lineal} = \frac{V_{out}^{sensor}[V]}{Sensibilidad\left[mV/g\right]}$$

Integramos la aceleración para obtener la velocidad, en dominio discreto:

$$v_{lineal}[\tau] = v_{lineal}[\tau - 1] + a_{lineal}[\tau] \cdot \partial\tau$$

Integramos la velocidad para obtener la posición, en dominio discreto:

$$s_{lineal}[\tau] = s_{lineal}[\tau - 1] + v_{lineal}[\tau] \cdot \partial\tau$$

Este proceso se realiza para cada uno de los 3 acelerómetros, obteniendo de esta manera un primer vector de desplazamiento que refleja el movimiento físico del módulo inercial.

- Giroscopios:

$$\omega_{angular} = \frac{V_{out}^{sensor}[V]}{Sensibilidad\left[mV/^{o}/s\right]}$$

Integramos la velocidad angular para obtener los grados, en dominio discreto:

$$\theta[\tau] = \theta[\tau - 1] + \omega_{angular}[\tau] \cdot \partial\tau$$

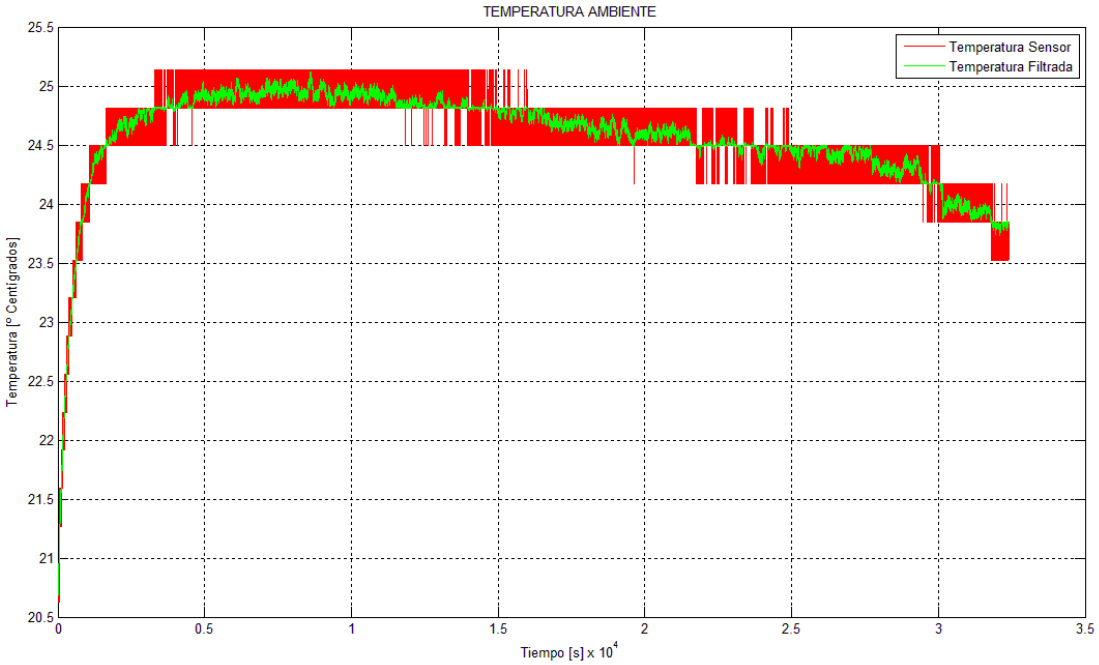
Realizando este proceso para cada uno de los 3 giroscopios, obtenemos igualmente un segundo vector de desplazamiento que refleja el movimiento físico de la IMU.

d) Modelos de error de los sensores.

Como se comentó anteriormente hemos realizado los test a tres frecuencias de muestreo diferentes, para estudiar si existe dependencia frecuencial de los modelo de error obtenidos.

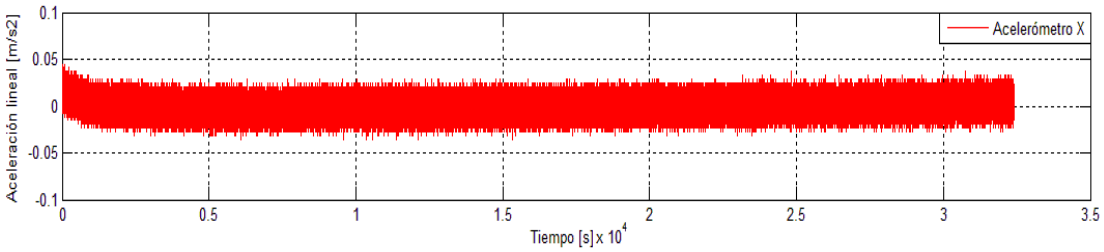
A continuación presentaremos, con las gráficas correspondientes, los resultados obtenidos para la frecuencia de muestreo F_s = 100Hz. Para las gráficas detalladas de F_s = 100Hz y los resultados graficados a las frecuencias F_s = 200Hz y F_s = 250Hz dirigirse al Anexo D.

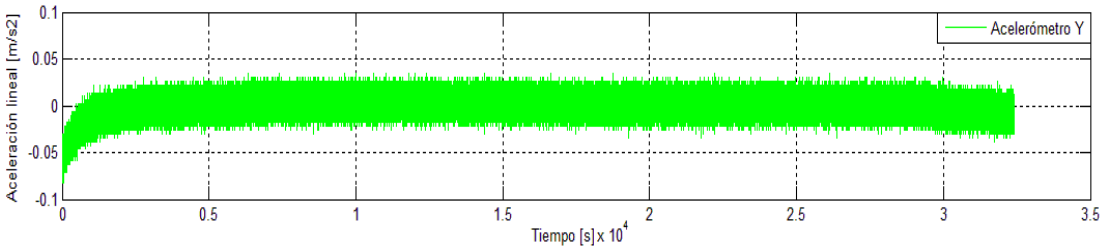
Observamos los resultados de la operación de filtrado de las lecturas de temperatura:

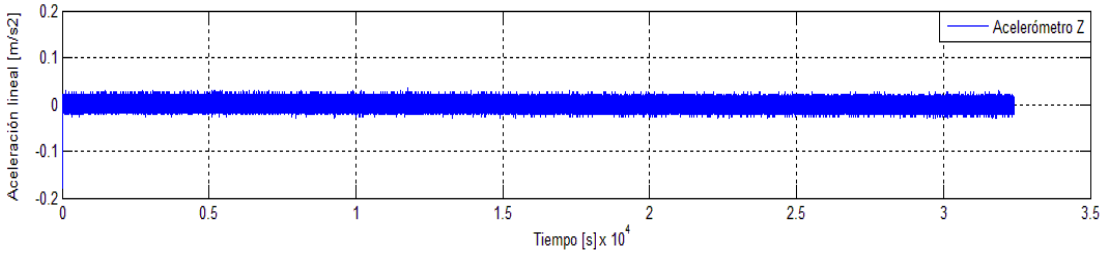


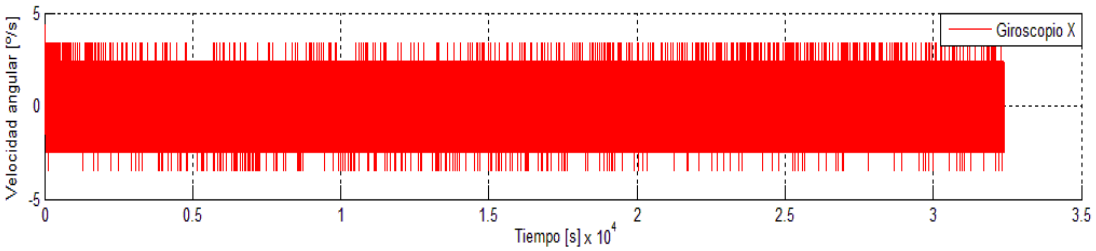
En este trabajo no haremos nada con los datos de la temperatura, pero queda el sistema preparado para futuras mejoras de los modelos, completándolos con la dependencia térmica.

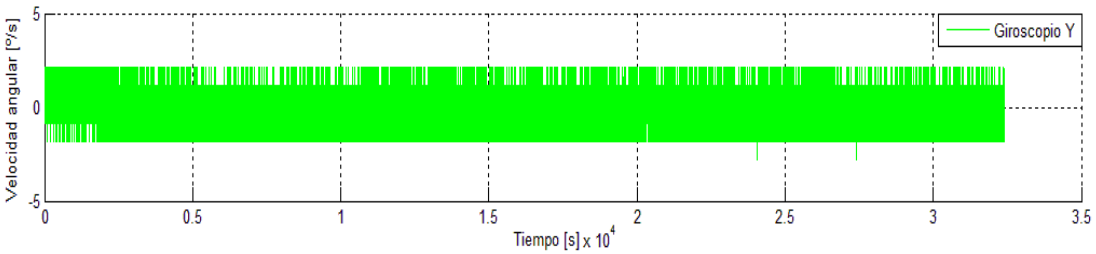
Presentamos a continuación las lecturas de los acelerómetros y giroscopios una vez centradas las medidas sobre las referencias de cero:

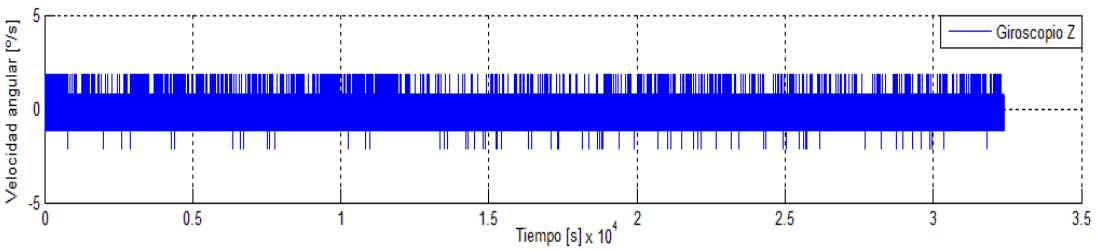




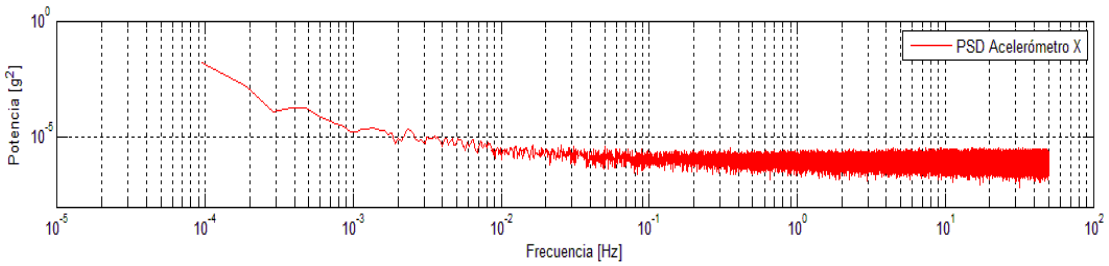


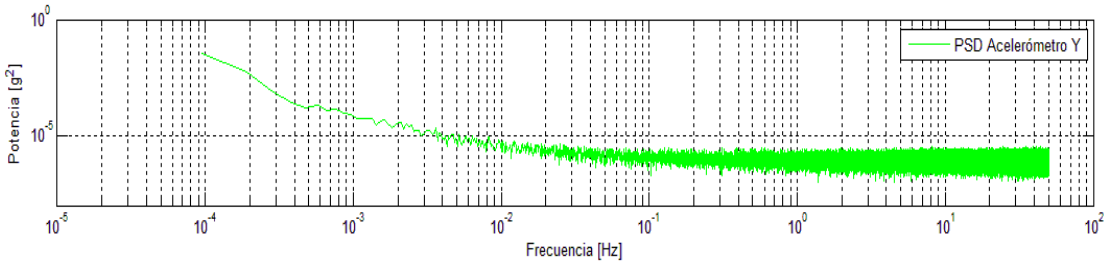


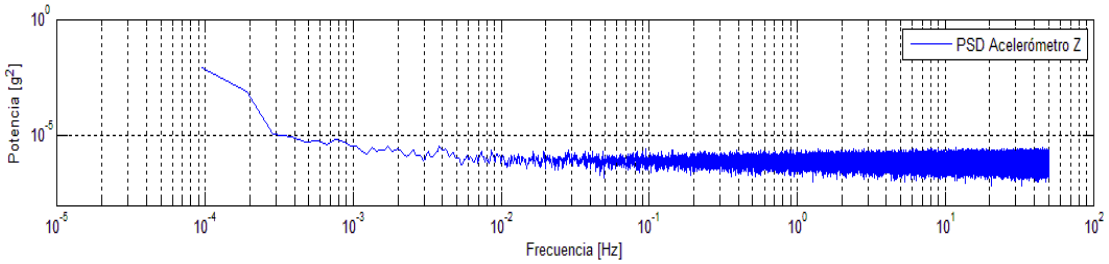


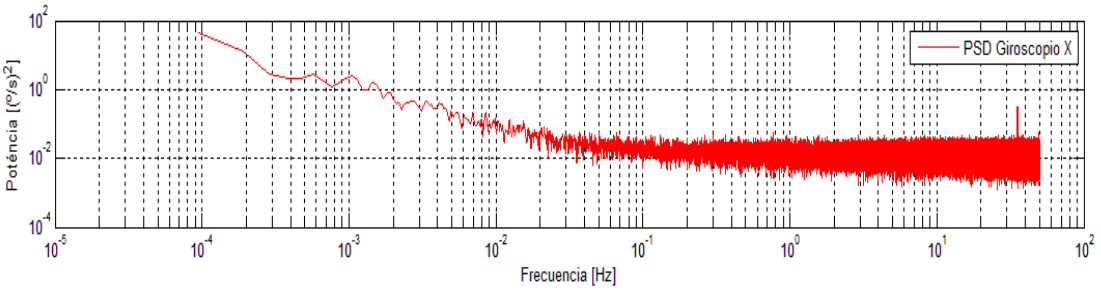


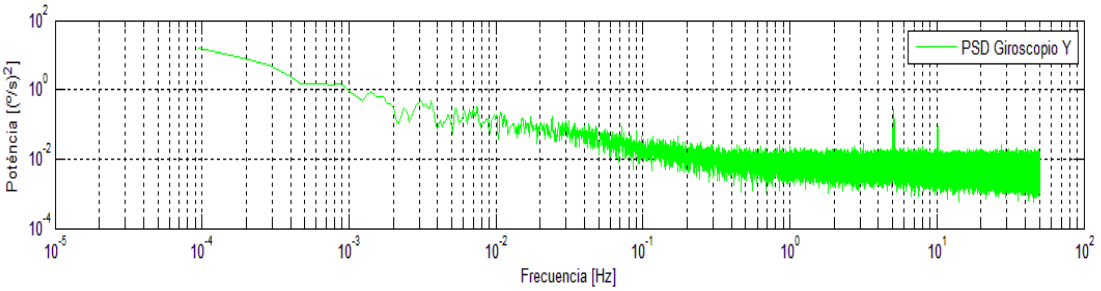
Continuamos con la densidad espectral de potencia del ruido de los sensores:

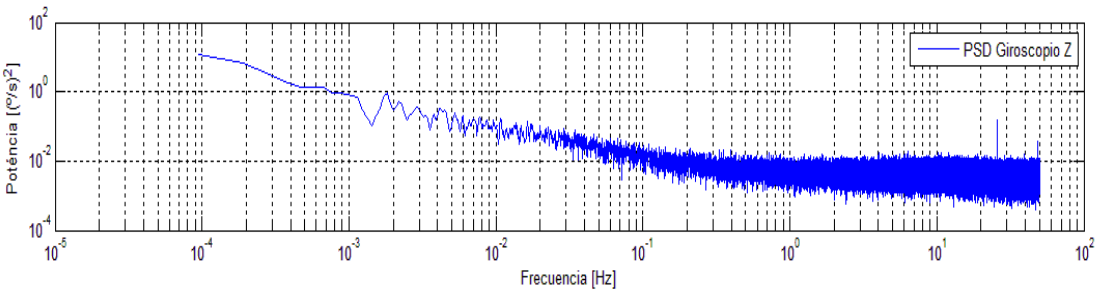






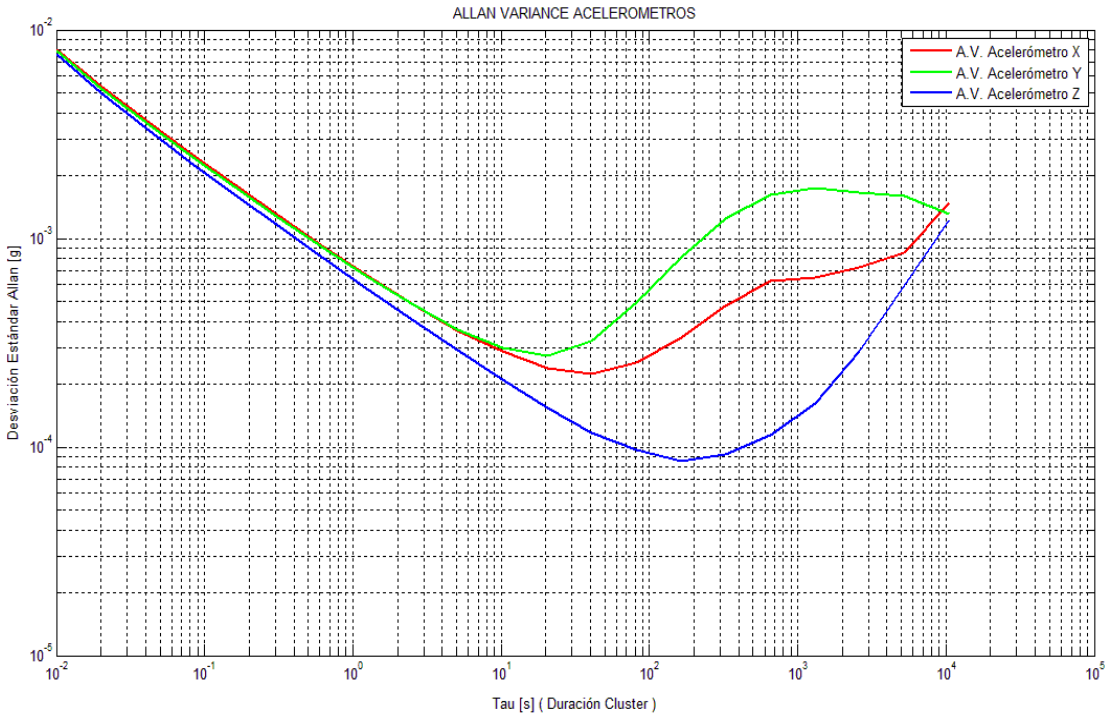




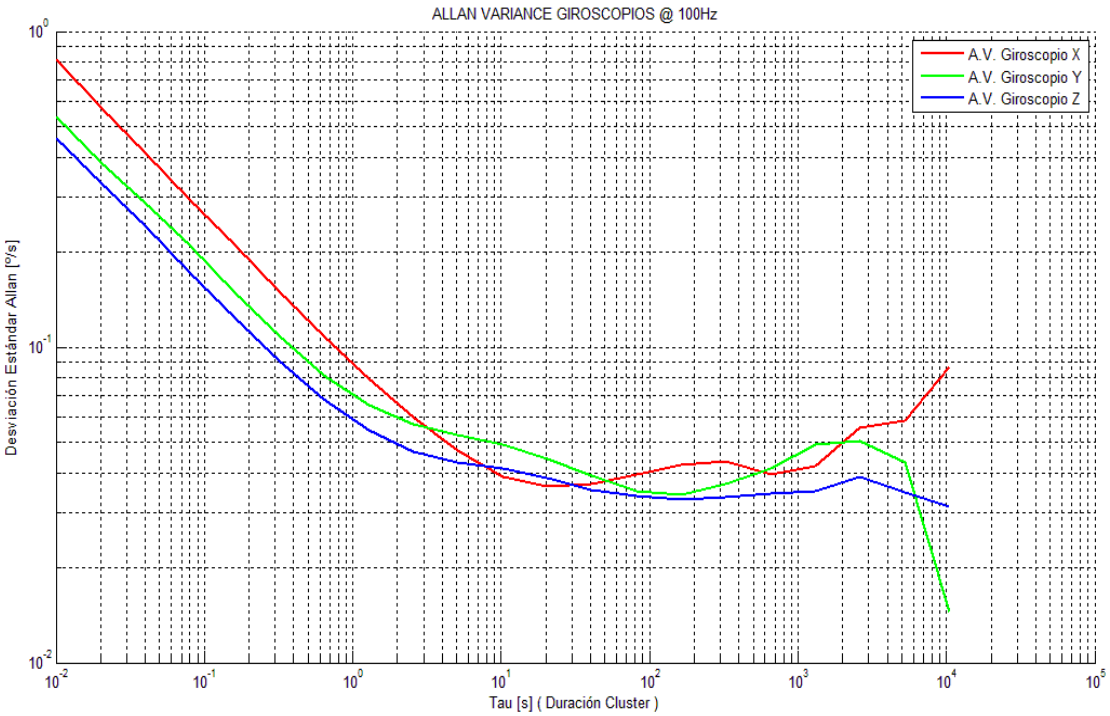


A continuación procedemos a calcular la Varianza Allan para poder obtener las constantes que nos permitirán implementar los modelos de error de los sensores.

Los resultados gráficos del análisis de Varianza Allan son:

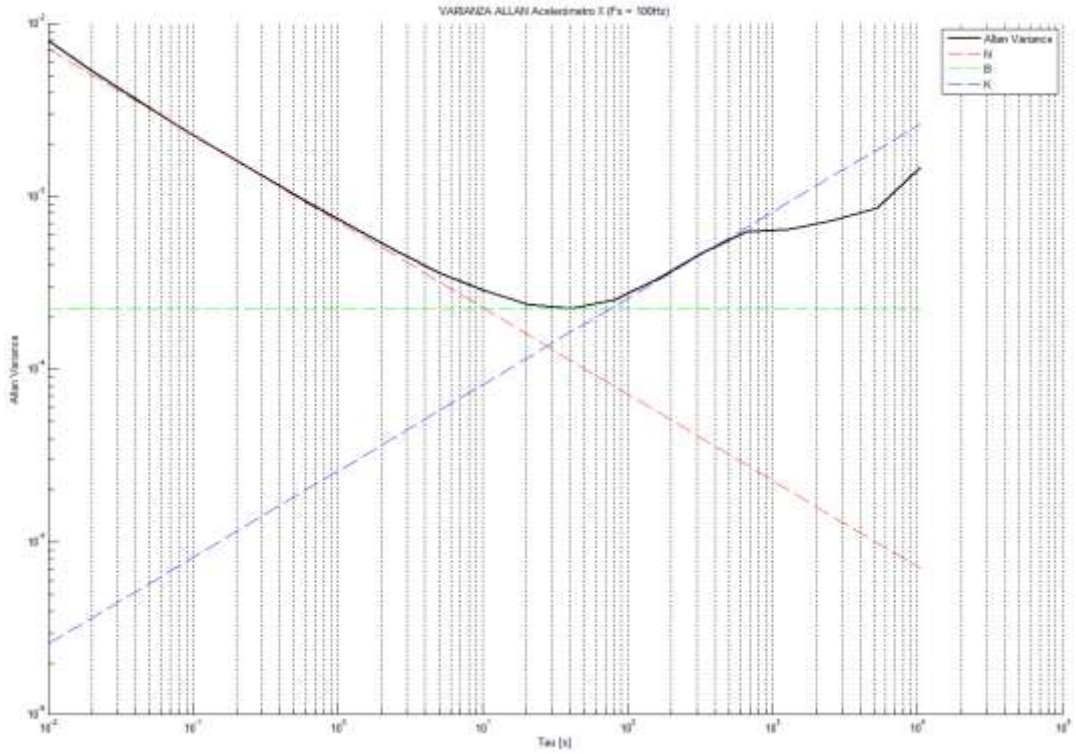


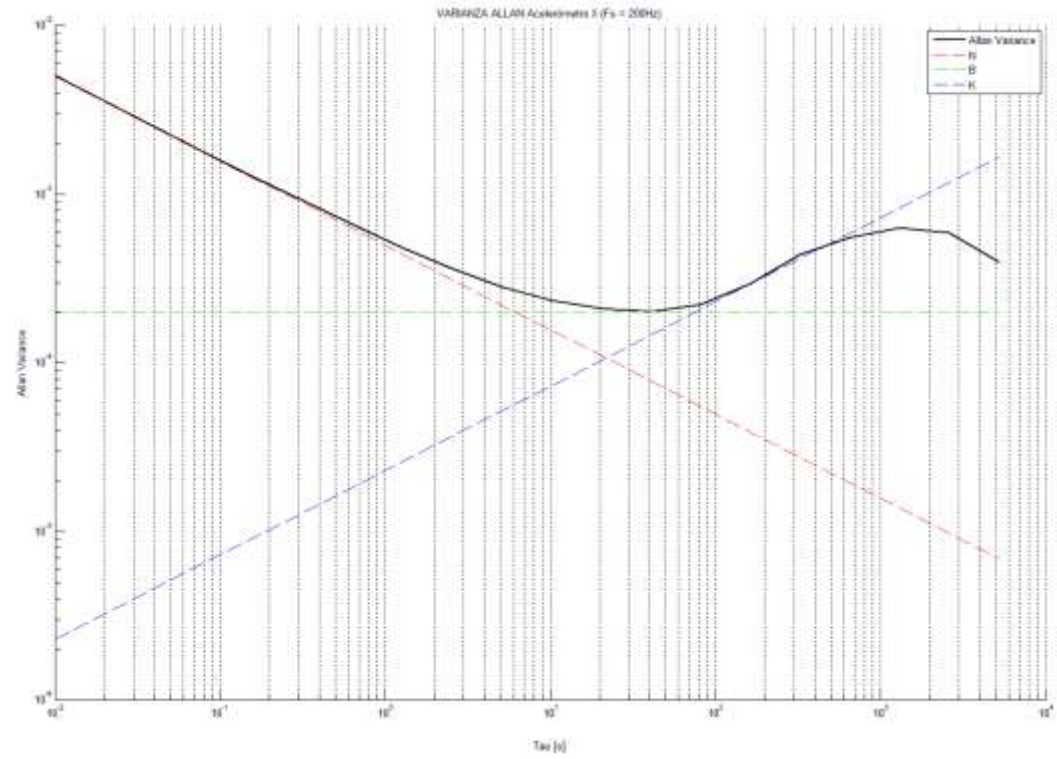
Los resultados gráficos del análisis de Varianza Allan para los giroscopios son:

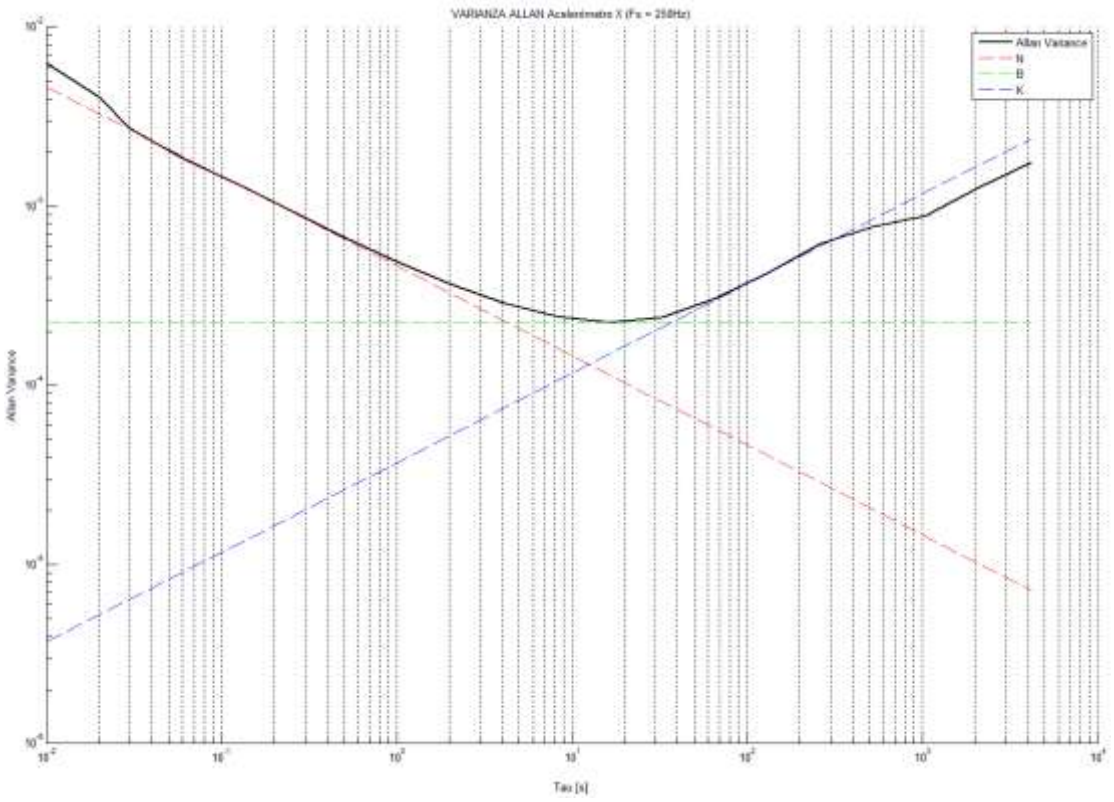


De las gráficas de Allan Variance extraemos los datos necesarios para la composición de los modelos de error de cada uno de los sensores a cada una de las tres frecuencias de muestreo.

Veamos los ejemplos con el acelerómetro X a las tres frecuencias:







De la teoría de la Varianza de Allan, sabemos que para obtener cada constante debemos buscar el punto en el que su recta delimitadora cruza con una asíntota paralela al eje de ordenadas, eje Y, en el instante de tiempo adecuado.

Tomamos como límite superior de TAU el valor 10^3, puesto que es a partir de este valor es donde se aprecia el comportamiento erróneo de las gráficas de Varianza Allan en la determinación de las constantes de error.

Recordemos pues donde corresponde cada uno:

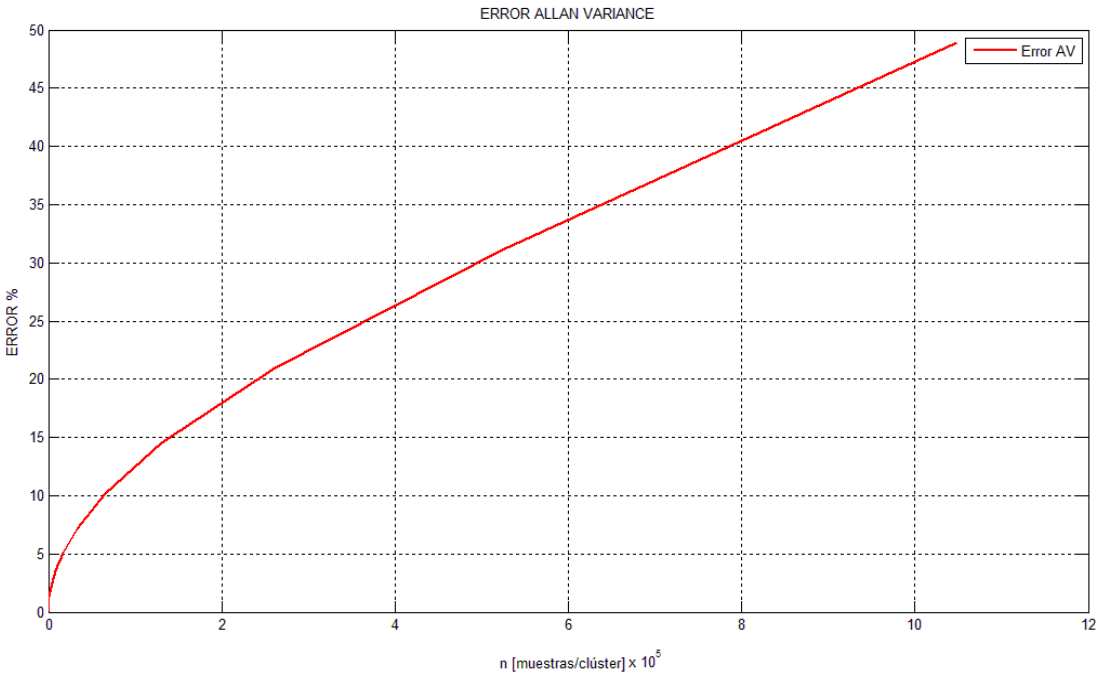
CONSTANTE	**Q**	**N**	**K**	**R**
TIEMPO (s)	$\sqrt{3}$	**1**	**3**	$\sqrt{2}$

Así pues, obtenemos los valores de las constantes de error para cada sensor y cada frecuencia de muestreo.

Con las constantes ya determinadas procedemos a determinar cada uno de los modelos de error de cada sensor[20].

Además extraemos el error de la Varianza Allan para cada tamaño de clúster, utilizando la definición[21]:

$$\sigma(\delta) = \frac{1}{\sqrt{2\left(\frac{N}{n} - 1\right)}} \quad \text{siendo} \begin{cases} \sigma(\delta): \text{ \% error estimación} \\ N: \text{ total muestras del test} \\ n: \text{ muestras del clúster} \end{cases}$$



Se extrae de la figura del error de Allan Variance, que cuanto mayor es el tamaño del clúster mayor es el error que se comete al estimar la varianza.

Con esta definición podemos además centrarnos en el análisis de cada una de las componentes del error, determinando la desviación del parámetro correspondiente puesto que conocemos el tamaño y duración de clúster para el instante de tiempo en que aparece cada error.

[20] Los modelos de los errores están extraídos de Ref.[2]
[21] Página 44 de Ref.[1]

Angular/Velocity Randon Walk (N)

Se modela, en dominio discreto, como un ruido blanco de desviación estándar:

$$\sigma_{AVRW} = \sqrt{\sigma^2_{sensor} - \sigma^2_{Q} - \sigma^2_{Bias} - \sigma^2_{RRW} - \sigma^2_{RateRamp}}$$

Bias (B)

Se modela, en dominio discreto, de la forma:

$$N_{BIAS}(k+1) = a_d N_{BIAS}(k) + b_d \eta(k)$$

Acceleration/Rate Random Walk (K)

Se modela, en dominio discreto, como un ruido blanco filtrado.

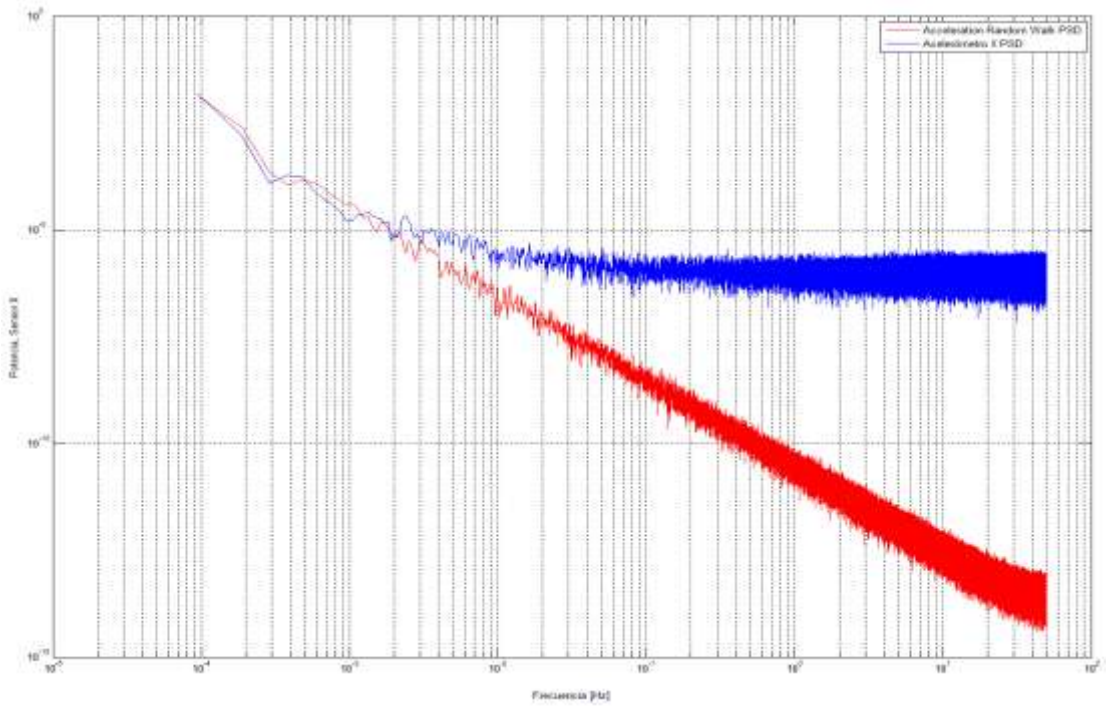
$$rrw(z) = K \cdot dfilt(z) \cdot w(z)$$

$$dfilt(z) = \frac{\Delta T_s}{(z-1)}$$

En este punto debemos determinar la varianza, σ^2, de $\omega(z)$, que está definido como un ruido blanco.

Para determinar el valor de la varianza debemos aproximar las gráficas de PSD del sensor y PSD del modelo "*rrw(z)*" en su parte de baja frecuencia, origen de las gráficas. Esto lo haremos variando la desviación estándar de $\omega(z)$ hasta conseguir la mayor proximidad entre las gráficas.

Para F_s = 100Hz y el Acelerómetro X, con $\sigma_{\omega(z)}$ = 5 obtenemos la gráfica:



Una vez obtenida la desviación estándar adecuada y su Acceleration/Rate Random Walk asociado, podemos determinar el Angular/Velocity Random Walk, definido como un ruido blanco, cuya desviación estándar es:

$$\sigma_{ARW} = \sqrt{\sigma^2_{ruido\_sensor} - B^2 - \sigma^2_{RRW}}$$

Una vez obtenidos los valores de desviación estándar de cada uno de los errores de cada sensor podemos proceder a generar los modelos de ruido completos.

Quedando pues dicho modelo definido como:

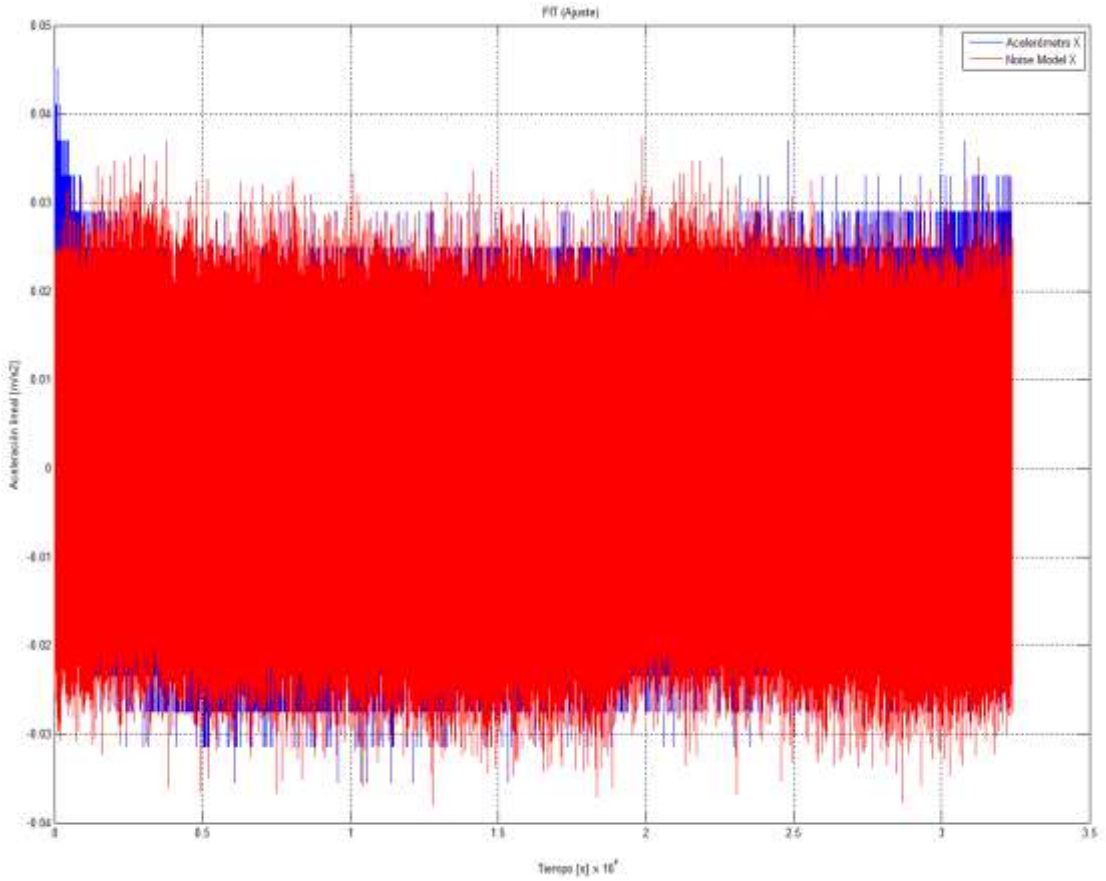
$$N_{sensor} = N_{BIAS} + N_{RRW} + N_{AVRW}$$

Con el modelo determinado comprobamos el FIT para verificar la eficacia que tiene el modelo obtenido.

Para obtener el FIT, coincidencia en tanto por ciento, de los modelos obtenidos, haremos uso de la expresión:

$$FIT = \left(1 - \frac{\sqrt{\sum_{k=1}^{L}\left(x_{sensor}(k) - x_{modelo}(k)\right)^2}}{\sqrt{\sum_{k=1}^{L}\left(x_{sensor}(k)\right)^2}}\right)100\%$$

Para F_s = 100Hz y el Acelerómetro X obtenemos: ***FIT = 98,58%***



En el Anexo D se presentan todas las gráficas detalladas de cada uno de los sensores para las tres frecuencias de muestreo.

5. RESULTADOS Y CONCLUSIONES

Llegados a este punto resta presentar los datos obtenidos de los sensores. Esto es cada error y su desviación correspondiente.

Los datos expuestos corresponden a los sensores para cada una de las tres frecuencias de muestreo testeadas: *100Hz, 200Hz y 250 Hz*

Para F_s= 100Hz:

Obtenemos los valores de las constantes de error y sus desviaciones:

SENSOR	N	B	K
Aceleróm. X	$(7{,}2\cdot10^{-4} \pm 2{,}83\cdot10^{-8})$ [m/s]	$(3{,}37\cdot10^{-4} \pm 8{,}48\cdot10^{-8})$ [m/s]	$(4{,}2\cdot10^{-5} \pm 2{,}86\cdot10^{-9})$ [m/s]
Aceleróm. Y	$(7{,}2\cdot10^{-4} \pm 2{,}83\cdot10^{-8})$ [m/s]	$(4{,}11\cdot10^{-4} \pm 7{,}31\cdot10^{-8})$ [m/s]	$(8{,}8\cdot10^{-5} \pm 5{,}99\cdot10^{-9})$ [m/s]
Aceleróm. Z	$(6{,}2\cdot10^{-4} \pm 2{,}44\cdot10^{-8})$ [m/s]	$(1{,}30\cdot10^{-4} \pm 6{,}53\cdot10^{-8})$ [m/s]	$(7{,}5\cdot10^{-6} \pm 5{,}10\cdot10^{-10})$ [m/s]
Giroscopio X	$(8{,}2\cdot10^{-2} \pm 3{,}22\cdot10^{-6})$ [º]	$(5{,}47\cdot10^{-2} \pm 9{,}74\cdot10^{-6})$ [º]	----------
Giroscopio Y	$(6{,}5\cdot10^{-2} \pm 2{,}55\cdot10^{-6})$ [º]	$(5{,}12\cdot10^{-2} \pm 2{,}58\cdot10^{-5})$ [º]	----------
Giroscopio Z	$(5{,}4\cdot10^{-2} \pm 2{,}12\cdot10^{-6})$ [º]	$(4{,}96\cdot10^{-2} \pm 2{,}50\cdot10^{-5})$ [º]	----------

Obtenemos los valores de desviación estándar en las medidas y los modelos y el FIT resultante:

SENSOR	σ_{SENSOR}	σ_{BIAS}	σ_{AVRW}	σ_{ARRW}	FIT
Acelerómetro X	$7{,}57\cdot10^{-3}$ [m/s^2]	$7{,}45\cdot10^{-4}$	$7{,}56\cdot10^{-3}$	5,00	98,58%
Acelerómetro Y	$9{,}43\cdot10^{-3}$ [m/s^2]	$1{,}29\cdot10^{-3}$	$9{,}42\cdot10^{-3}$	5,00	98,54%
Acelerómetro Z	$6{,}57\cdot10^{-3}$ [m/s^2]	$1{,}43\cdot10^{-4}$	$6{,}57\cdot10^{-3}$	9,00	98,58%
Giroscopio X	$8{,}27\cdot10^{-1}$ [º/s]	$1{,}71\cdot10^{-1}$	$8{,}25\cdot10^{-1}$	-----	98,59%
Giroscopio Y	$5{,}64\cdot10^{-1}$ [º/s]	$5{,}65\cdot10^{-2}$	$5{,}61\cdot10^{-1}$	-----	98,59%
Giroscopio Z	$4{,}78\cdot10^{-1}$ [º/s]	$5{,}48\cdot10^{-2}$	$4{,}76\cdot10^{-1}$	-----	98,59%

Para F_s= 200Hz:

Obtenemos los valores de las constantes de error y sus desviaciones:

SENSOR	N	B	K
Acceleróm. X	$(5{,}0\cdot10^{-4} \pm 2{,}78\cdot10^{-8})$ [m/s]	$(3{,}03\cdot10^{-4} \pm 1{,}08\cdot10^{-7})$ [m/s]	$(3{,}9\cdot10^{-5} \pm 3{,}75\cdot10^{-9})$ [m/s]
Acceleróm. Y	$(5{,}0\cdot10^{-4} \pm 2{,}78\cdot10^{-8})$ [m/s]	$(3{,}57\cdot10^{-4} \pm 8{,}98\cdot10^{-8})$ [m/s]	$(7{,}0\cdot10^{-5} \pm 6{,}74\cdot10^{-9})$ [m/s]
Acceleróm. Z	$(4{,}3\cdot10^{-4} \pm 2{,}39\cdot10^{-8})$ [m/s]	$(1{,}24\cdot10^{-4} \pm 8{,}85\cdot10^{-8})$ [m/s]	$(5{,}5\cdot10^{-6} \pm 5{,}29\cdot10^{-10})$ [m/s]
Giroscopio X	$(6{,}0\cdot10^{-2} \pm 3{,}33\cdot10^{-6})$ [º]	$(5{,}19\cdot10^{-2} \pm 9{,}24\cdot10^{-6})$ [º]	----------
Giroscopio Y	$(4{,}5\cdot10^{-2} \pm 2{,}50\cdot10^{-6})$ [º]	$(5{,}19\cdot10^{-2} \pm 2{,}62\cdot10^{-5})$ [º]	----------
Giroscopio Z	$(4{,}2\cdot10^{-2} \pm 2{,}33\cdot10^{-6})$ [º]	$(3{,}50\cdot10^{-2} \pm 7{,}33\cdot10^{-5})$ [º]	----------

Obtenemos los valores de desviación estándar en las medidas y los modelos y el FIT resultante:

SENSOR	σ_{SENSOR}	σ_{BIAS}	σ_{AVRW}	σ_{ARRW}	FIT
Acelerómetro X	$7{,}35\cdot10^{-3}$ [m/s^2]	$9{,}42\cdot10^{-4}$	$7{,}34\cdot10^{-3}$	4,00	98,59%
Acelerómetro Y	$9{,}40\cdot10^{-3}$ [m/s^2]	$1{,}58\cdot10^{-3}$	$9{,}39\cdot10^{-3}$	7,45	98,47%
Acelerómetro Z	$6{,}44\cdot10^{-3}$ [m/s^2]	$1{,}94\cdot10^{-4}$	$6{,}44\cdot10^{-3}$	13,50	98,58%
Giroscopio X	$8{,}24\cdot10^{-1}$ [º/s]	$3{,}25\cdot10^{-1}$	$8{,}22\cdot10^{-1}$	-----	98,59%
Giroscopio Y	$5{,}69\cdot10^{-1}$ [º/s]	$1{,}15\cdot10^{-1}$	$5{,}66\cdot10^{-1}$	-----	98,59%
Giroscopio Z	$4{,}63\cdot10^{-1}$ [º/s]	$1{,}93\cdot10^{-2}$	$4{,}61\cdot10^{-1}$	-----	98,58%

Para F_s= 250Hz:

Obtenemos los valores de las constantes de error y sus desviaciones:

SENSOR	N	B	K
Aceleróm. X	$(4{,}3\ 10^{-4} \pm 2{,}67\cdot10^{-8})$ [m/s]	$(3{,}38\cdot10^{-4} \pm 8{,}15\cdot10^{-8})$ [m/s]	$(6{,}2\ 10^{-5} \pm 6{,}67\cdot10^{-9})$ [m/s]
Aceleróm. Y	$(4{,}3\ 10^{-4} \pm 2{,}67\cdot10^{-8})$ [m/s]	$(4{,}43\cdot10^{-4} \pm 7{,}88\cdot10^{-8})$ [m/s]	$(1{,}4\ 10^{-4} \pm 1{,}51\cdot10^{-8})$ [m/s]
Aceleróm. Z	$(3{,}9\ 10^{-4} \pm 2{,}42\cdot10^{-8})$ [m/s]	$(1{,}36\cdot10^{-4} \pm 6{,}86\cdot10^{-8})$ [m/s]	$(9{,}0\ 10^{-6} \pm 9{,}68\ 10^{-10})$ [m/s]
Giroscopio X	$(3{,}3\ 10^{-2} \pm 2{,}05\ 10^{-6})$ [º]	$(5{,}27\cdot10^{-2} \pm 1{,}32\cdot10^{-5})$ [º]	----------
Giroscopio Y	$(2{,}4\ 10^{-2} \pm 1{,}49\ 10^{-6})$ [º]	$(4{,}47\cdot10^{-2} \pm 9{,}37\cdot10^{-5})$ [º]	----------
Giroscopio Z	$(1{,}8\ 10^{-2} \pm 1{,}12\ 10^{-6})$ [º]	$(4{,}30\cdot10^{-2} \pm 3{,}07\cdot10^{-5})$ [º]	----------

Obtenemos los valores de desviación estándar en las medidas y los modelos y el FIT resultante:

SENSOR	σ_{SENSOR}	σ_{BIAS}	σ_{AVRW}	σ_{ARRW}	FIT
Acelerómetro X	$7{,}59\cdot10^{-3}$ [m/s^2]	$1{,}87\cdot10^{-3}$	$7{,}58\cdot10^{-3}$	10,00	98,53%
Acelerómetro Y	$9{,}13\cdot10^{-3}$ [m/s^2]	$3{,}46\cdot10^{-3}$	$9{,}12\cdot10^{-3}$	10,00	98,30%
Acelerómetro Z	$6{,}47\cdot10^{-3}$ [m/s^2]	$3{,}76\cdot10^{-4}$	$6{,}47\cdot10^{-3}$	10,00	98,58%
Giroscopio X	$8{,}22\cdot10^{-1}$ [º/s]	$2{,}91\cdot10^{-1}$	$8{,}21\cdot10^{-1}$	-----	98,59%
Giroscopio Y	$5{,}69\cdot10^{-1}$ [º/s]	$3{,}08\cdot10^{-2}$	$5{,}67\cdot10^{-1}$	-----	98,59%
Giroscopio Z	$4{,}68\cdot10^{-1}$ [º/s]	$8{,}40\cdot10^{-2}$	$4{,}66\cdot10^{-1}$	-----	98,59%

CONCLUSIONES

Con respecto a las desviaciones estándar de las medidas de los sensores, σ_{SENSOR}, no se aprecia ninguna dependencia clara con el incremento de la frecuencia de muestreo de la IMU. Las derivas entre las tres frecuencias se encuentran en el orden de 10^{-4} para los acelerómetros y 10^{-3} para los giroscopios.

Con respecto a las desviaciones estándar del modelo de BIAS de los sensores, σ_{BIAS}, se aprecia que existe cierta dependencia directa con la frecuencia de muestreo empleada. Al incrementarse la frecuencia se ve incrementada σ_{BIAS}. Podemos extraer que Bias Instability tiene dependencia directamente proporcional a la frecuencia de muestreo de la IMU.

Con respecto a las desviaciones estándar del modelo ANGULAR/VELOCITY RANDOM WALK de los sensores, σ_{AVRW}, no se aprecia una clara dependencia con el incremento de la frecuencia de muestreo de la IMU. Las derivas entre las tres frecuencias se encuentran en el orden de 10^{-4} para los acelerómetros y 10^{-3} para los giroscopios.

Con respecto a las desviaciones estándar del modelo de ACCELERATION/RATE RANDOM WALK de los sensores, σ_{ARRW}, se aprecia una dependencia directamente proporcional con el incremento de la frecuencia de muestreo de la IMU.

Con respecto a la coincidencia, FIT, de los modelos generados con las medidas de los sensores, no se observa dependencia frecuencial a excepción del acelerómetro del eje Y. El FIT de dicho sensor se ve afectado con el incremento de la frecuencia de muestreo, dependencia inversa, del orden del 0,1% en cada salto frecuencial.

Con respecto al valor de la constante N, ANGULAR/VELOCITY RANDOM WALK, se observa una dependencia inversa con la frecuencia de muestreo. Al incrementar esta, se produce un decremento en su modulo.

Con respecto al valor de la constante B, BIAS INSTABILITY, no se observa una dependencia directa entre el incremento de la frecuencia de muestreo y la variación leve del módulo de la constante, del orden de 10^{-5} para los acelerómetros y 10^{-3} para los giroscopios.

Con respecto al valor de la constante K, RATE RANDOM WALK, no se aprecia una dependencia directa entre el incremento de la frecuencia de muestreo y la variación del módulo de la constante de los acelerómetros. Para los giroscopios no se ha obtenido contribución de este efecto en las frecuencias utilizadas.

Según los datos obtenidos de los test realizados y el análisis posterior mediante el método de modelado de Varianza de Allan, podemos extraer que solo el BIAS INSTABILITY tiene una dependencia clara con la frecuencia de muestreo de la IMU, siendo esta dependencia directamente proporcional. Al incrementar la frecuencia de muestreo utilizada se incrementa el BIAS.

6. LINEAS FUTURAS

Para futuros trabajos con esta IMU, se recomienda la mejora de los modelos de error obtenidos. Aplicando para ello un estudio de la variación y dependencia térmicas de los errores estocásticos que afectan a los sensores de la IMU aprovechando la adaptación de un sensor térmico a la plataforma hardware.

Además, podría ser interesante observar las dependencias de los parámetros extraídos del análisis mediante Varianza de Allan con la frecuencia de muestreo y la temperatura.

7. BIBLIOGRAFIA

[1] - "**Analysis and Modeling of MEMS based Inertial Sensors**", Claudia C. Meruane Naranjo. Signal Processing School of Electrical Engineering Kungliga Tekniska Hgskolan. Stockholm, 2008.

[2] - "**Stochastic Modeling of MEMS Inertial sensors**", Petko Petkov & Tsonyo Slavov. Bulgarian Academy of Sciences, Department of Automatics, Technical University of Sofia. Sofia, 2010.

[3] - "**IEEE Std 952™-1997 (R2008)**", The Institute of Electrical and Electronics Engineers, Inc. New York (USA), Reaffirmed 10 December 2008.

[4] - "**Allan Variance analisis on error characters of MEMS Inertial sensors for an FPGA-based GPS/INS system**", Xin Zhang, Yong Li, Peter Mumford, Chris Rizos. School of Surveying and Spatial Information Systems, University of New South Wales, Australia.

[5] - "**An introduction to inertial navigation**", Olivier J. Woodman, August 2007.

[6] - "**Analysis and modeling of inertial sensors using Allan Variance**", Naser El-Sheimy, Haiying Hou and Xiaoji Niu.

[7] - "**Angle random walk**", Dr. Walter Stockwell. Crossbow Technology, Inc.

[8] - "**Bias stability measurement: Allan Variance**", Dr. Walter Stockwell. Crossbow Technology, Inc.

[9] - "**Calibration and stochastic modeling of inertial navigation sensor errors**", Mohammed El-Diasty and Spiros Pagiatakis. Dept. of Earth & Space Science & Engineering, York University, Canada.

[10] - "**Characterization of various IMU error sources and the effect on navigation performance**", Warren S. Flenniken IV, John H. Wall, David M. Bevly, Auburn University.

[11] - "**Cost-effective testing and calibration of low cost MEMS sensors for integrated positioning, navigation and mapping systems**", Priyanka AGGARWAL, Zainab SYED, Dr. Xiaoji NIU and Dr. Naser EL-SHEIMY, Canada.

[12] - "**Development & Assessment of a low dynamic vehicle navigation system**", Shahram Moafipoor, Lydia Bock, Jeffrey A. Fayman, Gerry Mader, Paul J. de Jonge. Geodetics, Inc., San Diego, CA.

[13] - "**Embedded low cost inertial navigation system**", Kevin J. Walchko, Michael C. Nechyba, Eric Schwartz and Antonio Arroyo. University of Florida, Gainesville.

[14] - "**Error Analysis and Stochastic Modeling of MEMS based Inertial Sensors for Land Vehicle Navigation Applications**", Minha Park, Department of Geomatics Engineering, University of Calgary. April 2004.

[15] - "**Error and performance analysis of MEMS-based inertial sensors with a low-cost GPS receiver**", Minha Park and Yang Gao, 2008.

[16] – "**Estimation techniques for low-cost inertial navigation**", Eun-Hwan Shin, Department of Geomatics Engineering, University of Calgary. May 2005.

[17] - "**Improving the inertial navigation system (INS) error model for INS and INS/DGPS applications**", Sameh Nassar, Department of Geomatics Engineering, University of Calgary. November 2003.

[18] - "**Inertial navigation sensors**", Neil M. Barbour, Charles Stark Draper Laboratory. Cambridge, USA. (O.T.A.N.)

[19] - "**Modeling and bounding low cost inertial sensor errors**", Zhiqiang Xing and Demoz Gebre-Egziabher. University of Minnesota, Twin Cities, Minneapolis, Minnesota.

[20] - "**Modeling and numerical simulation of an algorithm for the inertial sensors errors reduction and for the increase of the strap-down navigator redundancy degree in a low cost architecture**", Lucian T. Grigorie, Ruxandra M. Botez. August 2008.

[21] - "**Modeling inertial sensors errors using Allan Variance**", Haiying Hou, Department of Geomatics Engineering, University of Calgary. September 2004.

[22] - "**Modeling the stochastic drift of a MEMS-based gyroscope in Gyro/Odometer/GPS integrated navigation**", Jacques Georgy (member IEEE), Aboelmagd Noureldin (Senior member IEEE), Michael J. Korenberg, and Mohamed M. Bayoumi. IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 11, NO. 4, DECEMBER 2010

[23] - "**Noise reduction and estimation in multiple micro-electro-mechanical inertial systems**", Adrian Waegli, Jan Skaloud, Stéphane Guerrier, Maria Eulàlia Parés and Ismael Colomina. IOP PUBLISHING, March 2010.

[24] - "**Online estimation of state space error model for MEMS IMU**", Vaibhav Saini, S C Rana, MM Kuber. HyperSciences Publisher, August 2010.

[25] - "**Performance improvement of GPS/INS integrated system using Allan Variance analysis**", Hyunseok Kim, Jang Gyu Lee and Chan Gook Park. Seoul National University, Korea. Presented at GNSS 2004 International Symposium on GNSS/GPS. Sydney, Australia.

[26] - "**Reliability testing procedure for MEMS IMUs applied to vibrating environments**", Giorgio De Pasquale and Aurelio Somà. Department of Mechanics, Politecnico di Torino. Torino, Italy. January 2010.

[27] - "S**tochastic models of decisions about motion direction: Behavior and physiology**", Jochen Ditterich, Center for Neuroscience and Section of Neurobiology, Physiology and Behavior, University of California. Davis, California (USA). May 2006.

[28] - "**Suppression of Correlated Noise**", Jan Aelterman, Bart Goossens, Aleksandra Pizurica & Wilfred Philips. GHENT University, Belgium.

[29] - http://www.allanstime.com/index.html.

[30] - http://damien.douxchamps.net/research/imu/

ANEXOS

a) Allan Variance.[22]

El método de Varianza de Allan fue desarrollado por el físico norteamericano David W. Allan en 1966 para estudiar la estabilidad frecuencial de los osciladores, con el fin de generar una señal horaria internacional de precisión. Se trata de una técnica de análisis de secuencias de datos en el dominio temporal.

Lo atractivo de este sistema de análisis es que al graficar en escalas logarítmicas los resultados obtenidos tras el análisis, se puede diferenciar a simple vista sobre la gráfica de Varianza Allan las diferentes contribuciones de las fuentes de error examinando las variaciones en las pendientes de los tramos de dicha gráfica.

Este método es matemáticamente sencillo de desarrollar e implementar, aunque computacionalmente es tedioso pues requiere de un elevado número de iteraciones en el proceso de cálculo.

Sea la salida instantánea del sensor a analizar Ω(t), la media de cada CLUSTER se define como:

$$\overline{\Omega}_k(\tau) = \frac{1}{\tau}\int_{t_k}^{t_k+\tau} \Omega(t)\,dt \ , \quad \tau = n\tau_0$$

Esta expresión representa la media de un clúster genérico, que empieza en el término "*k*" y contiene "*n*" muestras consecutivas de datos.

La Varianza Allan de longitud "τ" se define entonces como:

$$\sigma^2(\tau) = \frac{1}{2(N-2n)} \sum_{k=1}^{N-2n} [\overline{\Omega}_{k+\tau}(\tau) - \overline{\Omega}_k(\tau)]^2$$

[22] Referencia [1]

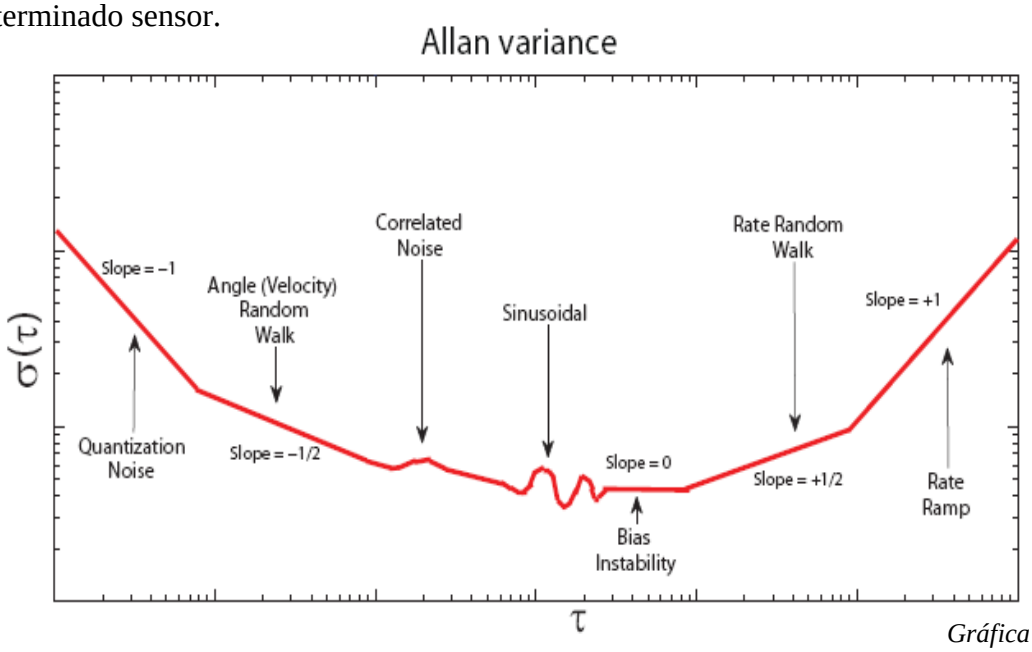
La varianza Allan está relacionada con la Densidad Espectral de Potencia del ruido de los datos originales mediante la expresión:

$$\sigma^2(\tau) = 4\int_0^{+\infty} S_\Omega(f)\frac{\sin^4(\pi f\tau)}{(\pi f\tau)^2}df$$

Esta relación nos indica que la varianza Allan es proporcional a la potencia total de salida del proceso aleatorio, ruido blanco, cuando pasa a través de un filtro cuya función de transferencia genérica es de la forma: $\boldsymbol{sen^2(x)/x^2}$.

Esta particular función de transferencia proviene del método utilizado para crear los clústeres y trabajar con ellos.

De la expresión que relaciona la PSD y la varianza Allan se extrae que la banda de paso del filtro depende de "τ", con lo que variando dicho parámetro podremos examinar diferentes tipos de procesos aleatorios.
La representación gráfica, en ejes logarítmicos, de la raíz cuadrada de la varianza Allan, $\sigma(\tau)$, respecto a "τ" nos proporciona una sencilla manera de identificar y cuantificar los diferentes términos asociados al ruido que existen en los datos de un determinado sensor.



Gráfica 2[23]

[23] Gráfica 2: Resultados genéricos del estudio mediante Allan Variance. Referencia [1]

Se aprecia en la imagen las posibles fuentes de error en las medidas. Cada sensor, en función de su proceso de fabricación, calibración y factores ambientales durante el proceso de obtención de datos, muestra efectos dominantes de algunas de estas fuentes de error.

Análisis de las fuentes de error

A. Ruido de cuantificación:

Como su nombre indica, este efecto se debe a los inherentes errores que se producen durante el proceso de discretización/digitalización de una señal analógica. Pues siempre se aproxima el valor continuo de dicha señal por otro fijo más próximo, definido por la resolución del conversor.

La PSD de este proceso se define como:

$$S_\theta(f) = \begin{cases} \tau_0 Q_z^2 \left(\frac{\sin^2(\pi f \tau_0)}{(\pi f \tau_0)^2}\right) & \\ \approx \tau_0 Q_z^2 & f < \frac{1}{2\tau_0} \end{cases}$$

Q_z: *coeficiente de ruido de cuantificación.*

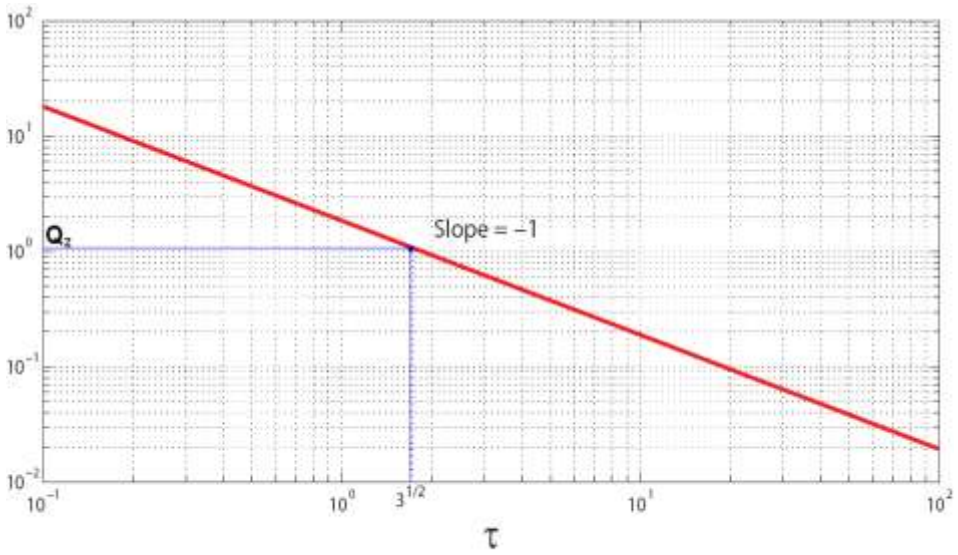
τ_0: *intervalo de muestreo.*

Para pruebas realizadas en condiciones de tiempo de muestreo uniforme y fijo, el límite teórico de Q_z es: $S/\sqrt{12}$ **"*S*":** *factor de escala del sensor*.

De la relación anterior se extrae que la varianza Allan para este error se expresa como:

$$\sigma^2(\tau) = \frac{3Q_z^2}{\tau^2} \quad \text{o} \quad \sigma(\tau) = \sqrt{3}\frac{Q_z}{\tau}$$

El ruido de cuantificación se representa como una recta de pendiente "-1" en un gráfico logarítmico σ(τ) vs "τ". Podemos leer el valor de la constante característica de este error sobre la misma gráfica en: $\tau - \sqrt{3}$



$\sigma(\tau)$ *Plot for quantization noise (after IEEE Std.952-1997)*

Gráfica 3[24]

B. Random Walk (Angular/Velocity):

Error producido por ruido blanco. Término de ruido debido a las altas frecuencias, puede ser observado a la salida como pequeñas variaciones. Su tiempo de correlación es mucho menor que el tiempo de muestreo. Se caracteriza por un espectro de ruido blanco en la salida.

La PSD asociada a este ruido viene definida como:

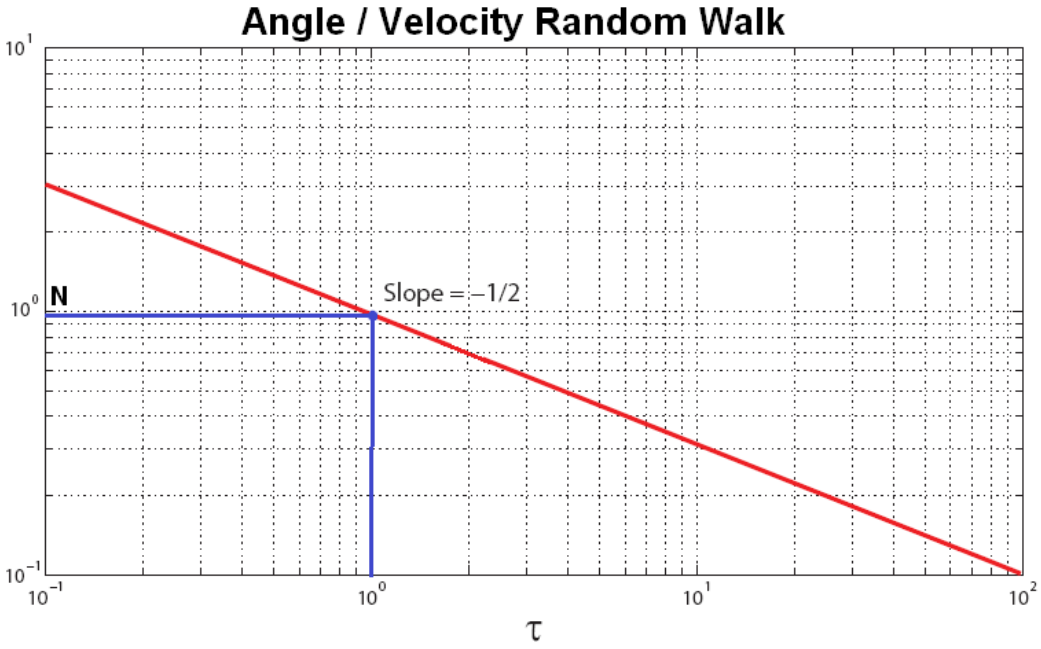
$$S_\Omega(f) = N^2$$ **N**: *coeficiente Angular/Velocity Random Walk*

De la relación anterior se extrae que la varianza Allan para este error se expresa como:

$$\sigma^2(\tau) = \frac{N^2}{\tau} \quad \text{o} \quad \sigma(\tau) = \frac{1}{\sqrt{\tau}} N$$

[24] Gráfica 3: Ruido cuantificación. Referencia [1]

Este ruido se representa como una recta de pendiente "-1/2" en un gráfico logarítmico σ(τ) vs "τ". Podemos leer el valor de la constante característica de este error sobre la misma gráfica en: $\tau = 1$



$\sigma(\tau)$ *Plot for angle (velocity) random walk (after IEEE Std. 952-1997)*

Gráfica 3[25]

C. Inestabilidad Bias:

También conocido como "*flicker noise*", es un ruido de baja frecuencia debido a las fluctuaciones en los datos medidos. Su origen se encuentra en la propia electrónica y en cualquier dispositivo susceptible de generar este tipo de fenómeno. Se trata de un proceso aleatorio estacionario que puede ser considerado como un proceso de Gauss-Markov de bajo orden y media cero.

La PSD asociada a este ruido viene definida como:

$$S_\Omega(f) = \begin{cases} \left(\frac{B^2}{2\pi}\right)\frac{1}{f} & f \le f_0 \\ 0 & f > f_0 \end{cases}$$

B bias instability coefficient
f_0 cutoff frequency

[25] Gráfica 3: Angular/Velocity Random Walk. Referencia [1]

De la relación anterior se extrae que la varianza Allan para este error se expresa como:

$$\sigma^2(T) = \frac{2B^2}{\pi} \times \left[\ln 2 - \frac{\sin^3 x}{2x^2}(\sin x + 4x\cos x) + C_{\mathrm{i}}(2x) - C_{\mathrm{i}}(4x) \right]$$

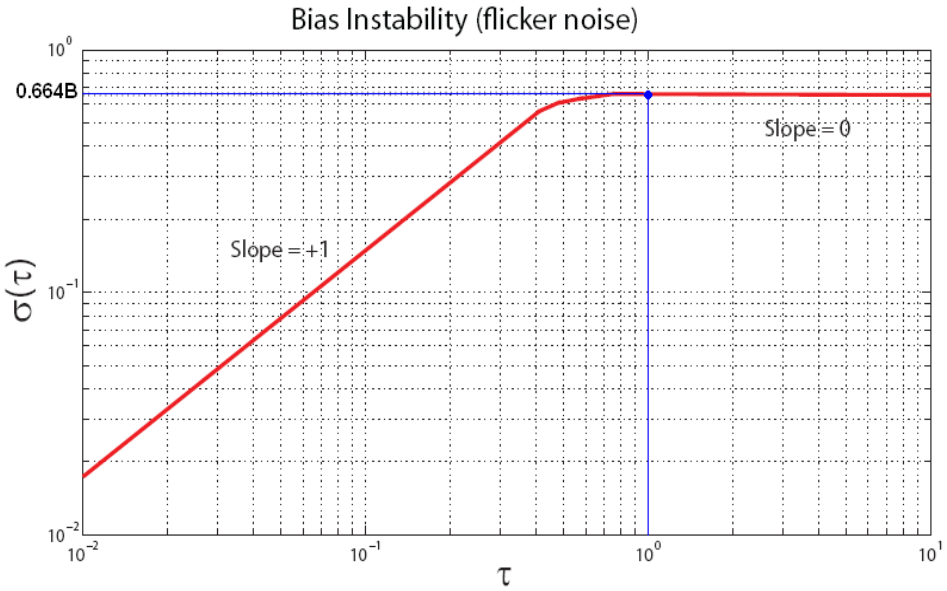
Siendo:

$$x = \pi f_0 \tau \qquad Ci(x) = \int_x^{\infty} \frac{\cos t}{t} dt$$

Analizando con detenimiento la expresión de la varianza Allan podemos aproximar:

$$\sigma^2(\tau) \Rightarrow \frac{2B^2}{\pi} \ln 2 = \sqrt{\frac{2\ln 2}{\pi}} B \approx 0.664B \quad \text{para } \tau >> \frac{1}{f_0}$$

Este ruido se representa como una gráfica de dos tramos. Un primer tramo es una recta de pendiente "***1***" para $\tau << \frac{1}{f_0}$. El segundo tramo que es una recta asintótica horizontal en el valor: ***0.664B***



Plot for bias instability (for $f_0 = 1$)(after IEEE Std.952-1997)

Gráfica 4[26]

[26] Gráfica 4: Bias Instability. Referencia [1]

D. Rate Random Walk:

El origen de este efecto es debido a ruido blanco en la aceleración angular. En ocasiones se le relaciona con un caso límite de ruido correlado exponencial con tiempo de correlación muy grande.

La PSD asociada a este ruido viene definida como:

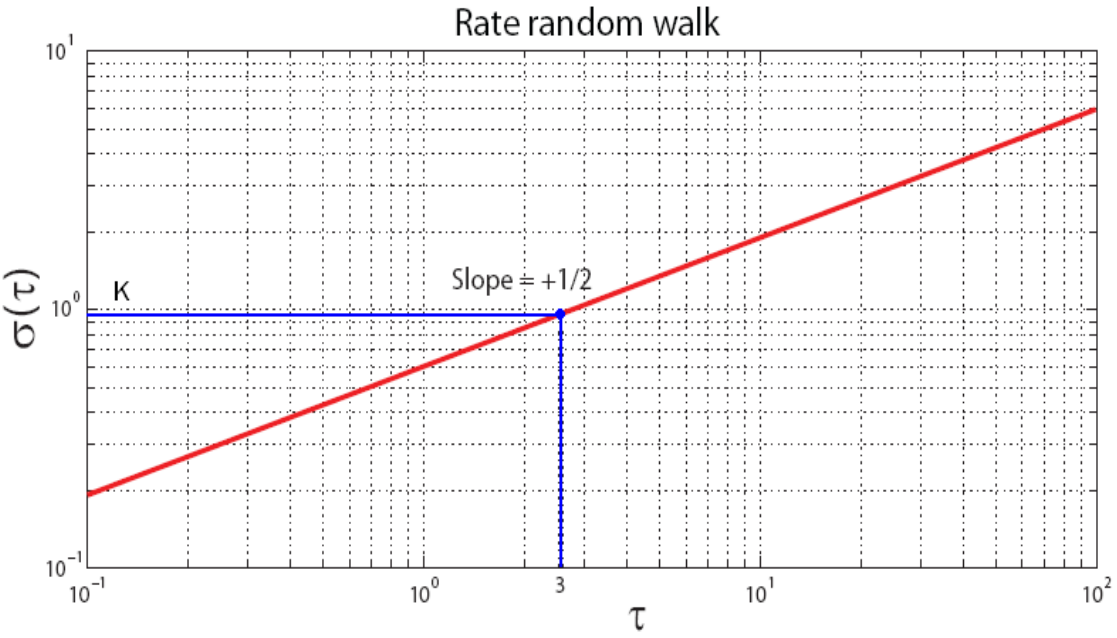
$$S_\Omega(f) = \left(\frac{K}{2\pi}\right)^2 \frac{1}{f^2}$$

K: *coeficiente Rate Random Walk*

De la relación anterior se extrae que la varianza Allan para este error se expresa como:

$$\sigma^2(T) = \frac{K^2 T}{3} \quad \Longrightarrow \quad \sigma(\tau) = \sqrt{\frac{\tau}{3}}\, K$$

Este ruido se representa como una recta de pendiente "1/2" en un gráfico logarítmico σ(τ) vs "τ". Podemos leer el valor de la constante característica de este error en la misma gráfica en: $\tau = 3$



Plot for rate random walk (after IEEE Std.952-1997)

Gráfica 5 [27]

[27] Gráfica 5: Bias Instability. Referencia [1]

E. Rampa de ratio de deriva (Drift Rate Ramp):

Este error pertenece a la categoría de errores determinísticos. Consiste en lentos cambios a la salida, durante largos periodos de tiempo.

Este ruido se define como: $\Omega = Rt$ ***R***: *coeficiente Drift Rate Ramp*

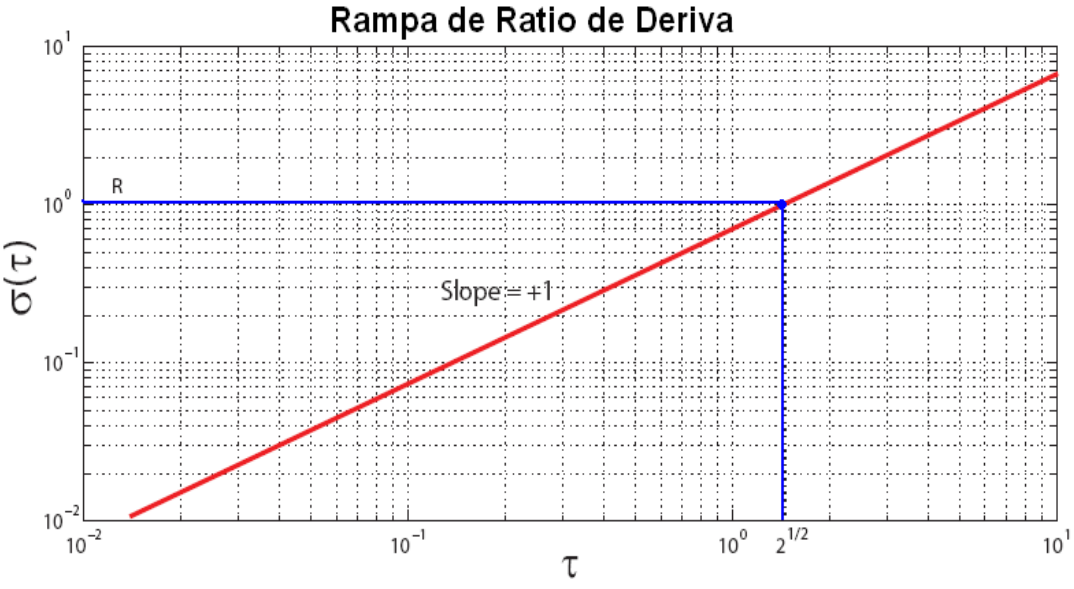
La PSD asociada a este ruido viene definida como:

$$S_\Omega(f) = \frac{R^2}{(2\pi f)^3}$$

De la relación anterior se extrae que la varianza Allan para este error se expresa como:

$$\sigma^2(\tau) = \frac{R^2\tau^2}{2} \Longrightarrow \sigma(\tau) = \frac{\tau}{\sqrt{2}}R$$

Este ruido se representa como una recta de pendiente "1" en un gráfico logarítmico σ(τ) vs "τ". Podemos leer el valor de la constante característica de este error sobre la misma gráfica en el instante de tiempo: $\tau = \sqrt{2}$



$\sigma(\tau)$ Plot for drift rate ramp (after IEEE Std.952-1997)

Gráfica 6 [28]

[28] Gráfica 6: Drift Rate. Referencia [1]

F. Ruido correlado (Correlated Noise)

Aquellos procesos en los que los datos se ven corrompidos, a menudo no pueden ser descritos usando un modelo de ruido blanco aditivo gaussiano simple. Sin embargo muchos de estos procesos se pueden modelar como un proceso de filtrado lineal de una fuente de ruido blanco gaussiano, que resulta en ruido correlacionado.[Ref. 3]

Este ruido está caracterizado por una función exponencial decreciente con tiempo de correlación finito:

$$\left[1 - \frac{T_c}{2\tau}\left(3 - 4e^{-\frac{\tau}{T_c}} + e^{-\frac{2\tau}{T_c}}\right)\right]$$

La PSD asociada a este ruido viene definida como:

$$S_\Omega(f) = \frac{(q_c T_c)^2}{1 + (2\pi f T_c)^2}$$

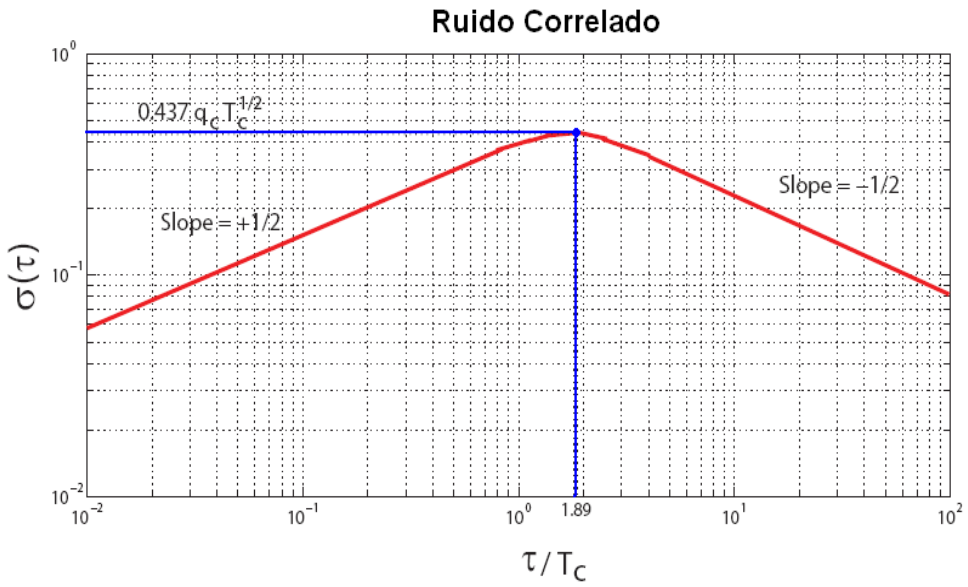
***q*$_c$**: *amplitud ruido*
***T*$_c$**: *tiempo correlación*

De la relación anterior se extrae que la varianza Allan para este error se expresa como:

$$\sigma^2(\tau) \begin{cases} \dfrac{(q_c T_c)^2}{\tau} & \text{para } \tau >> T_c \\ \dfrac{q_c^2}{3}\tau & \text{para } \tau << T_c \end{cases}$$

Este ruido se representa como una gráfica de dos tramos.

Un primer tramo es una recta de pendiente "***1/2***" para $\tau < \tau / T_C$, un segundo tramo que es una recta de pendiente "***-1/2***" para $\tau > \tau / T_C$ y un máximo en: $\tau = \tau / T_C$



$\sigma(\tau)$ Plot for correlated noise (after IEEE Std.952-1997)

Gráfica 7 [29]

G. Ruido sinusoidal:

Este efecto tiene su origen en la contribución de una o varias frecuencias sobre la señal a estudiar.

Asumiendo el mejor de los casos, solo depende de una única frecuencia, podemos expresar su PSD de la siguiente manera:

$$S_\Omega(f) = \frac{1}{2}\Omega_0^2\left[\delta(f - f_0) + \delta(f + f_0)\right]$$

Ω_0 : *amplitud*
f_0 : *frecuencia*
$\delta(x)$: *Delta de Dirac*

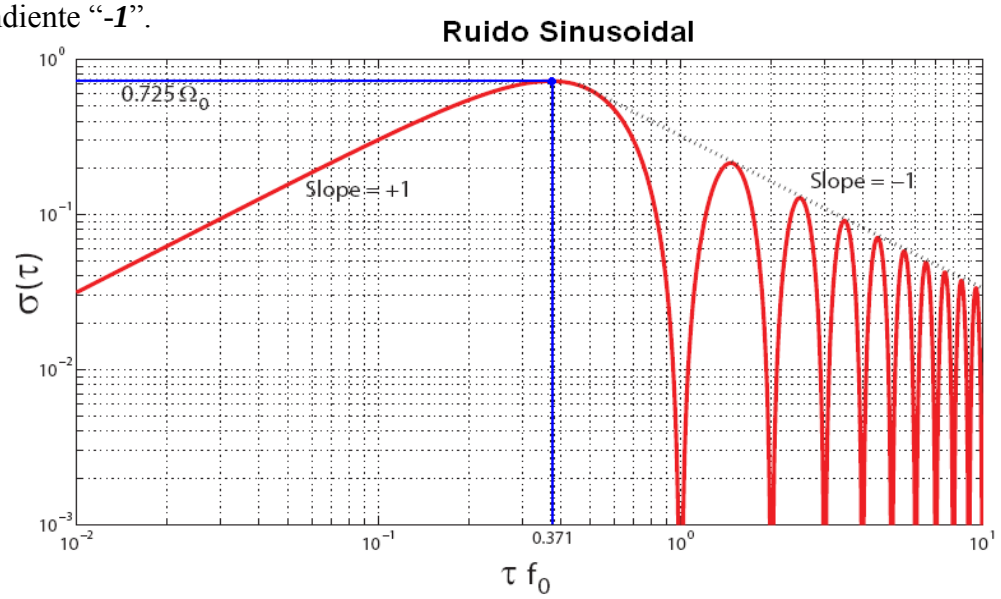
Para el caso de tener contribución de varias frecuencias, este error se representaría como la suma de las contribuciones de cada par amplitud/frecuencia.

[29] Gráfica 7: Ruido Correlado. Referencia [1]

De la relación anterior se extrae que la varianza Allan para este error se expresa como:

$$\sigma^2(\tau) = \Omega_0 \left(\frac{\sin^2(\pi f_0 \tau)}{\pi f_0 \tau} \right)^2$$

Un primer tramo es una recta de pendiente "***1***" hasta alcanzar un máximo en $t = \tau f_0$
A partir de este valor, el segundo tramo presenta picos atenuados por una recta de pendiente "***-1***".



$\sigma(\tau)$ Plot for sinusoidal noise (after IEEE Std.952-1997)

Gráfica 8 [30]

[30] Gráfica 8: Ruido Correlado. Referencia [1]

b) Comparativas.

A continuación se presentan unas tablas comparativas de los sensores que incorpora la IMU con otros modelos, de características similares, disponibles en el mercado y otras IMU.

ACELEROMETROS

FABRICANTE Y MODELO	MODELO	CARACTERISTICAS	PRECIO
Analog Devices	ADXL 325	- Triaxial - Consumo: 350uA (Typ) - Alimentación: 1,8V-3,6V - Rango: ±5g - Sensibilidad(V_s = 3V): 174mV/g (Typ) - No Lineal: ±0,2% Full Scale - Rango térmico: [-40,+85]ºC	2,38$
	ADXL 335	- Triaxial - Consumo: 350uA (Typ) - Alimentación: 1,8V-3,6V - Rango: ±3g - Sensibilidad(V_s = 3V): 300mV/g (Typ) - No Lineal: ±0,3% Full Scale - Rango térmico: [-40,+85]ºC	15$
	ADXL 337	- Triaxial - Consumo: 300uA (Typ) - Alimentación: 1,8V-3,6V - Rango: ±3g - Sensibilidad(V_s = 3V): 300mV/g (Typ) - No Lineal: ±0,3% Full Scale - Rango térmico: [-40,+85]ºC	--

Bosch	BMA 145	- Triaxial - Consumo: 200uA (Typ) - Alimentación: 1,8V-3,5V - Rango: ±4g - Sensibilidad: (V_{DD}/10)±4%[V/g](Typ) - No Lineal: ±0,5% Full Scale - Rango térmico: [-40,+85]ºC	--
Freescale Semiconductor	MMA7261QT	- Triaxial - Consumo: 500uA (Typ) - Alimentación: 2,2V-3,6V - Rango: ±2,5g/±3,3g/±6,7g/±10g - Resolución: 3,91mg - Deriva Sens(Temp): ±0,02%/ºC (Typ) - No Lineal: ±0,5% Full Scale - Rango térmico: [-40,+105]ºC	
	MMA7331LC	- Triaxial - Consumo: 400uA (Typ) - Alimentación: 2,2V-3,6V - Rango: ±4g/±9g - Deriva Sens(Temp): ±0,002%/ºC (Typ) - No Lineal: ±1% Full Scale - Rango térmico: [-40,+85]ºC	
	MMA7341LC	- Triaxial - Consumo: 400uA (Typ) - Alimentación: 2,2V-3,6V - Rango: ±3g/±9g - Deriva Sens(Temp): ±0,002%/ºC (Typ) - No Lineal: ±1% Full Scale - Rango térmico: [-40,+85]ºC	

	MMA7361LC	- Triaxial - Consumo: 400uA (Typ) - Alimentación: 2,2V-3,6V - Rango: ±1,5g/±6g - Deriva Sens(Temp): ±0,002%/ºC (Typ) - No Lineal: ±1% Full Scale - Rango térmico: [-40,+85]ºC	

GIROSCOPIOS

FABRICANTE Y MODELO	MODELO	CARACTERISTICAS	PRECIO
Analog Devices	ADXRS 300	- Mono axial - Hasta 2000g - Alimentación: 5V - Consumo: 6mA(Typ) - Ratio sensado: ±300º/s (Min) - Sensibilidad(@25ºC): 5mV/º/s (Typ) - No Lineal: ±0.1% Full Scale - Rango térmico: [-40,+85]ºC	--
	ADXRS 610/620	- Mono axial - Hasta 2000g - Alimentación: 4,75V-5,25V - Consumo: 3,5mA(Typ) - Ratio sensado: ±300º/s (Min) - Sensibilidad: 6mV/º/s (Typ) ±2% - No Lineal: ±0.1% Full Scale - Rango térmico: [-40,+105]ºC	67,01€
	ADXRS 646	- Mono axial - Hasta 10.000g - Alimentación: 5,75V-6,25V - Consumo: 4mA(Typ) - Ratio sensado: ±300º/s (Typ) ±3% - Sensibilidad: 9mV/º/s (Typ) ±3% - No Lineal: ±0.01% Full Scale - Rango térmico: [-40,+105]ºC	--
Invensense	ISZ-500/505	- Mono axial - Hasta 10.000g - Alimentación: 2,7V-3,3V - Consumo: 4,5mA(Typ) - Ratio sensado A: ±500º/s (Typ) - Sensibilidad A: 2mV/º/s (Typ) - Ratio sensado B: ±110º/s (Typ) - Sensibilidad B: 9,1mV/º/s (Typ) - Tolerancia: ±6% - No Lineal: ±0.5% Full Scale - Rango térmico: [-20,+85]ºC	9,55€

Invensense	ISZ-650/655	- Mono axial - Hasta 10.000g - Alimentación: 2,7V-3,3V - Consumo: 4,5mA(Typ) - Ratio sensado A: ±2000º/s (Typ) - Sensibilidad A: 0,5mV/º/s (Typ) - Ratio sensado B: ±440º/s (Typ) - Sensibilidad B: 2,7mV/º/s (Typ) - Tolerancia: ±6% - No Lineal: ±0.5% Full Scale - Rango térmico: [-20,+85]ºC	9,55€
ST Microelectronics	LY330ALH	- Mono axial - Alimentación: 2,7V-3,6V - Consumo: 4,2mA (Max) - Ratio sensado : ±300º/s - Sensibilidad: 3,752mV/(º/s) ± 0,01%/ºC - No Lineal: ±1% Full Scale - Rango térmico: [-40,+85]ºC	8,77€
	LY530ALH	- Mono axial - Alimentación: 2,7V-3,6V - Consumo: 5mA (Typ) - Ratio sensado : ±300(º/s)/ ±1200(º/s) - Sensibilidad: (3,3 / 0,83)[mV/(º/s)] ± 0,05%/ºC - No Lineal: ±1% Full Scale - Rango térmico: [-40,+85]ºC	--

IMU's[31]

FABRICANTE Y MODELO	MODELO	CARACTERISTICAS	PRECIO
CH ROBOTICS	CHR-6dm	- Alimentación: [3.3-12]V - Consumo: 58mA (Max) - 3xGyros: *±400º/s@2.5mV/(º/s)±0.03%/(º/s)* - 3xAccel: *±3.6g@300mV/g±0.01%/ºC±0.03%FS* - Magnetómetro incorporado - EKF (Extended Kalman Filter) - Auto calibración BIAS de Gyros - UART @ 115200Baud - 2xUART + SPI bus - Rango térmico: [-40,+85]ºC	124$
	UM6-LT	- Alimentación: [3.5-5]V - Consumo: 58mA (Max) - 3xGyros: *±2000º/s@14.375LSB/(º/s)±10%±0.2%FS(No Lineal)* - 3xAccel: *±2g@1mV/LSB±0.01%/ºC* - Magnetómetro incorporado - Auto calibración BIAS de Gyros - Termo compensación Gyro Rate - UART@115200Baud + SPI bus - Rango térmico: [-40,+85]ºC	100$

[31] http://damien.douxchamps.net/research/imu/

Invensense	IMU-3000A	- Alimentación: [2.1-3.6]V - Consumo: 58mA (Max) - ADC 16bits - 3xGyros: *±250[º/s] / ±500[º/s] / ±1000[º/s] / ±2000[º/s]* *131LSB/(º/s) / 65.5 LSB/(º/s) / 32.8 LSB/(º/s) / 16.4 LSB/(º/s)* *Error Sens: ±3%(Tolerance)± 2%(Temp)±0.2%(No Lineal)* - Permite acelerómetros externos (I^2C) - Sensor térmico interno - 2x I^2C interface - Rango térmico: [-40,+85]ºC	--
	MPU-6000/6050	- Alimentación: [2.375-3.46]V - Consumo: 3.9mA (Max) - 6xADC 16bits - 3xGyros: *±250[º/s] / ±500[º/s] / ±1000[º/s] / ±2000[º/s]* *131LSB/(º/s) / 65.5 LSB/(º/s) / 32.8 LSB/(º/s) / 16.4 LSB/(º/s)* *Error Sens: ±3%(Tolerance)± 2%(Temp)±0.2%(No Lineal)* - 3xAccel: *±2g / ±4g / ±8g / ±16g* *16.384LSB/g / 8.192 LSB/g / 4.096 LSB/g / 2.048 LSB/g* *Error Sens: ±3%(Tolerance)± 0.02%(Temp)±0.5%(No Lineal)* - Sensor térmico interno - I^2C + SPI interface - UART@115200Baud + SPI bus - Rango térmico: [-40,+85]ºC	15$
Ryan Mechatronics	CHIMU	- Alimentación: [3.1-6.5]V - Consumo: 36mA (Max) - ADC 12bits - 3xGyros: *±2000[º/s] ±0.5[º] ±0.07[º/s]* - 3xAccel: *±2g / ±4g / ±8g±* *±1mg / ±2mg / ±4mg* - Sensor térmico interno - UART@115200Baud + SPI bus - Rango térmico: [-30,+85]ºC	200$

ST Microelectronics	LSM320HAY30	- Alimentación: [2.7-3.6]V - Consumo: 12mA (Max) - ADC 16bits - 3xGyros: *±300[º/s] / ±3.33mV/(º/s)±0.07%/ºC±1%FS* *±1200[º/s] / ±0.83mV/(º/s)±0.07%/ºC±1%FS* - 3xAccel: *±2g / ±4g / ±8g±* *±1mg/(digit) / ±2mg/(digit) / ±3.9mg/(digit)* *Sensibilidad(Temp): ±0.01%/ºC* - Sensor térmico interno - I^2C+ SPI bus - Rango térmico: [-40,+85]ºC	--
x-io Technologies	x-IMU	- Alimentación: [3.6-6.3]V - Consumo: 50mA-150mA - 3xGyros: ADC 16bits *±250[º/s] / ±500[º/s] / ±1000[º/s] / ±2000[º/s]* *131LSB/(º/s) / 65.5 LSB/(º/s) / 32.8 LSB/(º/s) / 16.4 LSB/(º/s)* *Error Sens: ±3%(Tolerance)± 2%(Temp)±0.2%(No Lineal)* - 3xAccel: ADC 12bits *±2g / ±4g / ±8g±* *±1mg/(digit) / ±2mg/(digit) / ±3.9mg/(digit)* *Sensibilidad(Temp): ±0.01%/ºC* - 3xMagnetometro: ADC 12bits *±1.3G /±1.9G / ±2.5G /±4G /±4.7G /±5.6G /±8.1G /* *Resolución: 8mGaus* - Termo compensación Gyro - Termómetro (ADC 16bits) - USB + Bluetooth - Micro SD card - UART: [2.4-921.6]Kbaud - I^2C+ SPI bus - Rango térmico: [-30,+85]ºC	250£

c) Códigos fuente.

```
% ================================================================
% ==================  Obtención y Conversión de datos  ===================
% ================================================================

close all
clear all

fd = fopen('TestXX.txt','r');              % Abre "Test.txt" con derechos de lectura
DATA = fscanf(fd,'%d',[6,3.24e6]);         % Leer los datos del fichero en filas
fclose(fd);                                % Cerrar el fichero

% ========================  Vectores de datos  =========================


Max = DATA(1,:);        % Vector fila muestras en eje X
May = DATA(2,:);        % Vector fila muestras en eje Y
Maz = DATA(3,:);        % Vector fila muestras en eje Z
Mwr = DATA(4,:);        % Vector fila muestras Roll
Mwp = DATA(5,:);        % Vector fila muestras Pitch
Mwy = DATA(6,:);        % Vector fila muestras Yaw

Termo = DATA(7,:);      % Vector fila muestras Temperatura
LT = length(Termo);     % Longitud del vector Termo

% ================================================================


% ============================  CONSTANTES  ============================

Vref = 3.3;             % Voltaje de referencia de conversión AD [V]
dt = 1/xxx;             % Periodo de muestreo [s] TEST-XX
g = 9.80665;            % Gravedad estándar, valor promedio [m/s^2]
w_s = 3.3e-3;           % Sensibilidad giroscopios [mV/º/s]
T_s = 10e-3;            % Sensibilidad termo-sensor 10mV/º
T_Data = (1:LT).*dt;    % Tiempo [s] total del vector de datos

% ================================================================
```

```
% ============================  TEMPERATURA  ========================

% Termo: lectura térmica

Temp = ((Termo.*Vref)/1024)./T_s;       % Temperatura [ºC]
sd_Temp = std(Temp);                    % Desviación estándar de 'Temp'


% ------------------------------------------  Filtrado Temperatura  ------------------------------------------

A = 1;

for m = 0:max(T_Data)-1          % Recorrer por segundos

        B = (m+1)*(1/dt);
        TF = Temp(A:B);
        Temp_filt(m+1) = mean(TF);
        A = A + (1/dt);

end

figure(1);
plot(T_Data,Temp,'r'), grid on, title('TEMPERATURA AMBIENTE'), hold on;
plot(Temp_filt,'g'), legend('Temperatura Sensor','Temperatura Filtrada'),xlabel('Tiempo
[s]'),ylabel('Temperatura [º Centígrados]');

% ======================================================================


% ========================  VELOCIDAD LINEAL(V)  ========================

% Max: Lectura aceleración en X
% May: Lectura aceleración en Y
% Maz: Lectura aceleración en Z

% --------------------------------  Selección sensibilidad acelerómetros  --------------------------------

        ac_s = 800e-3;          % 1,5g -> 800mV/g
        % ac_s = 600e-3;        % 2,0g -> 600mV/g
```

```
% ac_s = 300e-3;        % 4,0g -> 300mV/g
% ac_s = 200e-3;        % 6,0g -> 200mV/g

% ------------------------------------ Cálculo de la velocidad ----------------------------------------------

Max1 = (Max.*Vref/1024)-(Vref/2);      % Referencia sobre Vref/2
ax = (Max1-mean(Max1))./ac_s;          % Centra medidas en CERO y convierte a m/s2 [g]
sd_ax = std(ax);                       % Desviación estándar de 'ax'

May1 = (May.*Vref/1024)-(Vref/2);      % Referencia sobre Vref/2
ay = (May1-mean(May1))./ac_s;          % Centra medidas en CERO y convierte a m/s2
sd_ay = std(ay);                       % Desviación estándar de 'ay'

Maz1 = (Maz.*Vref/1024)-(Vref/2);      % Referencia sobre Vref/2
az = (Maz1-mean(Maz1))./ac_s;          % Centra medidas en UNO y convierte a m/s2
sd_az = std(az);                       % Desviación estándar de 'az'


n = 1;

Vx = zeros(n,LT);      % Matriz (n)x(T_Data) de ceros Velocidad lineal en X
Vy = zeros(n,LT);      % Matriz (n)x(T_Data) de ceros Velocidad lineal en Y
Vz = zeros(n,LT);      % Matriz (n)x(T_Data) de ceros Velocidad lineal en Z

Vx(1) = ax(1)*dt;      % Integra 1er término de los datos de Velocidad en X
Vy(1) = ay(1)*dt;      % Integra 1er término de los datos de Velocidad en Y
Vz(1) = az(1)*dt;      % Integra 1er término de los datos de Velocidad en Z

for t = 2:LT       % Integramos las salidas de los sensores ([m/s2]=>[m/s])

    Vx(t) = Vx(t-1) + ax(t)*dt;        % Integra salida sensor acel => vel
    Vy(t) = Vy(t-1) + ay(t)*dt;        % Integra salida sensor acel => vel
    Vz(t) = Vz(t-1) + az(t)*dt;        % Integra salida sensor acel => vel

end;

% ================================================================
```

```
% ========================== ANGULO(PHI) ==========================

% Mwr (ROLL): ángulo inclinación lateral, rotación o vadeo. ( Eje X )
% Mwp (PITCH): ángulo ataque, dirección vertical de avance. ( Eje Y )
% Mwy (YAW): ángulo dirección horizontal de avance. ( Eje Z )

% ------------------------------------------- Cálculo del ángulo -------------------------------------------

Mwr1 = (Mwr.*Vref/1024)-(Vref/2);      % Referencia sobre Vref/2
wr = (Mwr1-mean(Mwr1))./w_s;           % Centra medidas en CERO y convierte a [º/s]
sd_wr = std(wr);                       % Desviación estándar de 'wr'

Mwp1 = ((Mwp.*Vref/1024)-(Vref/2));    % Referencia sobre Vref/2
wp = (Mwp1-mean(Mwp1))./w_s;           % Centra medidas en CERO y convierte a [º/s]
sd_wp = std(wp);                       % Desviación estándar de 'wp'

Mwy1 = ((Mwy.*Vref/1024)-(Vref/2));    % Referencia sobre Vref/2
wy = (Mwy1-mean(Mwy1))./w_s;           % Centra medidas en CERO y convierte a [º/s]
sd_wy = std(wy);                       % Desviación estándar de 'wy'



n = 1;

Wr = zeros(n,LT);      % Matriz (n)x(T_Data) de ceros Velocidad angular en X
Wp = zeros(n,LT);      % Matriz (n)x(T_Data) de ceros Velocidad angular en Y
Wy = zeros(n,LT);      % Matriz (n)x(T_Data) de ceros Velocidad angular en Z

Wr(1) = wr(1)*dt;      % Integra 1er término de los datos de Velocidad en X
Wp(1) = wp(1)*dt;      % Integra 1er término de los datos de Velocidad en Y
Wy(1) = wy(1)*dt;      % Integra 1er término de los datos de Velocidad en Z

for t = 2:LT       % Integramos las salidas de los sensores ([º/s]=>[º])

        Wr(t) = Wr(t-1) + wr(t)*dt;      % Integra salida sensor vel => posición
        Wp(t) = Wp(t-1) + wp(t)*dt;      % Integra salida sensor vel => posición
        Wy(t) = Wy(t-1) + wy(t)*dt;      % Integra salida sensor vel => posición

end;
```

```
% ------------------------------  Guardado Medidas / Integral Sensores  ----------------------------------

t = [dt,LT];                                                    % Genera vector “t”
dlmwrite('t.txt', t, 'delimiter', '\n', 'precision', '%.4f');   % Exporta vector “t” a fichero TXT

Datos = [ax; ay; az; wr; wp; wy]';                                      % Genera matriz “Datos”
dlmwrite('Datos.txt', Datos, 'delimiter', '\t', 'precision', '% +.5e'); % Exporta “Datos” a TXT

% Exporta vector ''Temp_filt'' a fichero TXT
dlmwrite('Temp_filt.txt', Temp_filt, 'delimiter', '\n', 'precision', '% 2.5f');

% Exporta vector “T_Data” a fichero TXT
dlmwrite('T_Data.txt', T_Data, 'delimiter', '\n', 'precision', '% .5e');

% Genera vector “desv_est” y exporta a TXT
desv_est = [sd_ax; sd_ay; sd_az; sd_wr; sd_wp; sd_wy; sd_Temp]';
dlmwrite('desv_est.txt', desv_est, 'delimiter', '\n', 'precision', '% +.5f');

V_W = [Vx; Vy; Vz; Wr; Wp; Wy]';                                % Genera matriz “V_W”
dlmwrite('V_W.txt', V_W, 'delimiter', '\t', 'precision', '% +.5e');   % Exporta “V_W” a TXT

% ================================================================================
```

```
% ================================================================
% ==================  Análisis de las fuentes de Ruido  ==========
% ================                                  ==============
% ================  METODO DE LA VARIANZA DE ALLAN  ==============
% ================================================================

close all
clear all

t = dlmread('t.txt', '\n');              % Importa "dt" y "LT" desde fichero TXT
dt = t(1);
LT = t(2);

T_Data = dlmread('T_Data.txt', '\n');    % Importa “T_Data” desde fichero TXT

Datos = dlmread('Datos.txt', '\t');      % Importa “Datos” desde fichero TXT

ax = Datos(:,1);
ay = Datos(:,2);
az = Datos(:,3);
wr = Datos(:,4);
wp = Datos(:,5);
wy = Datos(:,6);

V_W = dlmread('V_W.txt', '\t');          % Importa “V_W” desde fichero TXT

Vx = V_W(:,1);
Vy = V_W(:,2);
Vz = V_W(:,3);
Wr = V_W(:,4);
Wp = V_W(:,5);
Wy = V_W(:,6);


% ----------------------------------------  Medidas Sensores  ----------------------------------------

figure(3)        % Gráfica triple de los datos de los acelerómetros.
subplot(3,1,1);
plot(T_Data,ax,'R'); grid on, legend('Acelerómetro X'), xlabel('Tiempo [s]'), ylabel('Aceleración
lineal [m/s2]');
```

```
subplot(3,1,2);
plot(T_Data,ay,'G'); grid on, legend('Acelerómetro Y'), xlabel('Tiempo [s]'), ylabel('Aceleración
lineal [m/s2]');

subplot(3,1,3);
plot(T_Data,az,'B'); grid on, legend('Acelerómetro Z'), xlabel('Tiempo [s]'), ylabel('Aceleración
lineal [m/s2]');


figure(4)          % Gráfica triple de los datos de los giroscopios.
subplot(3,1,1);
plot(T_Data,wr,'R'); grid on, legend('Giroscopio X'), xlabel('Tiempo [s]'), ylabel('Velocidad
angular [º/s]');

subplot(3,1,2);
plot(T_Data,wp,'G'); grid on, legend('Giroscopio Y'), xlabel('Tiempo [s]'), ylabel('Velocidad
angular [º/s]');

subplot(3,1,3);
plot(T_Data,wy,'B'); grid on, legend('Giroscopio Z'), xlabel('Tiempo [s]'), ylabel('Velocidad
angular [º/s]');


% ==================  Densidad Espectral de Potencia  ==================

[psd_ax F] = pwelch(ax,[],[],[],1/dt);       % Cálculo de la PSD de los acelerómetros.
[psd_ay F] = pwelch(ay,[],[],[],1/dt);
[psd_az F] = pwelch(az,[],[],[],1/dt);

[psd_wr F] = pwelch(wr,[],[],[],1/dt);       % Cálculo de la PSD de los giroscopios.
[psd_wp F] = pwelch(wp,[],[],[],1/dt);
[psd_wy F] = pwelch(wy,[],[],[],1/dt);


figure(5)          % Gráfica triple de la PSD de los acelerómetros.
subplot(3,1,1);
loglog(F,psd_ax,'R'); grid on, legend('PSD Acelerómetro X'), xlabel('Frecuencia [Hz]'),
ylabel('Potencia [g^2]');
```

```
subplot(3,1,2);
loglog(F,psd_ay,'G'); grid on, legend('PSD Acelerómetro Y'), xlabel('Frecuencia [Hz]'),
ylabel('Potencia [g^2]');

subplot(3,1,3);
loglog(F,psd_az,'B'); grid on, legend('PSD Acelerómetro Z'), xlabel('Frecuencia [Hz]'),
ylabel('Potencia [g^2]');


figure(6)          % Gráfica triple de la PSD de los giroscopios.
subplot(3,1,1);
loglog(F,psd_wr,'R'); grid on, legend('PSD Giroscopio X'), xlabel('Frecuencia [Hz]'),
ylabel('Potencia [(º/s)^2]');

subplot(3,1,2);
loglog(F,psd_wp,'G'); grid on, legend('PSD Giroscopio Y'), xlabel('Frecuencia [Hz]'),
ylabel('Potencia [(º/s)^2]');

subplot(3,1,3);
loglog(F,psd_wy,'B'); grid on, legend('PSD Giroscopio Z'), xlabel('Frecuencia [Hz]'),
ylabel('Potencia [(º/s)^2]');

psd_ax = psd_ax';        % Transpone los vectores de datos PSD de los acelerómetros
psd_ay = psd_ay';
psd_az = psd_az';

psd_wr = psd_wr';        % Transpone los vectores de datos PSD de los giroscopios
psd_wp = psd_wp';
psd_wy = psd_wy';
```

```
% ==================  Modelado errores ALLAN VARIANCE ==================

% LT: Longitud temporal de los vectores de datos

% ---------------------- Comparando términos de vectores de MEDIAS ------------------------------

for j = 0:fix(log2(LT/2))            % FIX: redondea a entero menor

        n = 2^j;                     % Clústeres de 'n' elementos en potencias de 2
        tau(j+1) = n*dt;             % Tau: duración de cada clúster de datos
        CTE(j+1) = (0.5/((LT-2*n)*(tau(j+1)^2)));          % Cte que acompaña al sumatorio

        e1 = (LT/n)-1;
        E_AV(j+1) = (1/(sqrt(2*e1)))*100;          % ERROR Varianza Allan

        VX = 0;          % Reinicia el valor del sumatorio de VX
        VY = 0;          % Reinicia el valor del sumatorio de VY
        VZ = 0;          % Reinicia el valor del sumatorio de VZ

        WR = 0;          % Reinicia el valor del sumatorio de WR
        WP = 0;          % Reinicia el valor del sumatorio de WP
        WY = 0;          % Reinicia el valor del sumatorio de WY

        for i=1:LT-2*n           % Cálculo del sumatorio

                VX = VX + (Vx(i+2*n) - 2*Vx(i+n) + Vx(i))^2;
                VY = VY + (Vy(i+2*n) - 2*Vy(i+n) + Vy(i))^2;
                VZ = VZ + (Vz(i+2*n) - 2*Vz(i+n) + Vz(i))^2;

                WR = WR + (Wr(i+2*n) - 2*Wr(i+n) + Wr(i))^2;
                WP = WP + (Wp(i+2*n) - 2*Wp(i+n) + Wp(i))^2;
                WY = WY + (Wy(i+2*n) - 2*Wy(i+n) + Wy(i))^2;

        end;                     % FIN cálculo del sumatorio


        Vvx(j+1) = VX;           % Guarda resultado del sumatorio de Vvx(j) [(m/s)^2]
        Vvy(j+1) = VY;           % Guarda resultado del sumatorio de Vvy(j) [(m/s)^2]
        Vvz(j+1) = VZ;           % Guarda resultado del sumatorio de Vvz(j) [(m/s)^2]
```

```
        Vwr(j+1) = WR;          % Guarda resultado del sumatorio de Vwr(j) [º]
        Vwp(j+1) = WP;          % Guarda resultado del sumatorio de Vwp(j) [º]
        Vwy(j+1) = WY;          % Guarda resultado del sumatorio de Vwy(j) [º]

end;


% ------------------------------  Varianza Allan y Desviación Estándar  ------------------------------

AV_vx = (CTE.*Vvx);             % Varianza ALLAN acelerómetro en X
SD_vx = (AV_vx).^(1/2);         % Desviación estándar acelerómetro en X

AV_vy = CTE.*Vvy;               % Varianza ALLAN acelerómetro en Y
SD_vy = (AV_vy).^(1/2);         % Desviación estándar acelerómetro en Y

AV_vz = CTE.*Vvz;               % Varianza ALLAN acelerómetro en Z
SD_vz = (AV_vz).^(1/2);         % Desviación estándar acelerómetro en Z

AV_roll = CTE.*Vwr;             % Varianza ALLAN del ROLL
SD_roll = (AV_roll).^(1/2);     % Desviación estándar del ROLL

AV_pitch = CTE.*Vwp;            % Varianza ALLAN del PITCH
SD_pitch = (AV_pitch).^(1/2);   % Desviación estándar del PITCH

AV_yaw = CTE.*Vwy;              % Varianza ALLAN del YAW
SD_yaw = (AV_yaw).^(1/2);       % Desviación estándar del YAW


figure(7)                       % Gráfica triple de la SD ALLAN de los acelerómetros.
loglog(tau,SD_vx,'R'), grid on, hold on, title('ALLAN VARIANCE ACELEROMETROS');
loglog(tau,SD_vy,'G');
loglog(tau,SD_vz,'B'); legend('A.V. Acelerómetro X','A.V. Acelerómetro Y','A.V. Acelerómetro
Z'), xlabel('Tau [s] ( Duración Clúster )'), ylabel('Desviación Estándar Allan [g]');

figure(8)                       % Gráfica triple de la SD ALLAN de los giroscopios.
loglog(tau,SD_roll,'R'); grid on, hold on, title('ALLAN VARIANCE GIROSCOPIOS');
loglog(tau,SD_pitch,'G');
loglog(tau,SD_yaw,'B'); legend('A.V. Giroscopio X','A.V. Giroscopio Y','A.V. Giroscopio Z'),
xlabel('Tau [s] ( Duración Clúster )'), ylabel('Desviación Estándar Allan [º/s]');
```

```
% ------------------------------------  ERROR Varianza Allan  ----------------------------------------

% E_AV = 1/( sqrt( 2( (LT/n)-1 ) ) )

figure(9)           % Gráfica Error ALLAN VARIANCE
plot(tau/dt,E_AV, 'red'); grid on, title('ERROR ALLAN VARIANCE');
legend('Error AV'), xlabel(' n [muestras/clúster]'), ylabel('ERROR %');


% ------------------------------------  Guardado A.V. / S.D. Sensores  ---------------------------------

% Genera matriz “PSD” y exporta a fichero TXT
PSD = [psd_ax; psd_ay; psd_az; psd_wr; psd_wp; psd_wy]';
dlmwrite('PSD.txt', PSD, 'delimiter', '\t', 'precision', '% +.5e');

% Exporta vector “Tau” (tiempos clúster) a fichero TXT
dlmwrite('tau.txt', tau, 'delimiter', '\n', 'precision', '%.2f');

% Exporta vector “E_AV” (error AV) a fichero TXT
dlmwrite('Error_AV.txt', E_AV, 'delimiter', '\n', 'precision', '%.5f');

% Genera matriz “AV” y exporta a fichero TXT
AV = [AV_vx; AV_vy; AV_vz; AV_roll; AV_pitch; AV_yaw]';
dlmwrite('AV.txt', AV, 'delimiter', '\t', 'precision', '% +.10e');

% Genera matriz “SD” y exporta a fichero TXT
SD = [SD_vx; SD_vy; SD_vz; SD_roll; SD_pitch; SD_yaw]';
dlmwrite('SD.txt', SD, 'delimiter', '\t', 'precision', '% +.10e');

% ================================================================
```

```
% ====================================================================
% ====================  ERROR ESTIMACION A.V. ========================
% ====================================================================

% Ex = 1/(sqrt(2((LT/n)-1)));      %Error de estimación de Allan Variance

clear all

t = dlmread('t.txt', '\n');              % Importa datos desde fichero t.TXT
dt = t(1);
LT = t(2);

Fs = 1/dt;          % Frecuencia de muestreo

t_N = 1;            % Tiempo [s] para cte N
t_K = 3;            % Tiempo [s] para cte K

N = dlmread('N.txt', '\n');          % Importa datos desde fichero N.TXT
K = dlmread('K.txt', '\n');          % Importa datos desde fichero K.TXT

n_N = Fs*t_N;               % Número de muestras del clúster para cte N
n_K = Fs*t_K;               % Número de muestras del clúster para cte K

Bias = dlmread('Bias.txt', '\t');              % Importa datos desde fichero Bias.TXT
B = Bias(:,1);
Tb = Bias(:,2);

Bx = B(1);                  % Valor mínimo de AV-X
tb_X = Tb(1);               % Tiempo mínimo valor de AV-X

By = B(2);                  % Valor mínimo de AV-Y
tb_Y = Tb(2);               % Tiempo mínimo valor de AV-Y

Bz = B(3);                  % Valor mínimo de AV-Z
tb_Z = Tb(3);               % Tiempo mínimo valor de AV-Z

Broll = B(4);               % Valor mínimo de AV-Roll
tb_Roll = Tb(4);            % Tiempo mínimo valor de AV-Roll
```

```
Bpitch = B(5);          % Valor mínimo de AV-Pitch
tb_Pitch = Tb(5);       % Tiempo mínimo valor de AV-Pitch

Byaw = B(6);            % Valor mínimo de AV-Yaw
tb_Yaw = Tb(6);         % Tiempo mínimo valor de AV-Yaw


% ---------------------------------------- ERROR PARA CTE "N" ----------------------------------------------

E_N = (1/sqrt(2*((LT/n_N) - 1)))/100;     % Error de estimación en %

error_N = N.*E_N;                         % Valor error N de cada sensor


% ---------------------------------------- ERROR PARA CTE "B" ----------------------------------------------

E_Bx = 1/sqrt(2*((LT/(Fs*tb_X)) - 1));          %Error de estimación Allan Variance
error_Bx = Bx.*E_Bx;

E_By = 1/sqrt(2*((LT/(Fs*tb_Y)) - 1));          %Error de estimación Allan Variance
error_By = By.*E_By;

E_Bz = 1/sqrt(2*((LT/(Fs*tb_Z)) - 1));          %Error de estimación Allan Variance
error_Bz = Bz.*E_Bz;

E_Broll = 1/sqrt(2*((LT/(Fs*tb_Roll)) - 1));    %Error de estimación Allan Variance
error_Broll = Broll.*E_Broll;

E_Bpitch = 1/sqrt(2*((LT/(Fs*tb_Pitch)) - 1));  %Error de estimación Allan Variance
error_Bpitch = Bpitch.*E_Bpitch;

E_Byaw = 1/sqrt(2*((LT/(Fs*tb_Yaw)) - 1));      %Error de estimación Allan Variance
error_Byaw = Byaw.*E_Byaw;
```

```
% ---------------------------------------- ERROR PARA CTE "K" ----------------------------------------------

E_K = (1/sqrt(2*((LT/n_K) - 1)))/100;   %Error de estimación en %

error_K = K.*E_K;                    % Valor error K de cada sensor


% ----------------------------------------  Guardado Error Estimación  ---------------------------------------

dlmwrite('error_N.txt', error_N, 'precision', '% +.2e');

dlmwrite('error_K.txt', error_K, 'precision', '% +.2e');

% Crea vector Errores Bias y exporta a TXT
error_B = [error_Bx; error_By; error_Bz; error_Broll; error_Bpitch; error_Byaw]/100;
dlmwrite('error_B.txt', error_B, 'precision', '% +.2e');

% ================================================================================
```

```
% ================================================================
% ======================  BIAS INSTABILITY  ======================
% ================================================================

% Tx(t)+x(t)=v(t)  //  v(t): input white noise  //  x(t): Gauss-Markow process

clear all

t = dlmread('t.txt', '\n');              % Importa "dt" y "LT" desde fichero TXT
dt = t(1);
LT = t(2);

Bias = dlmread('Bias.txt', '\t');     % Importa datos desde fichero TXT
B = Bias(:,1);
Tb = Bias(:,2);

Bx = B(1);                % Valor mínimo de AV-X
tb_X = Tb(1);             % Tiempo mínimo valor de AV-X

By = B(2);                % Valor mínimo de AV-Y
tb_Y = Tb(2);             % Tiempo mínimo valor de AV-Y

Bz = B(3);                % Valor mínimo de AV-Z
tb_Z = Tb(3);             % Tiempo mínimo valor de AV-Z

Broll = B(4);             % Valor mínimo de AV-Roll
tb_Roll = Tb(4);          % Tiempo mínimo valor de AV-Roll

Bpitch = B(5);            % Valor mínimo de AV-Pitch
tb_Pitch = Tb(5);         % Tiempo mínimo valor de AV-Pitch

Byaw = B(6);              % Valor mínimo de AV-Yaw
tb_Yaw = Tb(6);           % Tiempo mínimo valor de AV-Yaw

e = exp(1);               % Número "e"
```

```
% ------------------------------------------- Cálculo de Ad ------------------------------------------------

Ad_X = e^((-(1/tb_X))*dt);
Ad_Y = e^((-(1/tb_Y))*dt);
Ad_Z = e^((-(1/tb_Z))*dt);

Ad_Roll = e^((-(1/tb_Roll))*dt);
Ad_Pitch = e^((-(1/tb_Pitch))*dt);
Ad_Yaw = e^((-(1/tb_Yaw))*dt);


% ---------------------------------------- Cálculo de Bd (integral) ----------------------------------------

Bd_X = -tb_X*( Ad_X - 1 );
Bd_Y = -tb_Y*( Ad_Y - 1 );
Bd_Z = -tb_Z*( Ad_Z - 1 );

Bd_Roll = -tb_Roll*( Ad_Roll - 1 );
Bd_Pitch = -tb_Pitch*( Ad_Pitch - 1 );
Bd_Yaw = -tb_Yaw*( Ad_Yaw - 1 );


% ---------------------------------- Desviación estándar Ruido Blanco ---------------------------------

SD_n_X = sqrt(1-(Ad_X)^2)*(Bx/Bd_X);
SD_n_Y = sqrt(1-(Ad_Y)^2)*(By/Bd_Y);
SD_n_Z = sqrt(1-(Ad_Z)^2)*(Bz/Bd_Z);

SD_n_Roll = sqrt(1-(Ad_Roll)^2)*(Broll/Bd_Roll);
SD_n_Pitch = sqrt(1-(Ad_Pitch)^2)*(Bpitch/Bd_Pitch);
SD_n_Yaw = sqrt(1-(Ad_Yaw)^2)*(Byaw/Bd_Yaw);


% ---------------------------------- BIAS ( proceso Gauss-Markov ) ------------------------------------

Bwn_X = randn(LT,1).*SD_n_X;
Bwn_Y = randn(LT,1).*SD_n_Y;
Bwn_Z = randn(LT,1).*SD_n_Z;
```

```
Bwn_Roll = randn(LT,1).*SD_n_Roll;
Bwn_Pitch = randn(LT,1).*SD_n_Pitch;
Bwn_Yaw = randn(LT,1).*SD_n_Yaw;

F_X(1) = 0;                 % condiciones iniciales nulas
F_Y(1) = 0;
F_Z(1) = 0;
F_Roll(1) = 0;
F_Pitch(1) = 0;
F_Yaw(1) = 0;

for k = 1:LT       % Generando modelo de Bias

        F_X(k+1) = Ad_X*F_X(k) + Bd_X*Bwn_X(k);
        F_X(k) = F_X(k+1);

        F_Y(k+1) = Ad_Y*F_Y(k) + Bd_Y*Bwn_Y(k);
        F_Y(k) = F_Y(k+1);

        F_Z(k+1) = Ad_Z*F_Z(k) + Bd_Z*Bwn_Z(k);
        F_Z(k) = F_Z(k+1);

        F_Roll(k+1) = Ad_Roll*F_Roll(k) + Bd_Roll*Bwn_Roll(k);
        F_Roll(k) = F_Roll(k+1);

        F_Pitch(k+1) = Ad_Pitch*F_Pitch(k) + Bd_Pitch*Bwn_Pitch(k);
        F_Pitch(k) = F_Pitch(k+1);

        F_Yaw(k+1) = Ad_Yaw*F_Yaw(k) + Bd_Yaw*Bwn_Yaw(k);
        F_Yaw(k) = F_Yaw(k+1);

end;

F_X = F_X(1:LT);                % Elimino el último término de F ( LT+1 )
F_Y = F_Y(1:LT);
F_Z = F_Z(1:LT);

F_Roll = F_Roll(1:LT);
F_Pitch = F_Pitch(1:LT);
F_Yaw = F_Yaw(1:LT);
```

```
% ---------------------------------------  Guardado Modelo BIAS  ---------------------------------------

% Genera matriz “F” (modelo BIAS) y exporta a fichero TXT
F = [F_X; F_Y; F_Z; F_Roll; F_Pitch; F_Yaw]';
dlmwrite('F.txt', F, 'delimiter', '\t', 'precision', '% +.6e');

% Genera vector “SD_n_Bias” y exporta a fichero TXT
SD_n_Bias = [SD_n_X; SD_n_Y; SD_n_Z; SD_n_Roll; SD_n_Pitch; SD_n_Yaw];
dlmwrite('SD_n_Bias.txt', SD_n_Bias, 'delimiter', '\n', 'precision', '% +.5e');

% ========================================================================
```

```
% ================================================================
% ==============  ACCELERATION/RATE RANDOM WALK  ==================
% ================================================================


t = dlmread('t.txt', '\n');          % Importa "dt" y "LT" desde fichero TXT
dt = t(1);
LT = t(2);

N = dlmread('N.txt', '\n');          % Importa Datos desde fichero N.TXT

Bias = dlmread('Bias.txt', '\t');    % Importa Datos desde fichero Bias.TXT
B = Bias(:,1);

K = dlmread('K.txt', '\n');          % Importa Datos desde fichero N.TXT

Nx = N(1);
Bx = B(1);
Kx = K(1);

Ny = N(2);
By = B(2);
Ky = K(2);

Nz = N(3);
Bz = B(3);
Kz = K(3);

Nroll = N(4);
Broll = B(4);
Kroll = K(4);

Npitch = N(5);
Bpitch = B(5);
Kpitch = K(5);

Nyaw = N(6);
Byaw = B(6);
Kyaw = K(6);
```

```
desv_est = dlmread('desv_est.txt', '\n');          % Importa “desv_est” desde fichero TXT

sd_ax = desv_est(1);
sd_ay = desv_est(2);
sd_az = desv_est(3);
sd_wr = desv_est(4);
sd_wp = desv_est(5);
sd_wy = desv_est(6);


Datos = dlmread('Datos.txt', '\t');          % Importa “Datos” desde fichero TXT

ax = Datos(:,1);
ay = Datos(:,2);
az = Datos(:,3);
wr = Datos(:,4);
wp = Datos(:,5);
wy = Datos(:,6);


% ================================================================
% ==============  ACCELERATION/RATE RANDOM WALK  ==================
% ================================================================

% rrw(z) = K*dfilt(z)*w(z)  //  w(z): ruido blanco de varianza desconocida  //  dfilt(z) = dt/(z-1)

%---------------------------------------  Acelerómetro X  ------------------------------------------------

sd_wn_X = XX;                          % Desviación estándar del Ruido Blanco

wn_X = randn(LT,1).*sd_wn_X;           % Generación del Ruido Blanco

nd_X = [-Kx*dt];                       % Numerador del Filtro
dd_X = [1 -1];                         % Denominador del Filtro

Acc_rw_X = filter(nd_X,dd_X,wn_X);     % Filtrado del Ruido Blanco (Ruido Coloreado)

[pds_Acc_rw_X F] = pwelch(Acc_rw_X,[],[],[],1/dt);       % PSD del Ruido Blanco
[pds_ax Fax] = pwelch(ax,[],[],[],1/dt);                 % PSD del Sensor
```

```
figure(40)          % Graficado de la PSD del sensor y la PSD del modelo generado.
loglog(F,pds_Acc_rw_X,'r'), grid on, hold on;
loglog(Fax,pds_ax), legend('Acceleration Random Walk PSD','Acelerómetro X PSD'),
xlabel('Frecuencia [Hz]'), ylabel('Potencia, Sensor X');


%--------------------------------------- Acelerómetro Y ----------------------------------------------------

sd_wn_Y = YY                        % Desviación estándar del Ruido Blanco

wn_Y = randn(LT,1).*sd_wn_Y;        % Generación del Ruido Blanco

nd_Y = [-Ky*dt];                    % Numerador del Filtro
dd_Y = [1 -1];                      % Denominador del Filtro

Acc_rw_Y = filter(nd_Y,dd_Y,wn_Y);  % Filtrado del Ruido Blanco (Ruido Coloreado)

[pds_Acc_rw_Y F] = pwelch(Acc_rw_Y,[],[],[],1/dt);      % PSD del Ruido Blanco
[pds_ay Fay] = pwelch(ay,[],[],[],1/dt);                % PSD del Sensor

figure(41)          % Graficado de la PSD del sensor y la PSD del modelo generado.
loglog(F,pds_Acc_rw_Y,'r'), grid on, hold on;
loglog(Fay,pds_ay), legend('Acceleration Random Walk PSD','Acelerómetro Y PSD'),
xlabel('Frecuencia [Hz]'), ylabel('Potencia, Sensor Y');


%--------------------------------------- Acelerómetro Z ----------------------------------------------------

sd_wn_Z = ZZ;                       % Desviación estándar del Ruido Blanco

wn_Z = randn(LT,1).*sd_wn_Z;        % Generación del Ruido Blanco

nd_Z = [-Kz*dt];                    % Numerador del Filtro
dd_Z = [1 -1];                      % Denominador del Filtro

Acc_rw_Z = filter(nd_Z,dd_Z,wn_Z);  % Filtrado del Ruido Blanco (Ruido Coloreado)

[pds_Acc_rw_Z F] = pwelch(Acc_rw_Z,[],[],[],1/dt);      % PSD del Ruido Blanco
[pds_az Faz] = pwelch(az,[],[],[],1/dt);                % PSD del Sensor
```

```
figure(42)          % Graficado de la PSD del sensor y la PSD del modelo generado.
loglog(F,pds_Acc_rw_Z,'r'), grid on, hold on;
loglog(Faz,pds_az), legend('Acceleration Random Walk PSD','Acelerómetro Z PSD'),
xlabel('Frecuencia [Hz]'), ylabel('Potencia, Sensor Z');


%----------------------------------------- Giroscopio X ----------------------------------------------------

sd_wn_Roll = 0;                         % Desviación estándar del Ruido Blanco

wn_Roll = randn(LT,1).*sd_wn_Roll;      % Generación del Ruido Blanco

nd_Roll = [-Kroll*dt];                  % Numerador del Filtro
dd_Roll = [1 -1];                       % Denominador del Filtro

rrw_Roll = filter(nd_Roll,dd_Roll,wn_Roll);             % Filtrado del Ruido Blanco

[pds_rrw_Roll F] = pwelch(rrw_Roll,[],[],[],1/dt);      % PSD del Ruido Blanco
[pds_wr Fwr] = pwelch(wr,[],[],[],1/dt);                % PSD del Sensor

figure(43)          % Graficado de la PSD del sensor y la PSD del modelo generado.
loglog(F,pds_rrw_Roll,'r'), grid on, hold on;
loglog(Fwr,pds_wr), legend('Rate Random Walk PSD','Giroscopio X PSD'), xlabel('Frecuencia
[Hz]'), ylabel('Potencia, Sensor X');


%----------------------------------------- Giroscopio Y ----------------------------------------------------

sd_wn_Pitch = 0;                        % Desviación estándar del Ruido Blanco

wn_Pitch = randn(LT,1).*sd_wn_Pitch;    % Generación del Ruido Blanco

nd_Pitch = [-Kpitch*dt];                % Numerador del Filtro
dd_Pitch = [1 -1];                      % Denominador del Filtro

rrw_Pitch = filter(nd_Pitch,dd_Pitch,wn_Pitch);         % Filtrado del Ruido Blanco

[pds_rrw_Pitch F] = pwelch(rrw_Pitch,[],[],[],1/dt);    % PSD del Ruido Blanco
[pds_wp Fwp] = pwelch(wp,[],[],[],1/dt);                % PSD del Sensor
```

```
figure(44)          % Graficado de la PSD del sensor y la PSD del modelo generado.
loglog(F,pds_rrw_Pitch,'r'), grid on, hold on;
loglog(Fwp,pds_wp), legend('Rate Random Walk PSD','Giroscopio Y PSD'),
xlabel('Frecuencia [Hz]'), ylabel('Potencia, Sensor Y');


%------------------------------------------  Giroscopio Z  ------------------------------------------------------

sd_wn_Yaw = 0;                                  % Desviación estándar del Ruido Blanco

wn_Yaw = randn(LT,1).*sd_wn_Yaw;                % Generación del Ruido Blanco

nd_Yaw = [-Kyaw*dt];                            % Numerador del Filtro
dd_Yaw = [1 -1];                                % Denominador del Filtro

rrw_Yaw = filter(nd_Yaw,dd_Yaw,wn_Yaw);         % Filtrado del Ruido Blanco

[pds_rrw_Yaw F] = pwelch(rrw_Yaw,[],[],[],1/dt);          % PSD del Ruido Blanco
[pds_wy Fwy] = pwelch(wy,[],[],[],1/dt);                  % PSD del Sensor

figure(45)          % Graficado de la PSD del sensor y la PSD del modelo generado.
loglog(F,pds_rrw_Yaw,'r'), grid on, hold on;
loglog(Fwy,pds_wy), legend('Rate random walk PSD','Giroscopio Z PSD'), xlabel('Frecuencia
[Hz]'), ylabel('Potencia, Sensor Z');
```

```
% ================================================================
% ================ S.D. VELOCITY / ANGULAR RANDOM WALK  ===============
% ================================================================

% sd_vrw = sqrt( (sd_sensor_noise)^2 - (B)^2 - (sd_arw)^2 ) = N  ACELEROMETROS
% sd_arw = sqrt( (sd_sensor_noise)^2 - (B)^2 - (sd_rrw)^2 ) = N  GIROSCOPIOS

% sd_vrw_X = Kx              sd_vrw_Y = Ky              sd_vrw_Z = Kz
% sd_arw_Roll = Kroll        sd_arw_Pitch = Kpitch      sd_arw_Yaw = Kyaw

% sd_sensor_noise_X = sd_ax        sd_sensor_noise_Y = sd_ay
% sd_sensor_noise_Z = sd_az        sd_sensor_noise_Roll = sd_wr
% sd_sensor_noise_Pitch = sd_wp    sd_sensor_noise_Yaw = sd_wy


sd_vel_rw_X = sqrt( (sd_ax)^2 - (Qx)^2 - (Bx)^2 - (Kx)^2 - (Rx)^2 );
sd_vel_rw_Y = sqrt( (sd_ay)^2 - (Qy)^2 - (By)^2 - (Ky)^2 - (Ry)^2 );
sd_vel_rw_Z = sqrt( (sd_az)^2 - (Qz)^2 - (Bz)^2 - (Kz)^2 - (Rz)^2 );


sd_ang_rw_X = sqrt( (sd_wr)^2 - (Qroll)^2 - (Broll)^2 - (Kroll)^2 - (Rroll)^2 );
sd_ang_rw_Y = sqrt( (sd_wp)^2 - (Qpitch)^2 - (Bpitch)^2 - (Kpitch)^2 - (Rpitch)^2 );
sd_ang_rw_Z = sqrt( (sd_wy)^2 - (Qyaw)^2 - (Byaw)^2 - (Kyaw)^2 - (Ryaw)^2 );


Vel_rw_X = randn(LT,1).*sd_vel_rw_X;      % Ruido Blanco
Vel_rw_Y = randn(LT,1).*sd_vel_rw_Y;      % Ruido Blanco
Vel_rw_Z = randn(LT,1).*sd_vel_rw_Z;      % Ruido Blanco


Ang_rw_X = randn(LT,1).*sd_ang_rw_X;      % Ruido Blanco
Ang_rw_Y = randn(LT,1).*sd_ang_rw_Y;      % Ruido Blanco
Ang_rw_Z = randn(LT,1).*sd_ang_rw_Z;      % Ruido Blanco
```

```
% ---------------------------------------- Guardado Modelo ARRW ----------------------------------------

% Genera vector “SD_WN” y exporta a fichero TXT
SD_WN = [sd_wn_X; sd_wn_Y; sd_wn_Z; sd_wn_Roll; sd_wn_Pitch; sd_wn_Yaw];
dlmwrite('SD_WN.txt', SD_WN, 'delimiter', '\n', 'precision', '% +.5e');

% Genera matriz “A_R_RW” y exporta a fichero TXT
A_R_RW = [Acc_rw_X'; Acc_rw_Y'; Acc_rw_Z'; rrw_Roll'; rrw_Pitch'; rrw_Yaw']';
dlmwrite('A_R_RW.txt', A_R_RW, 'delimiter', '\t', 'precision', '% +.5e');

% Genera vector “SD_RW” y exporta a fichero TXT
SD_RW = [sd_vel_rw_X; sd_vel_rw_Y; sd_vel_rw_Z; sd_ang_rw_X; sd_ang_rw_Y;
sd_ang_rw_Z];
dlmwrite('SD_RW.txt', SD_RW, 'delimiter', '\n', 'precision', '% +.5e');

% Genera matriz “V_A_RW” y exporta a fichero TXT
V_A_RW = [Vel_rw_X'; Vel_rw_Y'; Vel_rw_Z'; Ang_rw_X'; Ang_rw_Y'; Ang_rw_Z']';
dlmwrite('V_A_RW.txt', V_A_RW,'delimiter', '\t', 'precision', '% +.5e');

% ========================================================================
```

```
% ================================================================
% ====================  SENSOR NOISE MODEL  ======================
% ================================================================

clear all

% ============================  Datos  ===============================

t = dlmread('t.txt', '\n');                    % Importa "dt" y "LT" desde fichero t.TXT
dt = t(1);
LT = t(2);

T_Data = dlmread('T_Data.txt', '\n');      % Importa “T_Data” desde fichero TXT


Datos = dlmread('Datos.txt', '\t');          % Importa “Datos” desde fichero TXT

ax = Datos(:,1);
ay = Datos(:,2);
az = Datos(:,3);

wr = Datos(:,4);
wp = Datos(:,5);
wy = Datos(:,6);


% ----------------------------------------  BIAS MODEL  ----------------------------------------------

F = dlmread('F.txt', '\t');              % Importa “F” desde fichero TXT

F_X = F(:,1);
F_Y = F(:,2);
F_Z = F(:,3);

F_Roll = F(:,4);
F_Pitch = F(:,5);
F_Yaw = F(:,6);
```

```
% ---------------------------------------- RANDOM MODEL ----------------------------------------------------

A_R_RW = dlmread('A_R_RW.txt', '\t');          % Importa “A_R_RW” desde fichero TXT

Acc_rw_X = A_R_RW(:,1);
Acc_rw_Y = A_R_RW(:,2);
Acc_rw_Z = A_R_RW(:,3);

rrw_Roll = A_R_RW(:,4);
rrw_Pitch = A_R_RW(:,5);
rrw_Yaw = A_R_RW(:,6);


V_A_RW = dlmread('V_A_RW.txt', '\t');          % Importa “V_R_RW” desde fichero TXT

Vel_rw_X = V_A_RW(:,1);
Vel_rw_Y = V_A_RW(:,2);
Vel_rw_Z = V_A_RW(:,3);

Ang_rw_X = V_A_RW(:,4);
Ang_rw_Y = V_A_RW(:,5);
Ang_rw_Z = V_A_RW(:,6);


% ------------- Accel_Model = F_N(k) + Vel_rw_N(k) + Acc_rw_N(k) + Q(k) + R(k) ------------

ModelNoise_X = F_X + Vel_rw_X + Acc_rw_X;
ModelNoise_Y = F_Y + Vel_rw_Y + Acc_rw_Y;
ModelNoise_Z = F_Z + Vel_rw_Z + Acc_rw_Z;


% ---------------- Gyro_Model = F_N(k) + Ang_rw_N(k) + rrw_N(k) + Q(k) + R(k) ----------------

ModelNoise_Roll = F_Roll + Ang_rw_X + rrw_Roll;
ModelNoise_Pitch = F_Pitch + Ang_rw_Y + rrw_Pitch;
ModelNoise_Yaw = F_Yaw + Ang_rw_Z + rrw_Yaw;


% ---------------------------------------------------------------------------------------------------------
```

```
% ========================================================================
% ========================= MODEL FIT ============================
% ========================================================================

X_n = 0;
X_d = 0;

Y_n = 0;
Y_d = 0;

Z_n = 0;
Z_d = 0;

Roll_n = 0;
Roll_d = 0;

Pitch_n = 0;
Pitch_d = 0;

Yaw_n = 0;
Yaw_d = 0;


for k = 1:LT                % Cálculo de los sumatorios

        X_n = X_n + ((ax(k) - ModelNoise_X(k))^2);
        X_d = X_d + (ax(k))^2;

        Y_n = Y_n + ((ay(k) - ModelNoise_Y(k))^2);
        Y_d = Y_d + (ay(k))^2;

        Z_n = Z_n + ((az(k) - ModelNoise_Z(k))^2);
        Z_d = Z_d + (az(k))^2;

        Roll_n = Roll_n + ((wr(k) - ModelNoise_Roll(k))^2);
        Roll_d = Roll_d + (wr(k))^2;

        Pitch_n = Pitch_n + ((wp(k) - ModelNoise_Pitch(k))^2);
        Pitch_d = Pitch_d + (wp(k))^2;
```

```
        Yaw_n = Yaw_n + ((wy(k) - ModelNoise_Yaw(k))^2);
        Yaw_d = Yaw_d + (wy(k))^2;

end;


% ========================== FIT ================================

Fit_Acc_X = (1 - (sqrt(X_n)/sqrt(X_d))/100)*100;          % En tanto por ciento
Fit_Acc_Y = (1 - (sqrt(Y_n)/sqrt(Y_d))/100)*100;
Fit_Acc_Z = (1 - (sqrt(Z_n)/sqrt(Z_d))/100)*100;

Fit_Gyro_X = (1 - (sqrt(Roll_n)/sqrt(Roll_d))/100)*100;          % En tanto por ciento
Fit_Gyro_Y = (1 - (sqrt(Pitch_n)/sqrt(Pitch_d))/100)*100;
Fit_Gyro_Z = (1 - (sqrt(Yaw_n)/sqrt(Yaw_d))/100)*100;


figure(15)        % Gráfica triple del FIT de los acelerómetros
subplot(3,1,1);
plot(T_Data,ax); grid on, title('PRECISION'), hold on;
plot(T_Data,ModelNoise_X); grid on, legend('Acelerómetro X','ModelNoise_X'),
xlabel('Tiempo [s]'), ylabel('Aceleración lineal [m/s2]');

subplot(3,1,2);
plot(T_Data,ay); grid on, title('PRECISION'), hold on;
plot(T_Data,ModelNoise_Y); grid on, legend('Acelerómetro Y','ModelNoise_Y'),
xlabel('Tiempo [s]'), ylabel('Aceleración lineal [m/s2]');

subplot(3,1,3);
plot(T_Data,az); grid on, title('PRECISION'), hold on;
plot(T_Data,ModelNoise_Z); grid on, legend('Acelerómetro Z','ModelNoise_Z'), xlabel('Tiempo
[s]'), ylabel('Aceleración lineal [m/s2]');


figure(16)        % Gráfica triple del FIT de los giroscopios
subplot(3,1,1);
plot(T_Data,wr); grid on, title('PRECISION'), hold on;
plot(T_Data,ModelNoise_Roll); grid on, legend('Giroscopio X','ModelNoise_Roll'),
xlabel('Tiempo [s]'), ylabel('Velocidad angular [º/s]');
```

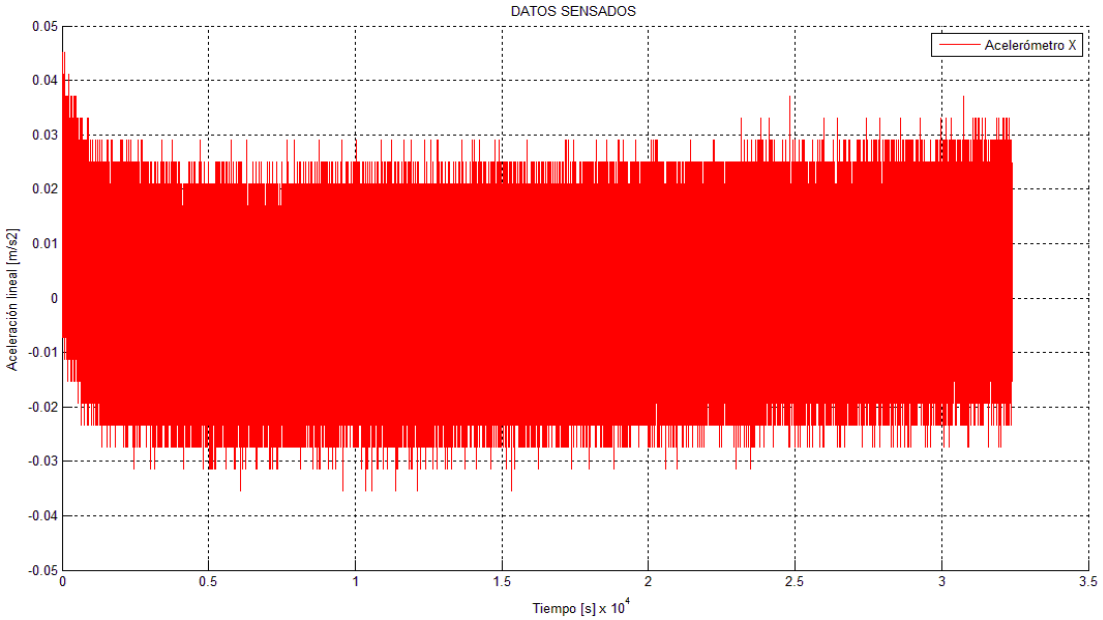
```
subplot(3,1,2);
plot(T_Data,wp); grid on, title('PRECISION'), hold on;
plot(T_Data,ModelNoise_Pitch); grid on, legend('Giroscopio Y','ModelNoise_Pitch'),
xlabel('Tiempo [s]'), ylabel('Velocidad angular [º/s]');

subplot(3,1,3);
plot(T_Data,wy); grid on, title('PRECISION'), hold on;
plot(T_Data,ModelNoise_Yaw); grid on, legend('Giroscopio Z','ModelNoise_Yaw'),
xlabel('Tiempo [s]'), ylabel('Velocidad angular [º/s]');


% -------------------------------  Guardado Noise Models / FIT  ------------------------------------

% Genera matriz “Noise_Model” y exporta a fichero TXT
Noise_Model = [ModelNoise_X; ModelNoise_Y; ModelNoise_Z; ModelNoise_Roll;
ModelNoise_Pitch; ModelNoise_Yaw]';
dlmwrite('Noise_Model.txt', Noise_Model, 'delimiter', '\t', 'precision', '% +.5e');

% Genera matriz “FIT” y exporta a fichero TXT
FIT = [Fit_Acc_X; Fit_Acc_Y; Fit_Acc_Z; Fit_Gyro_X; Fit_Gyro_Y; Fit_Gyro_Z]';
dlmwrite('FIT.txt', FIT, 'delimiter', '\t', 'precision', '% +.2d');


% ==========================================================================
```

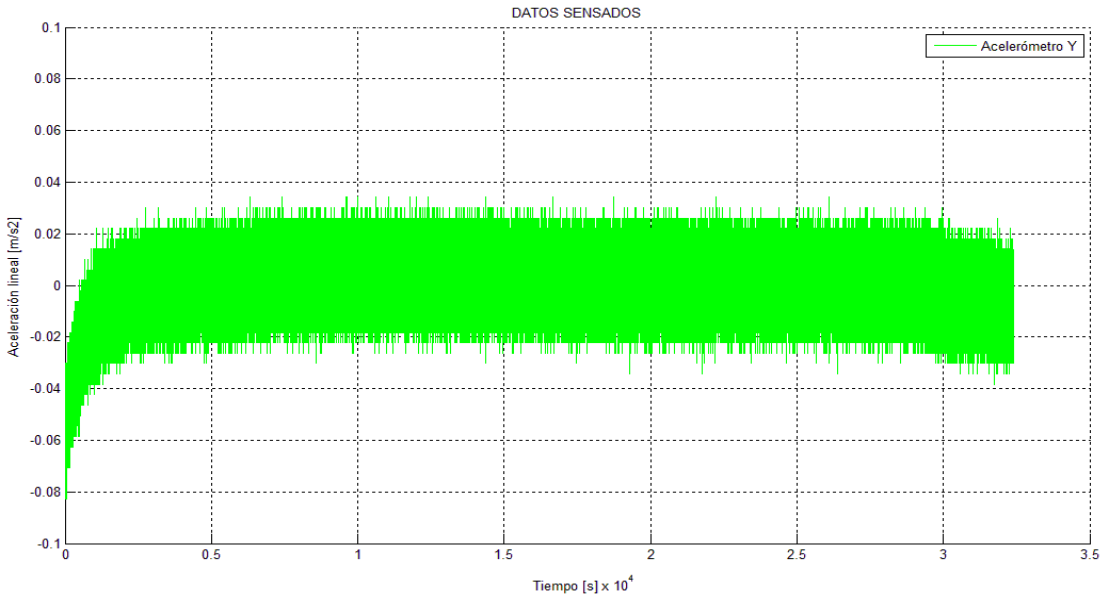
d) Gráficas.

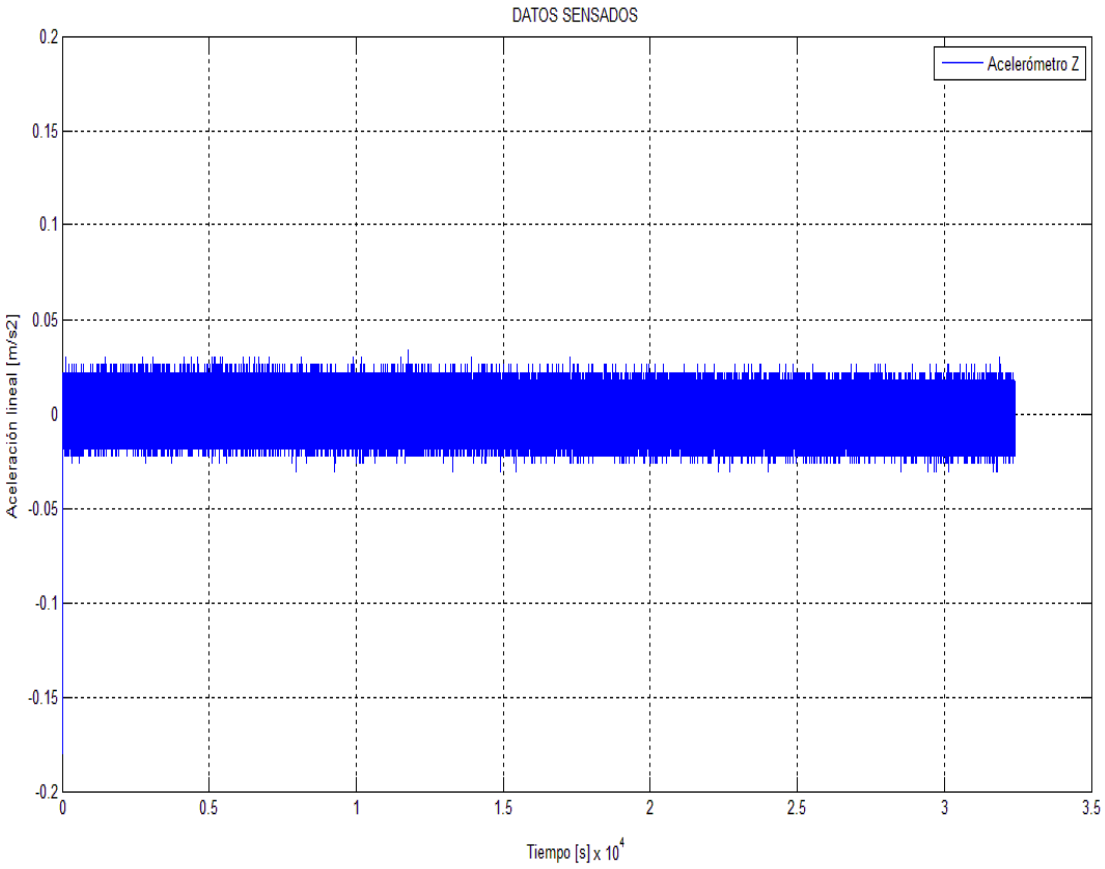
A continuación se presentan las gráficas de resultados de cada sensor para las frecuencias de muestreo seleccionadas.

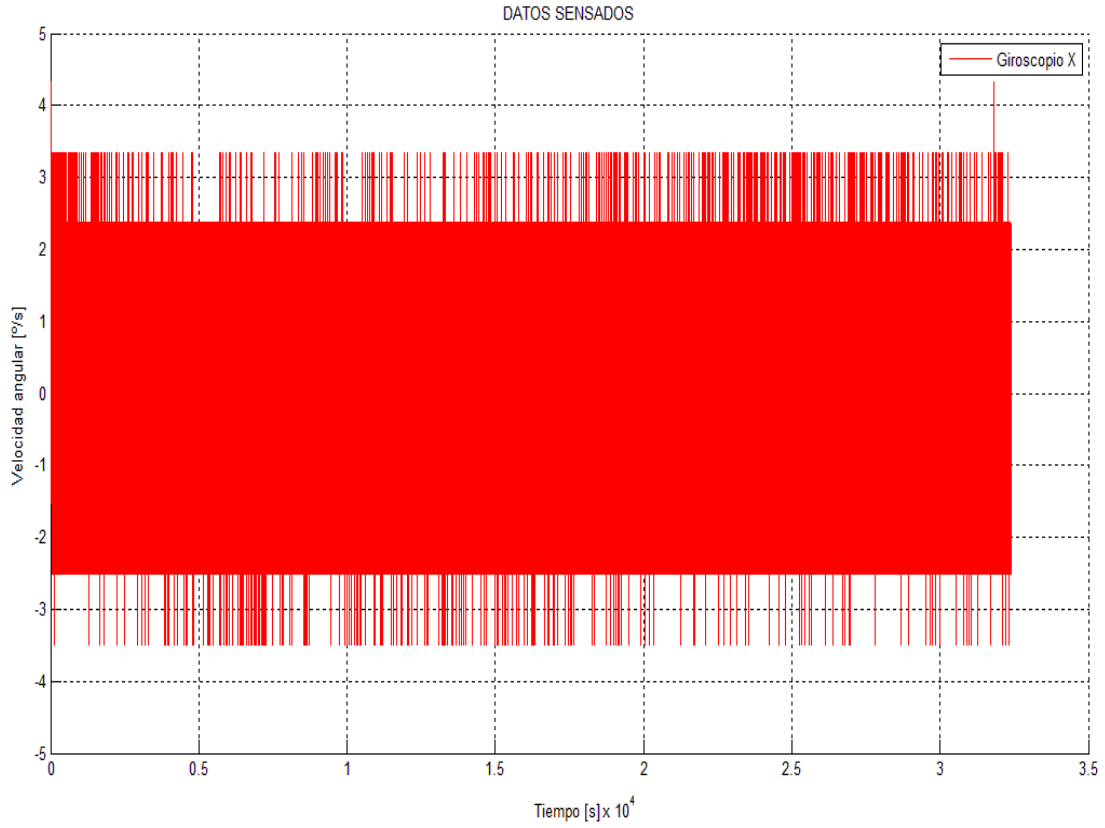
Fs = 100Hz

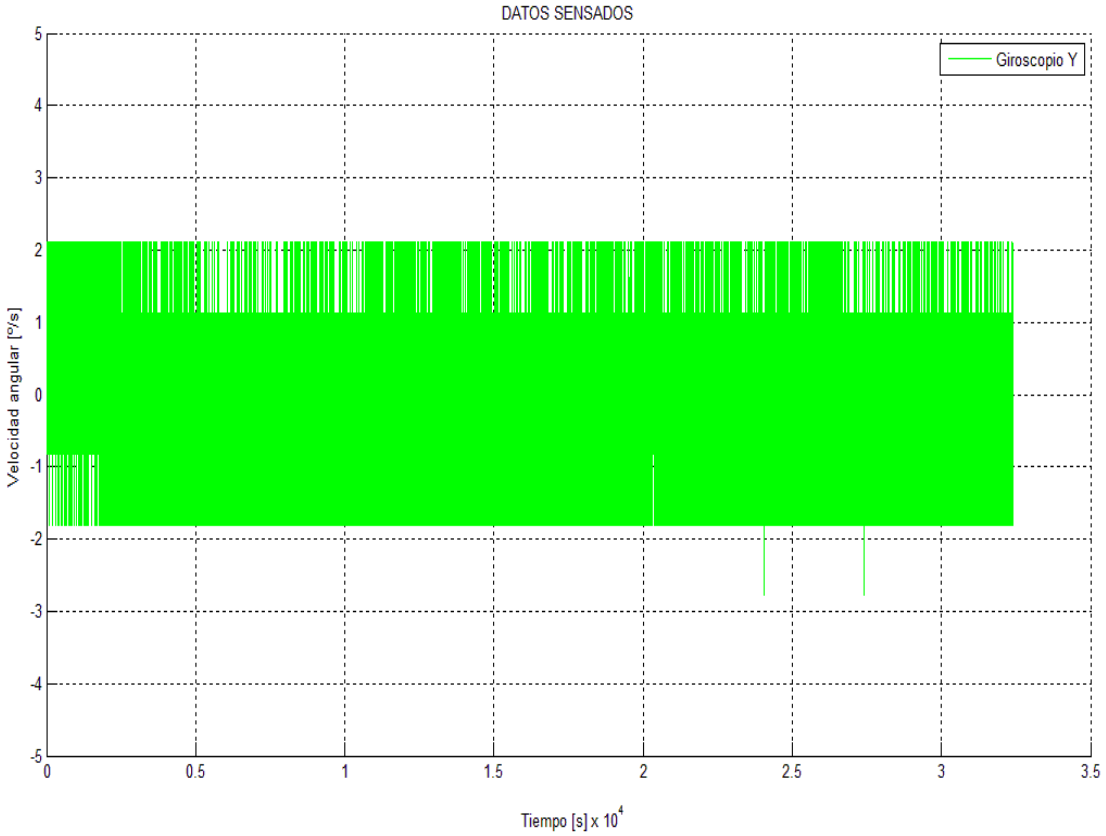
- Lecturas de los sensores:

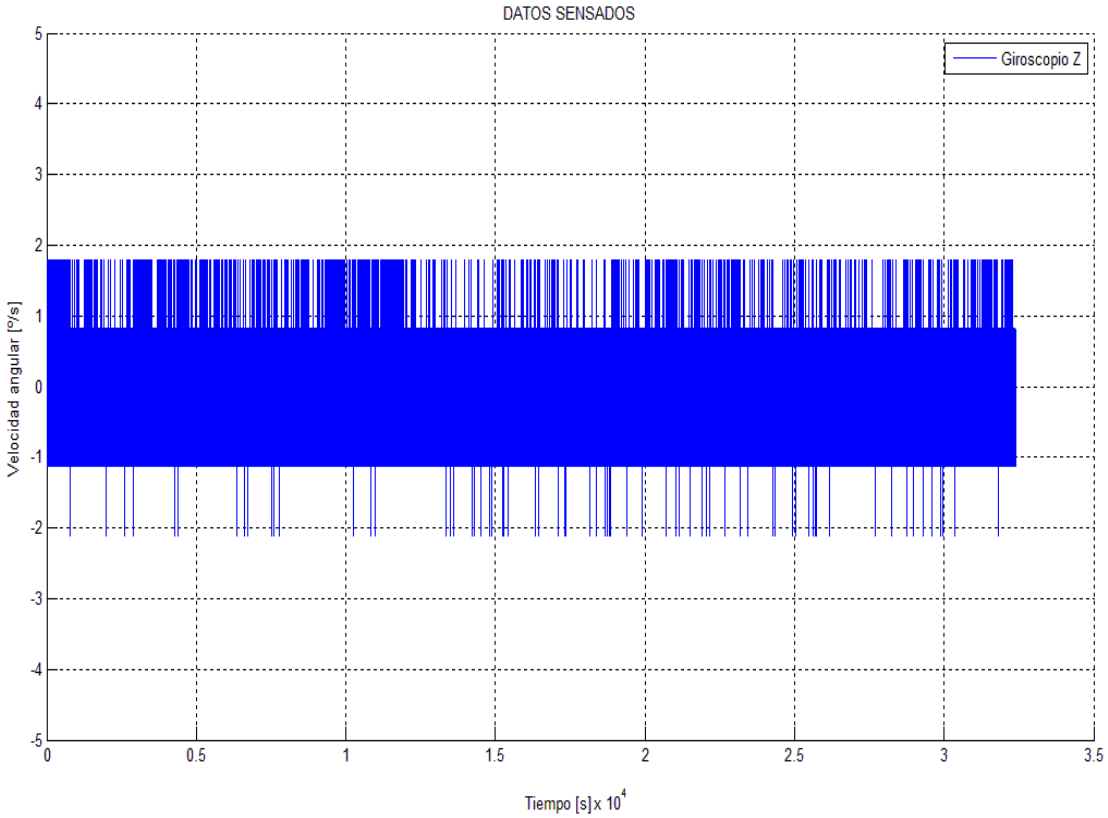




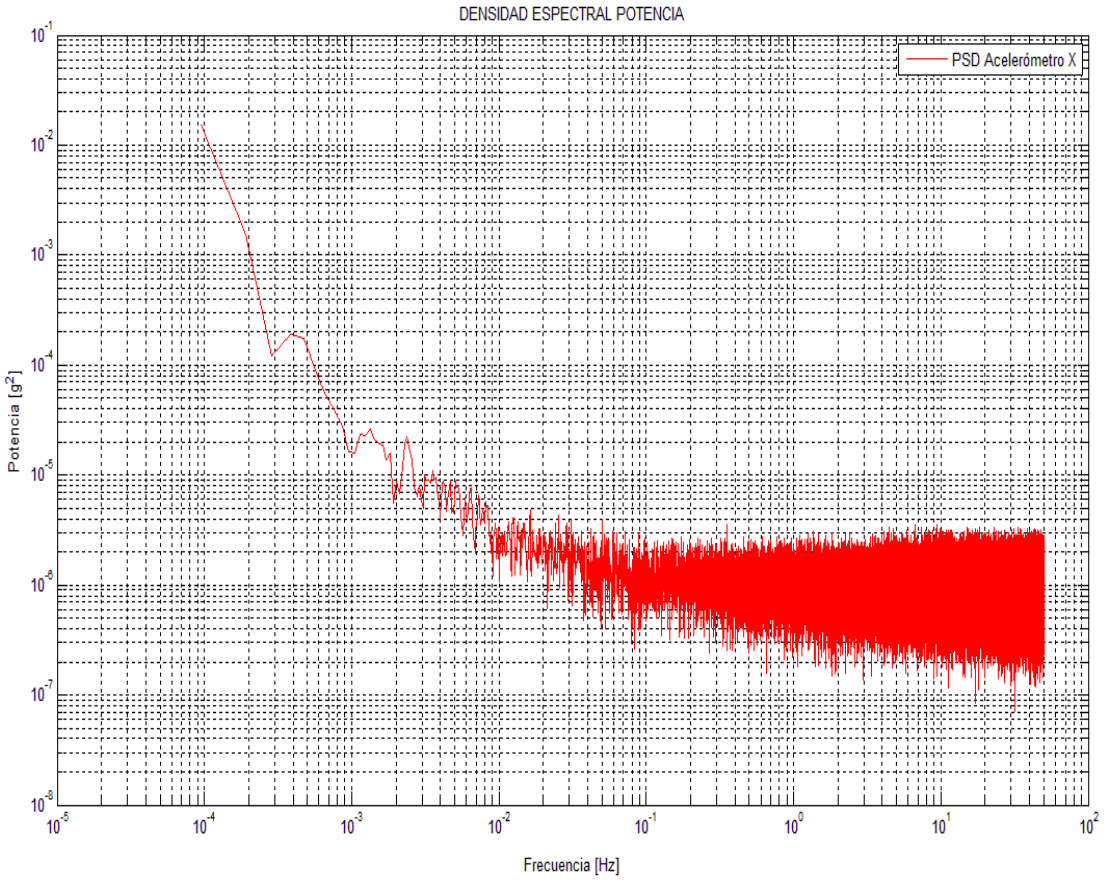

DATOS SENSADOS
Acelerómetro Z
Aceleración lineal [m/s2]
Tiempo [s] x 10^4
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
0
0.5
1
1.5
2
2.5
3
3.5

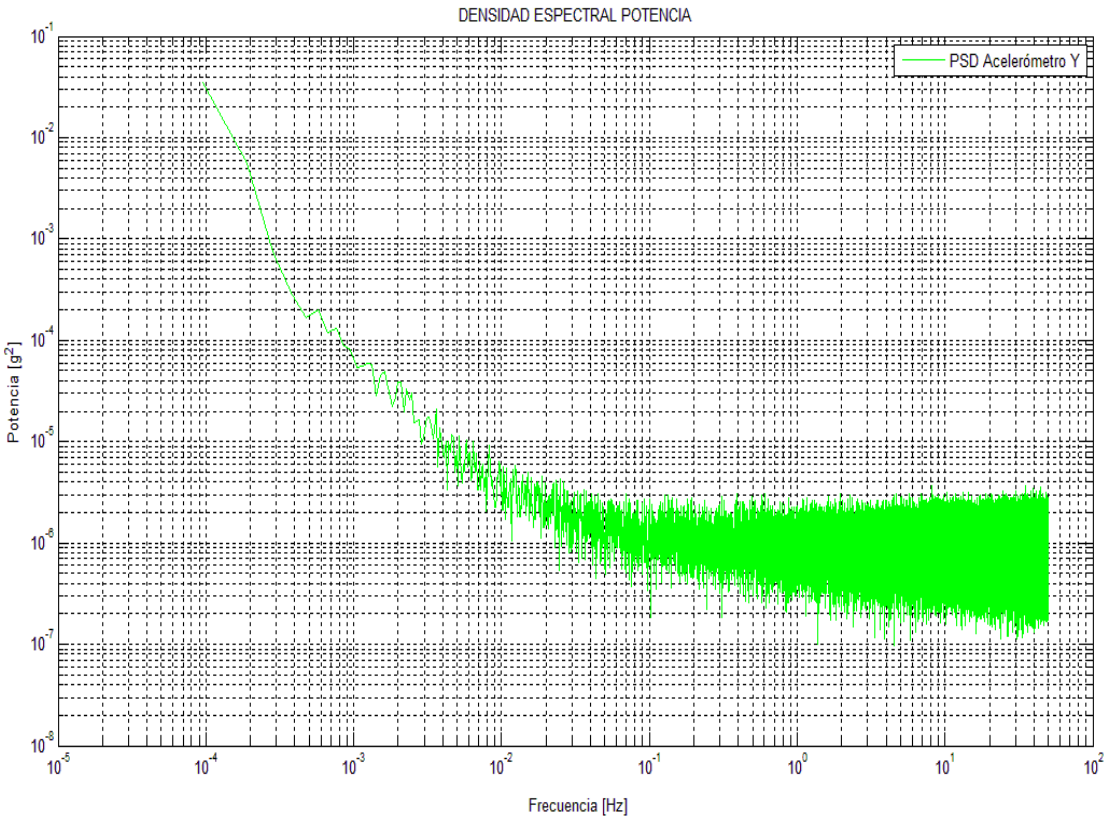

DATOS SENSADOS
Giroscopio X
Velocidad angular [º/s]
Tiempo [s] x 10^4
5
4
3
2
1
0
-1
-2
-3
-4
-5
0
0.5
1
1.5
2
2.5
3
3.5

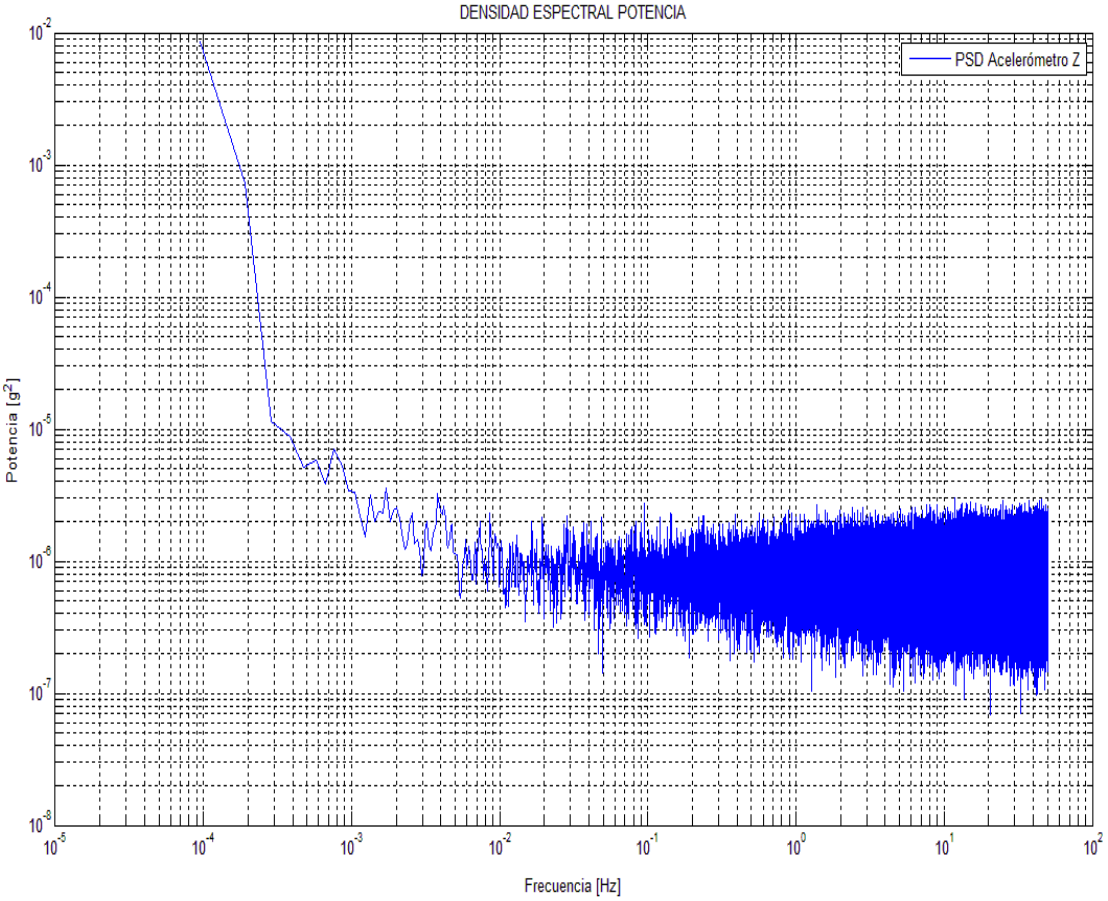

DATOS SENSADOS
Giroscopio Y
Velocidad angular [°/s]
Tiempo [s] x 10^4
5
4
3
2
1
0
-1
-2
-3
-4
-5
0
0.5
1
1.5
2
2.5
3
3.5

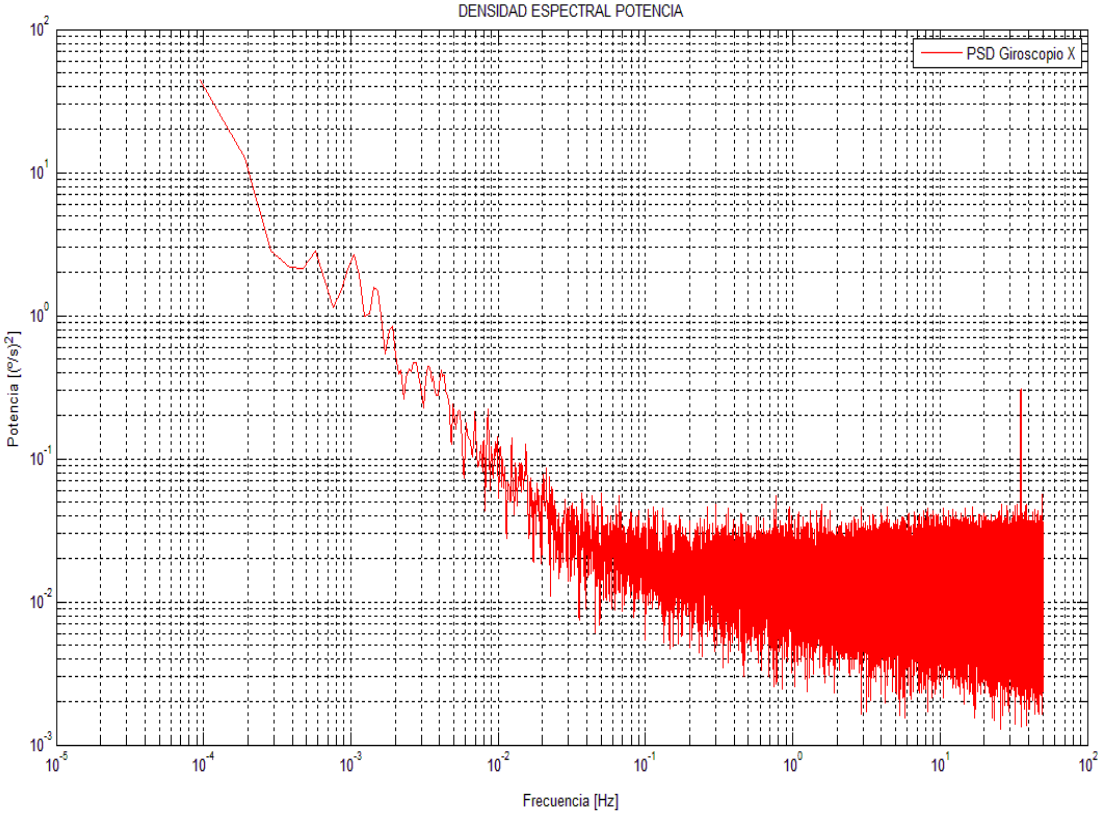

DATOS SENSADOS
Giroscopio Z
Velocidad angular [°/s]
Tiempo [s] x 10^4
5
4
3
2
1
0
-1
-2
-3
-4
-5
0
0.5
1
1.5
2
2.5
3
3.5

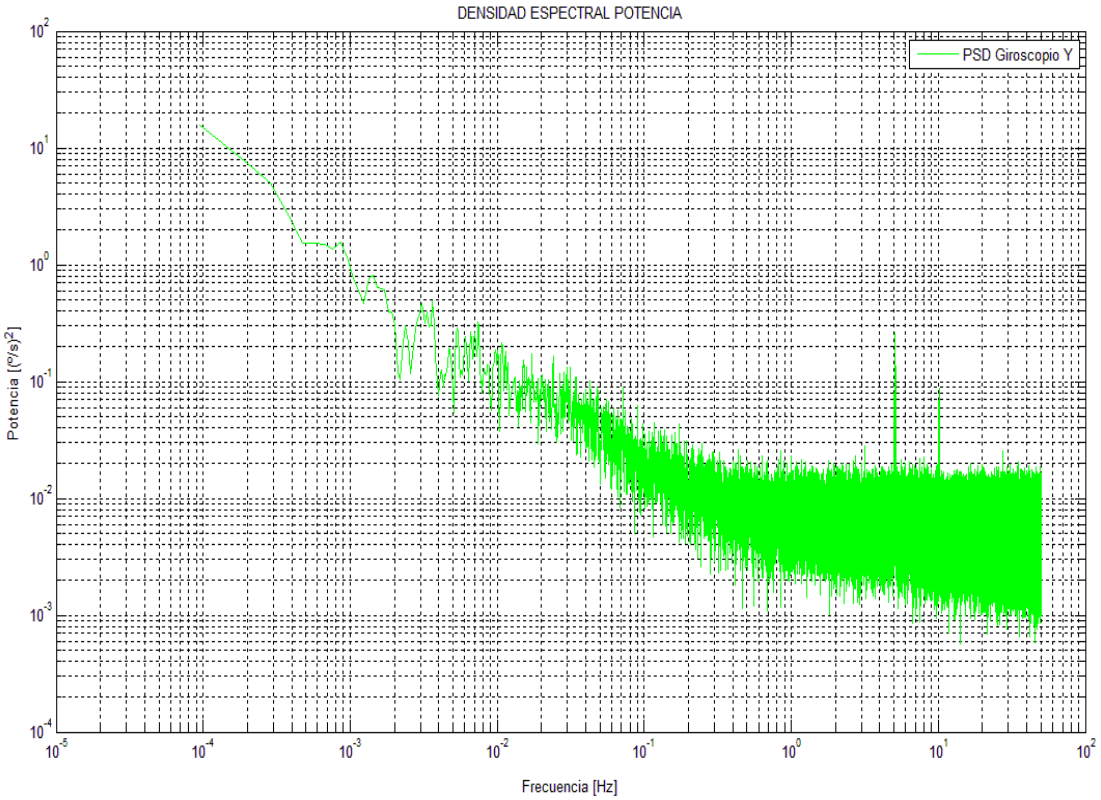
- Densidad espectral de potencia de los sensores:

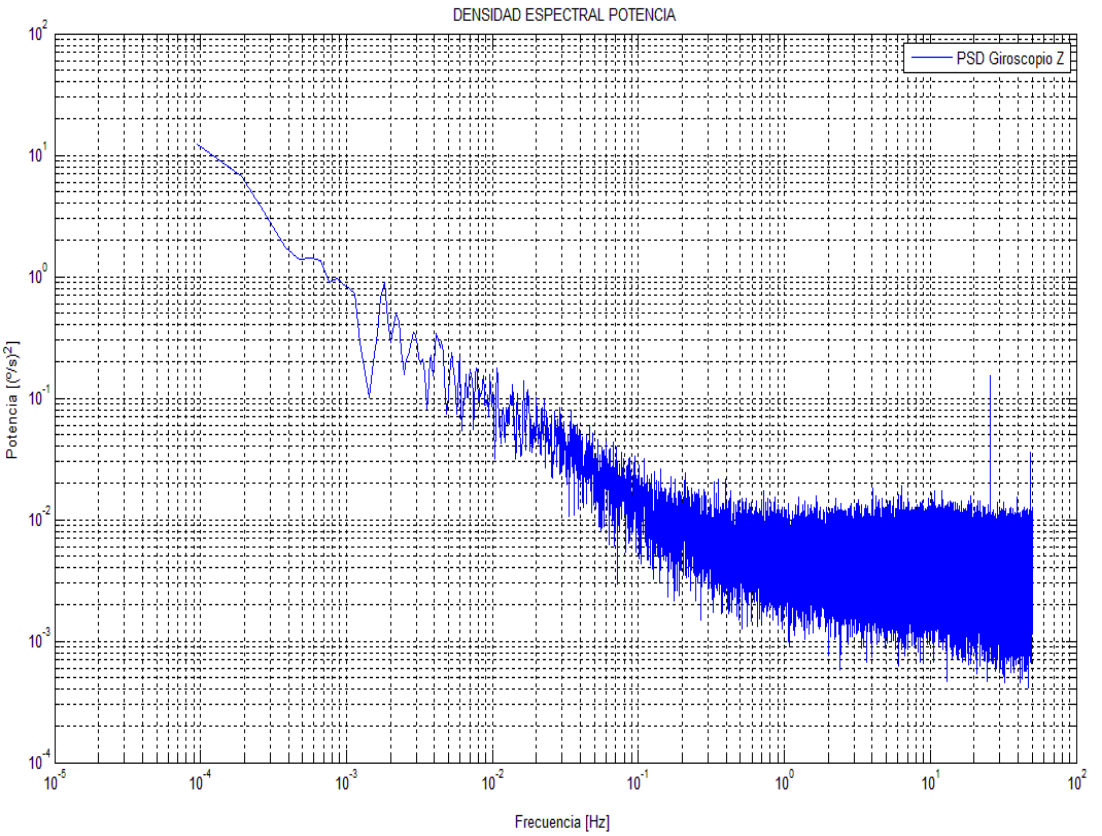




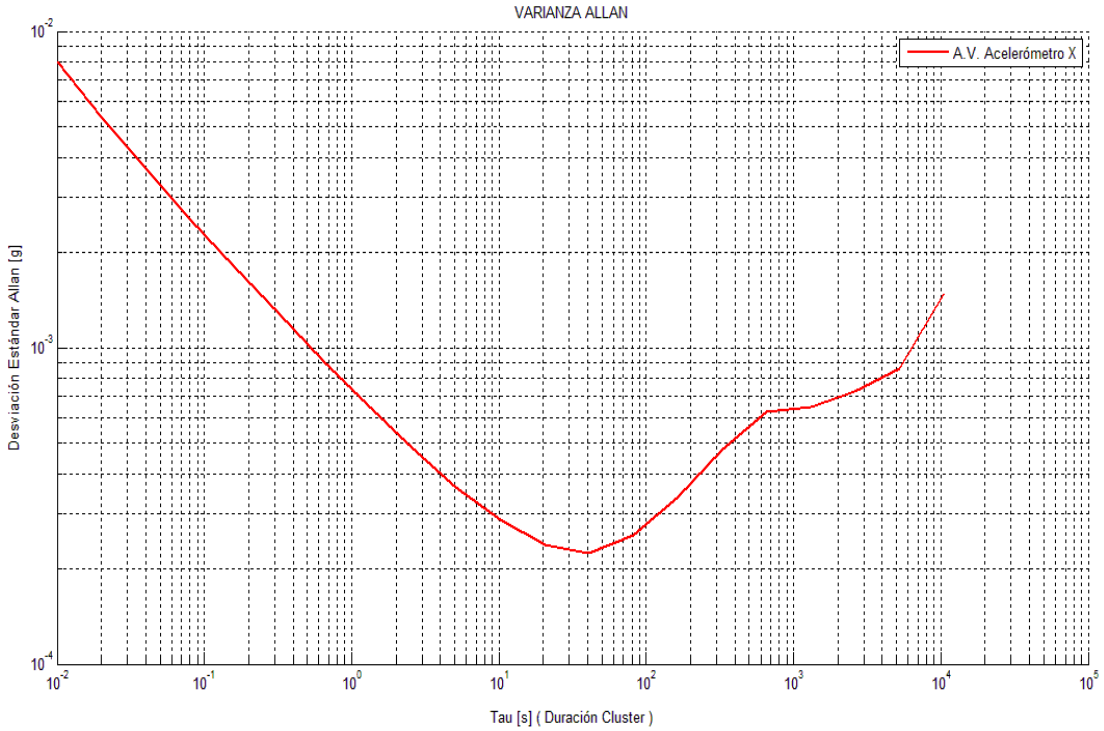

DENSIDAD ESPECTRAL POTENCIA
PSD Acelerómetro Z
Potencia [g²]
Frecuencia [Hz]

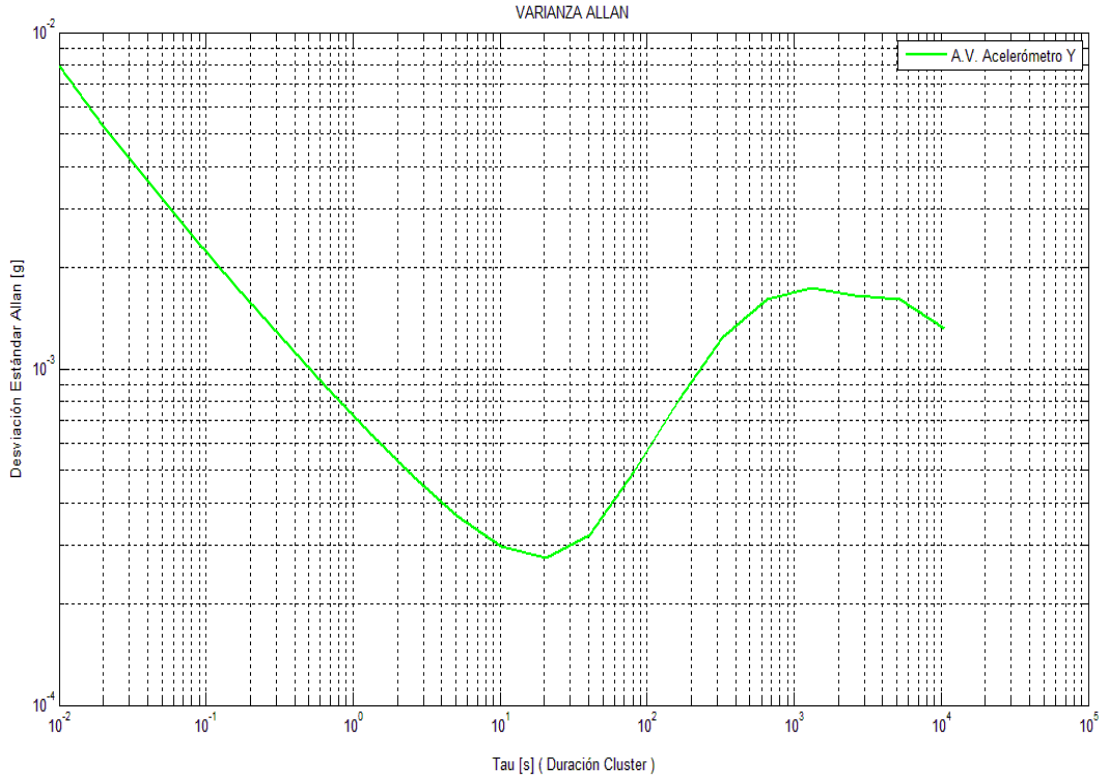

DENSIDAD ESPECTRAL POTENCIA
PSD Giroscopio X
Potencia [(º/s)²]
Frecuencia [Hz]

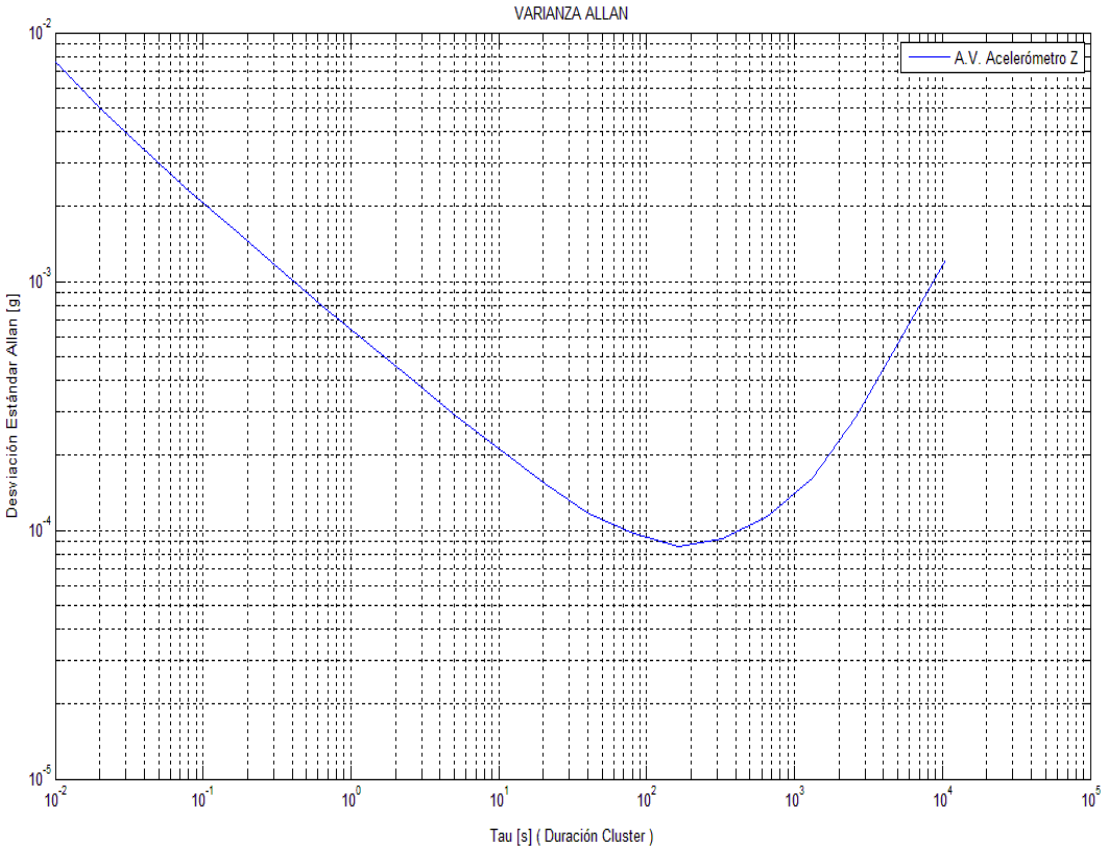

DENSIDAD ESPECTRAL POTENCIA
PSD Giroscopio Y
Potencia [(°/s)²]
Frecuencia [Hz]

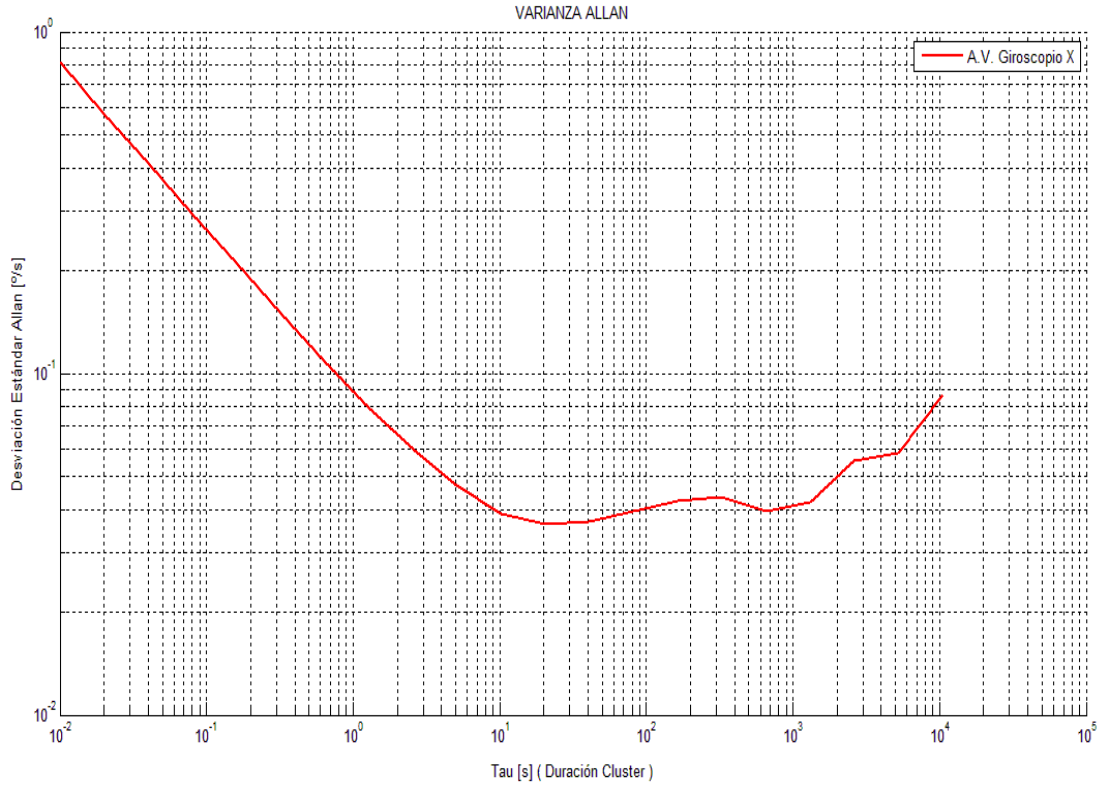

DENSIDAD ESPECTRAL POTENCIA
PSD Giroscopio Z
Potencia [(°/s)²]
Frecuencia [Hz]

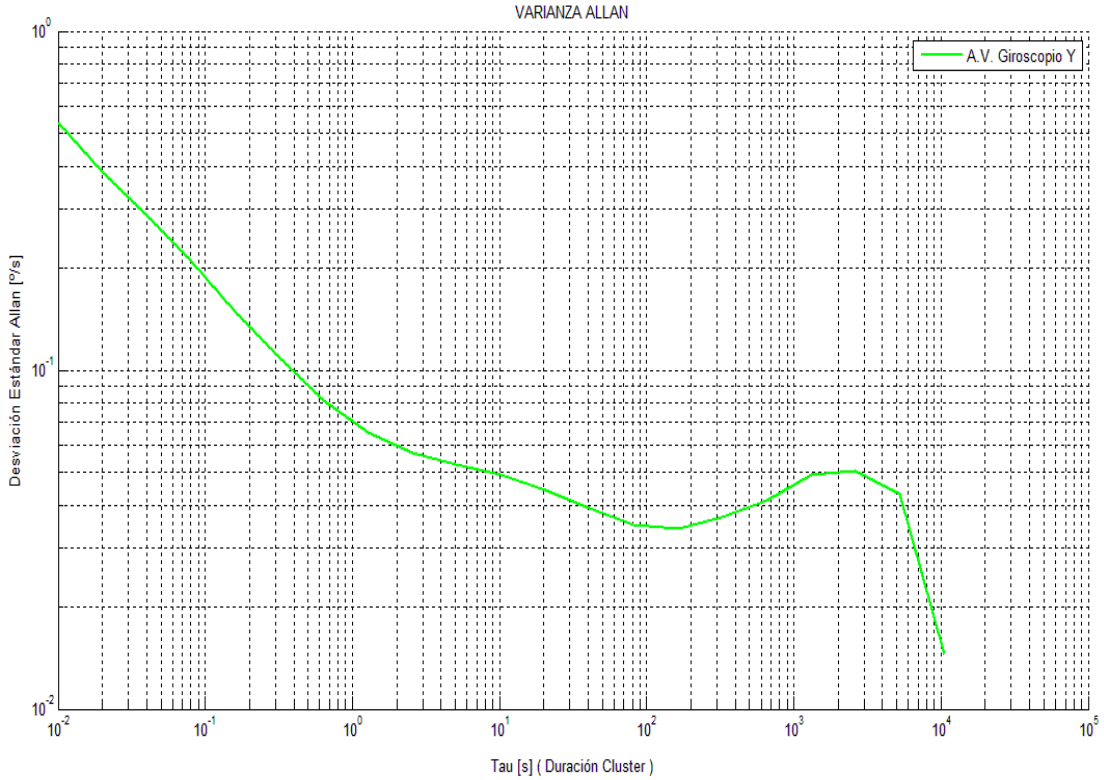
- Resultados del análisis de Allan Variance:

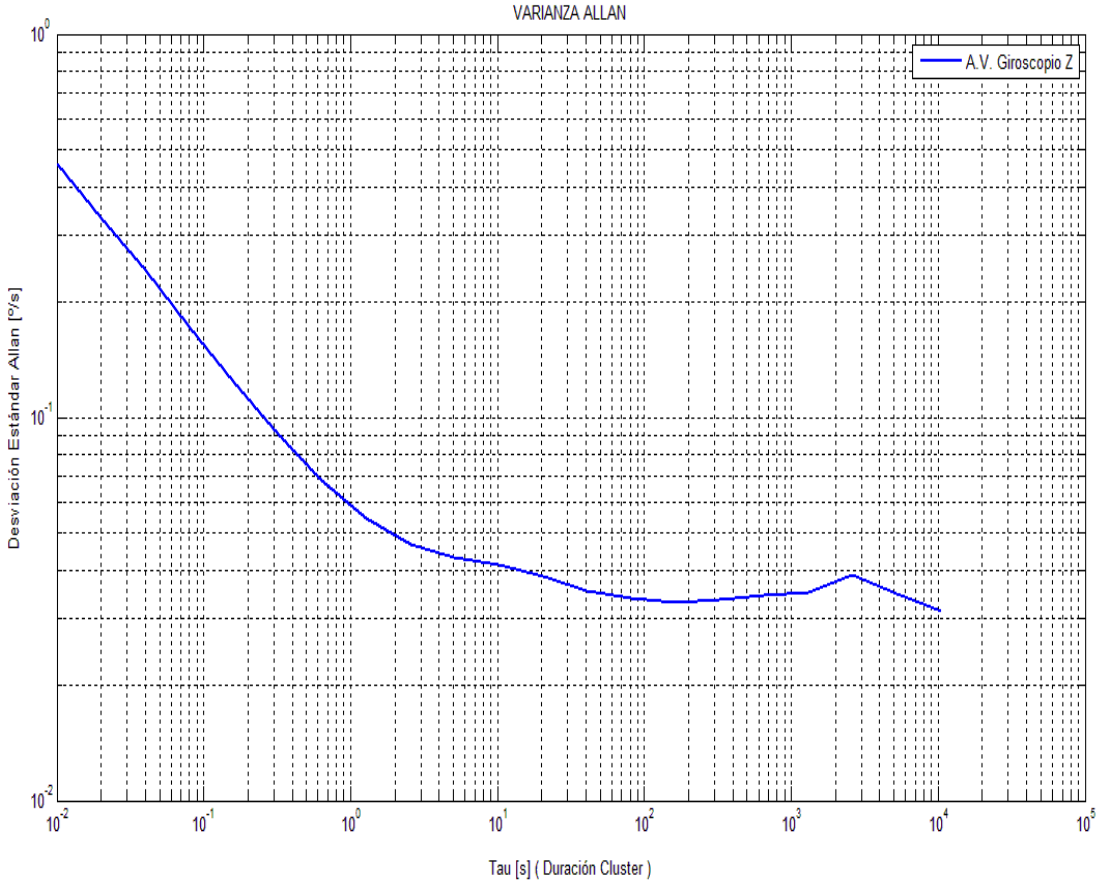




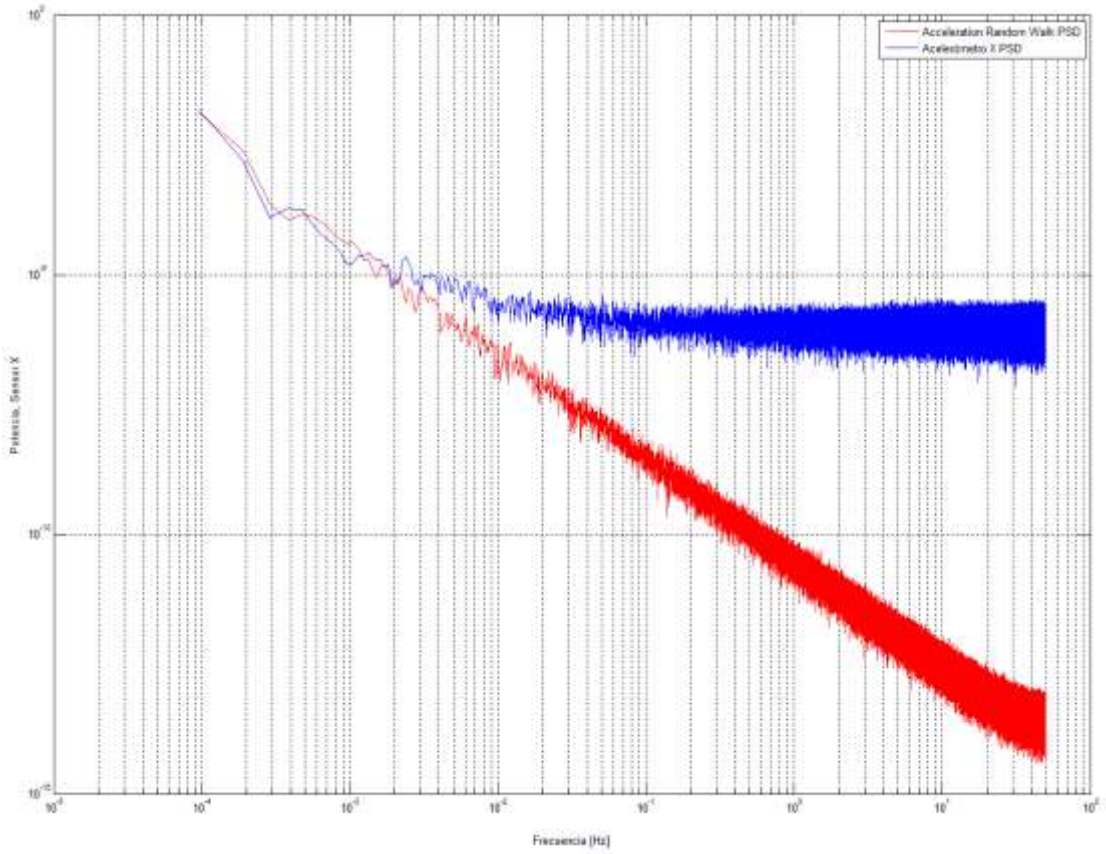

VARIANZA ALLAN
A.V. Acelerómetro Z
Desviación Estándar Allan [g]
Tau [s] (Duración Cluster)

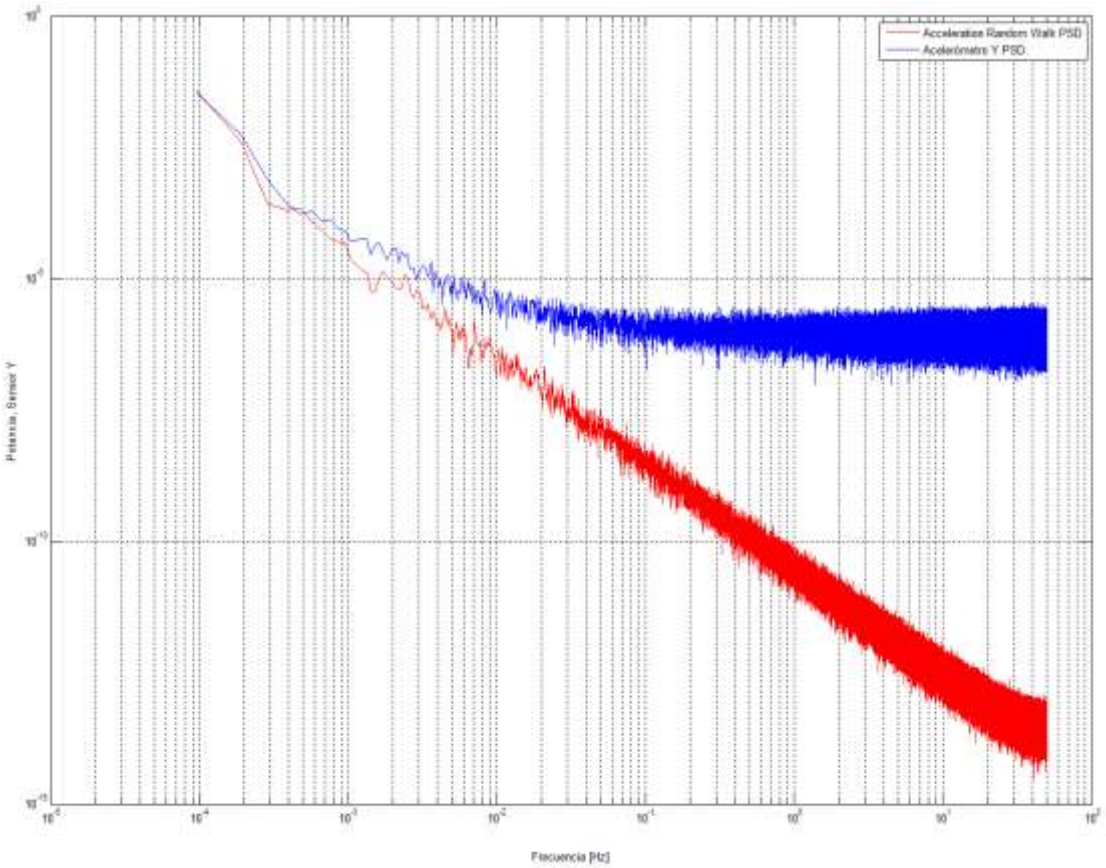

VARIANZA ALLAN
A.V. Giroscopio X
Desviación Estándar Allan [°/s]
Tau [s] (Duración Cluster)

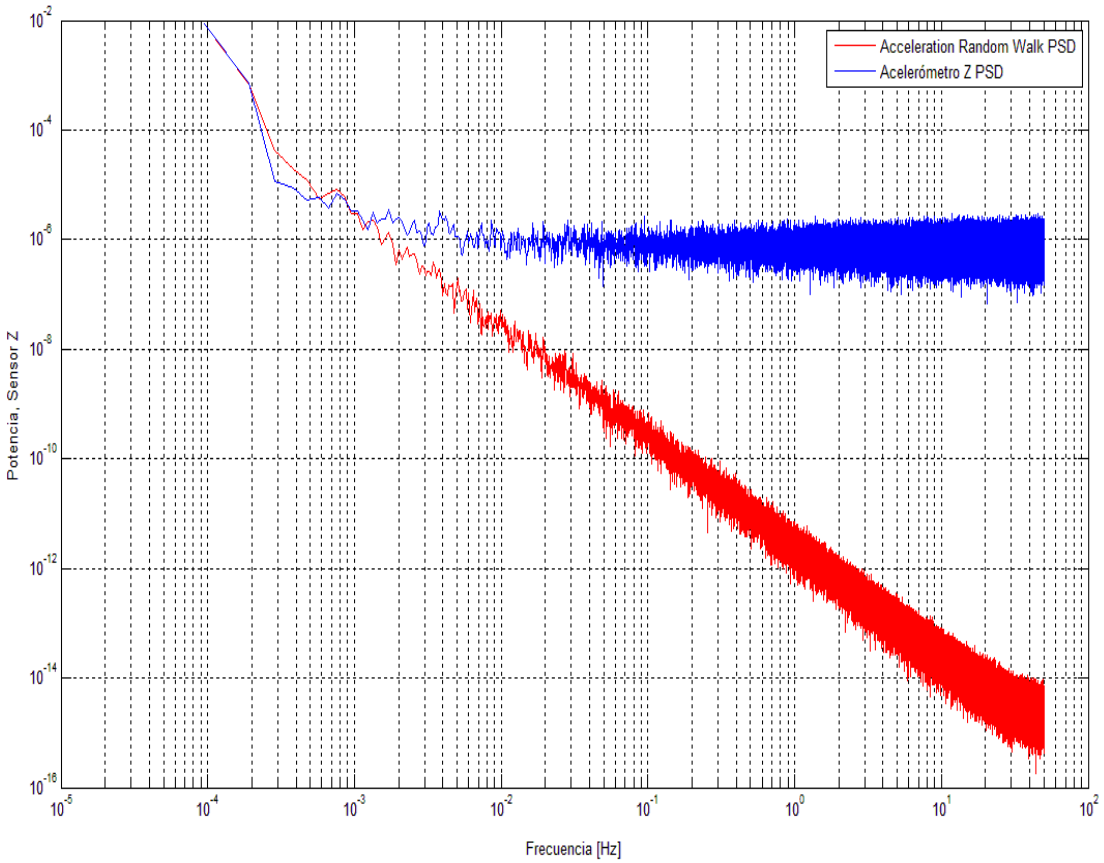

VARIANZA ALLAN
A.V. Giroscopio Y
Desviación Estándar Allan [°/s]
Tau [s] (Duración Cluster)


VARIANZA ALLAN
A.V. Giroscopio Z
Desviación Estándar Allan [°/s]
Tau [s] (Duración Cluster)

- Comparativa PSD de los sensores y PDS del modelo ARW:

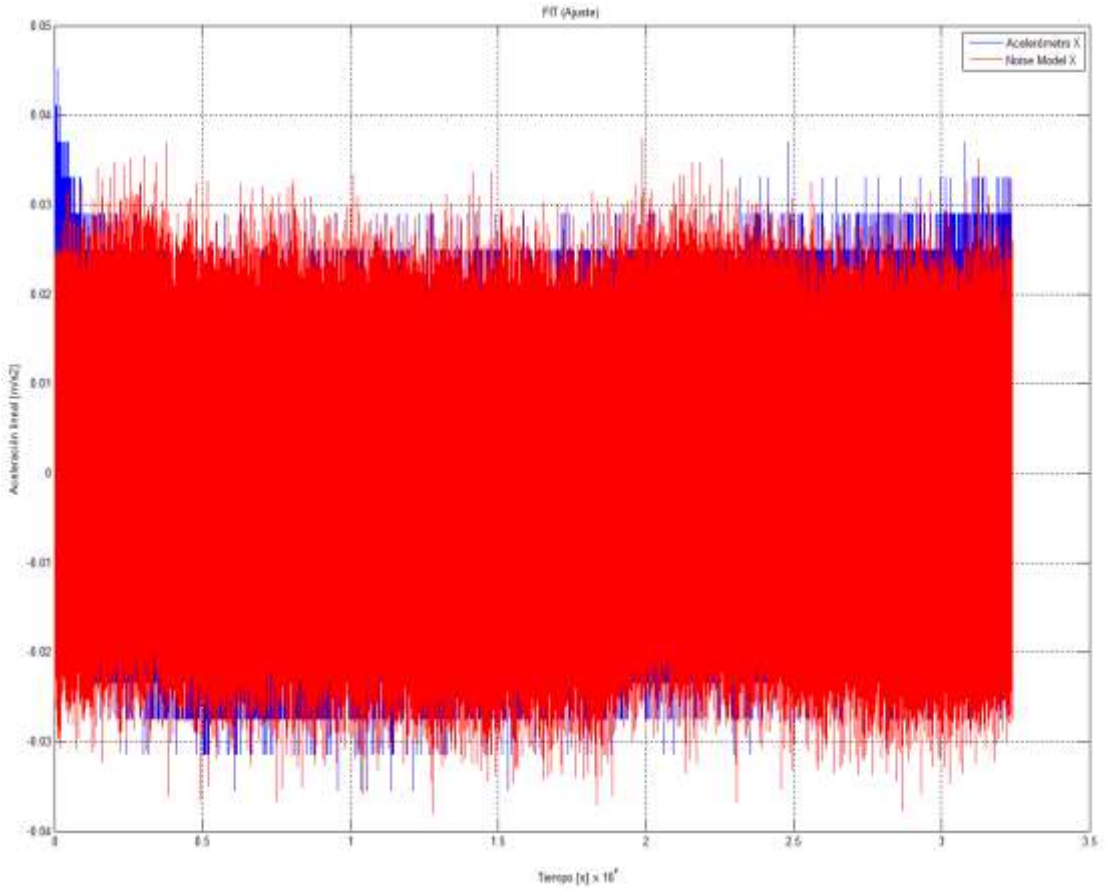


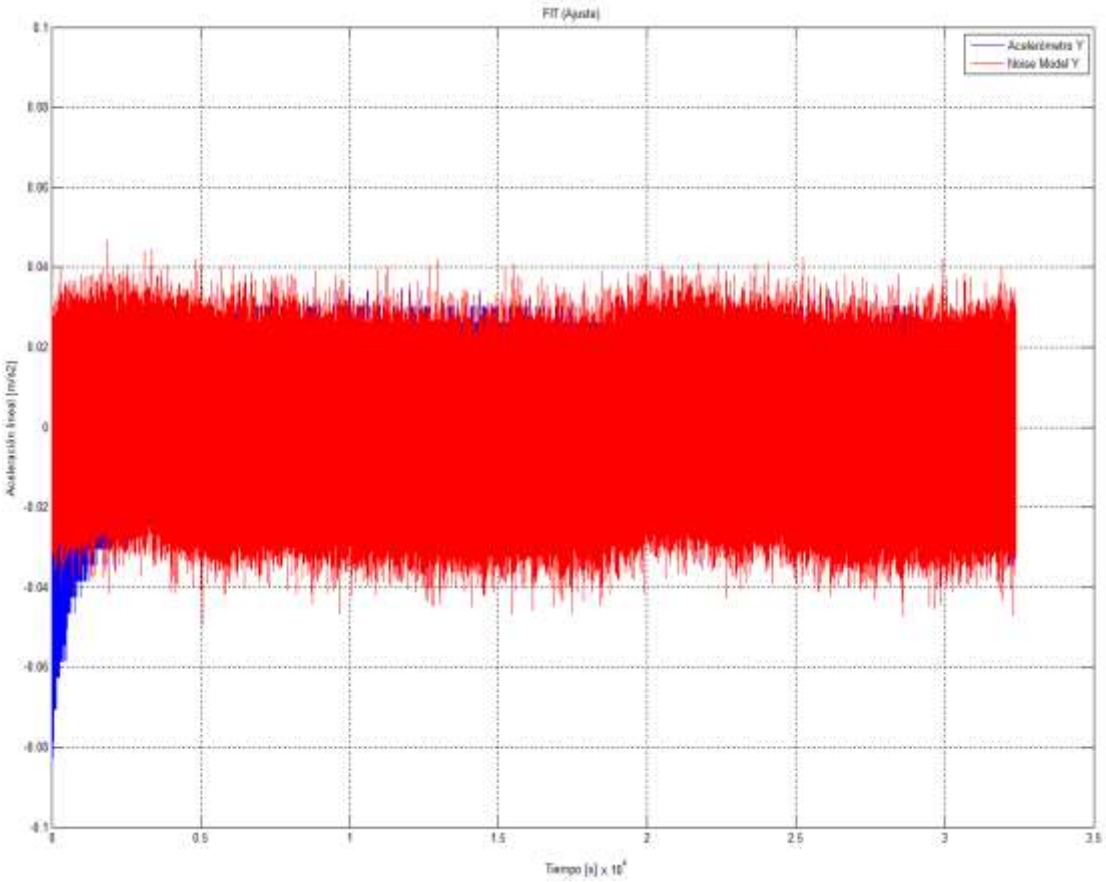


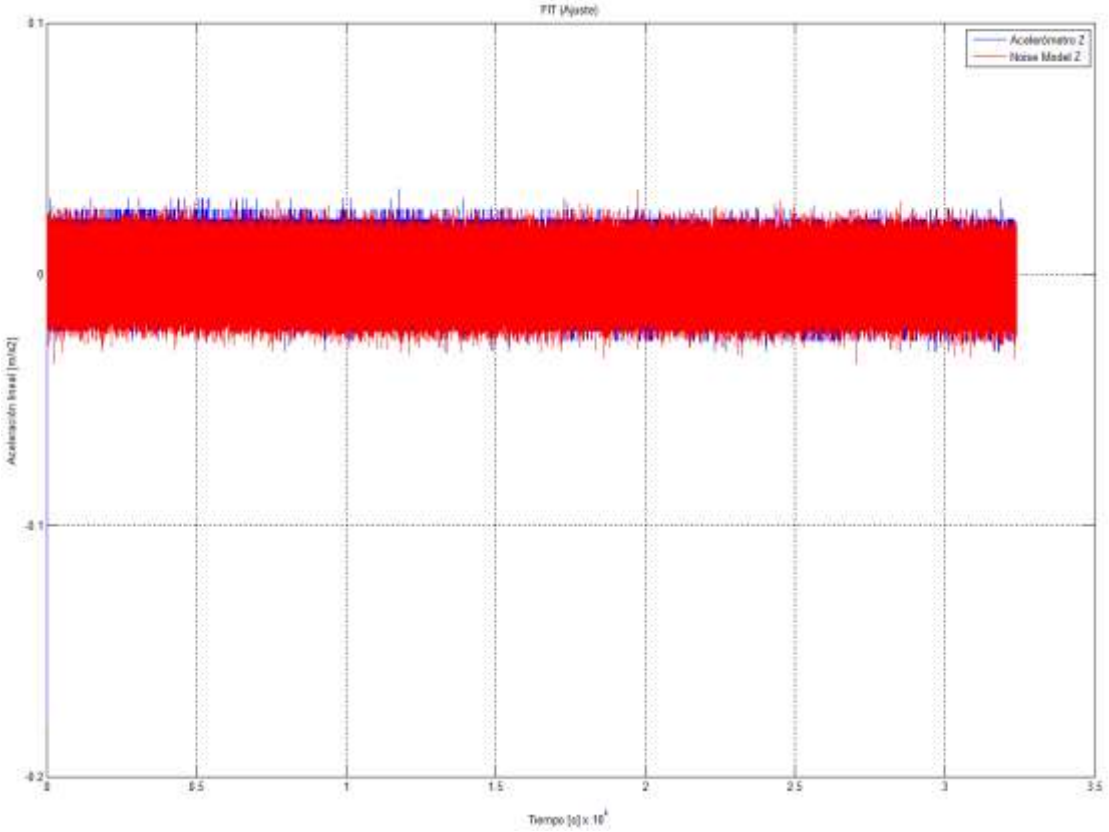


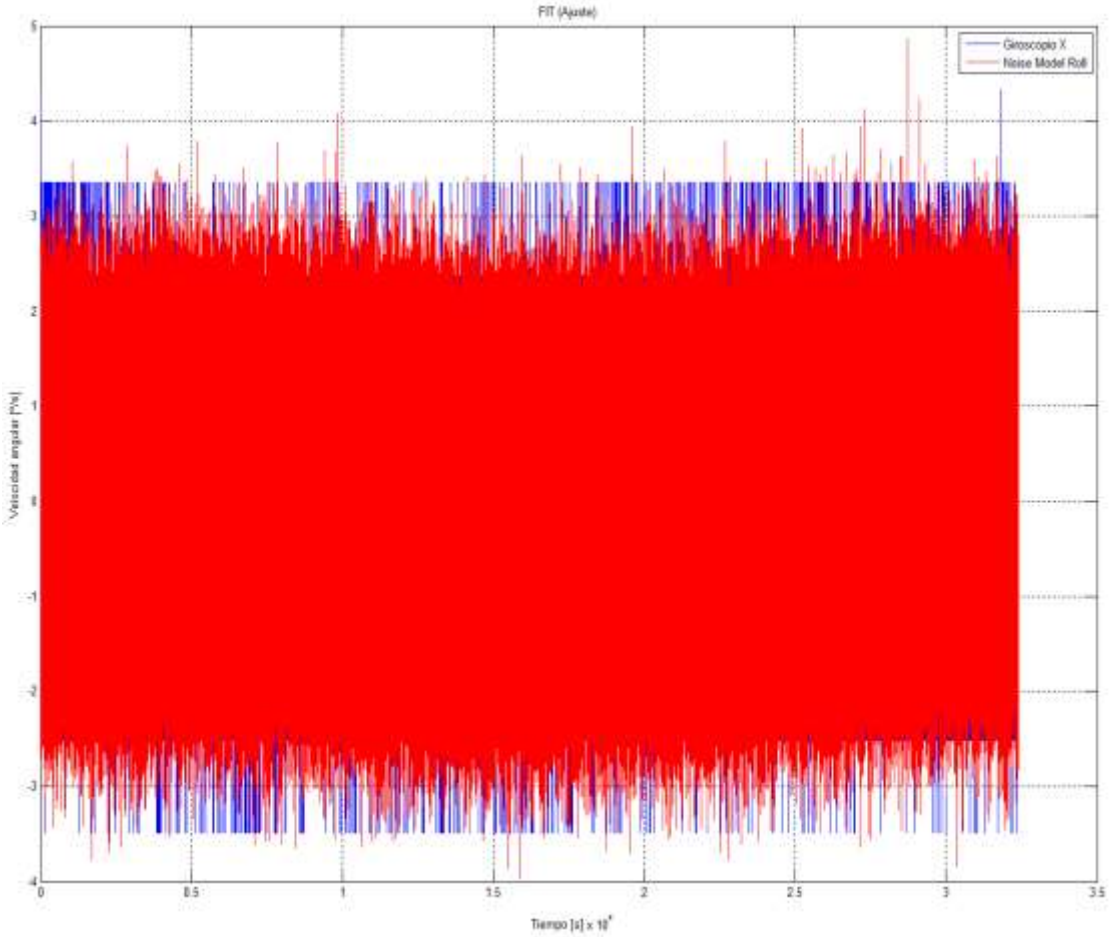
No aparece la contribución de K, Acceleration/Rate Random Walk, para los giroscopios.

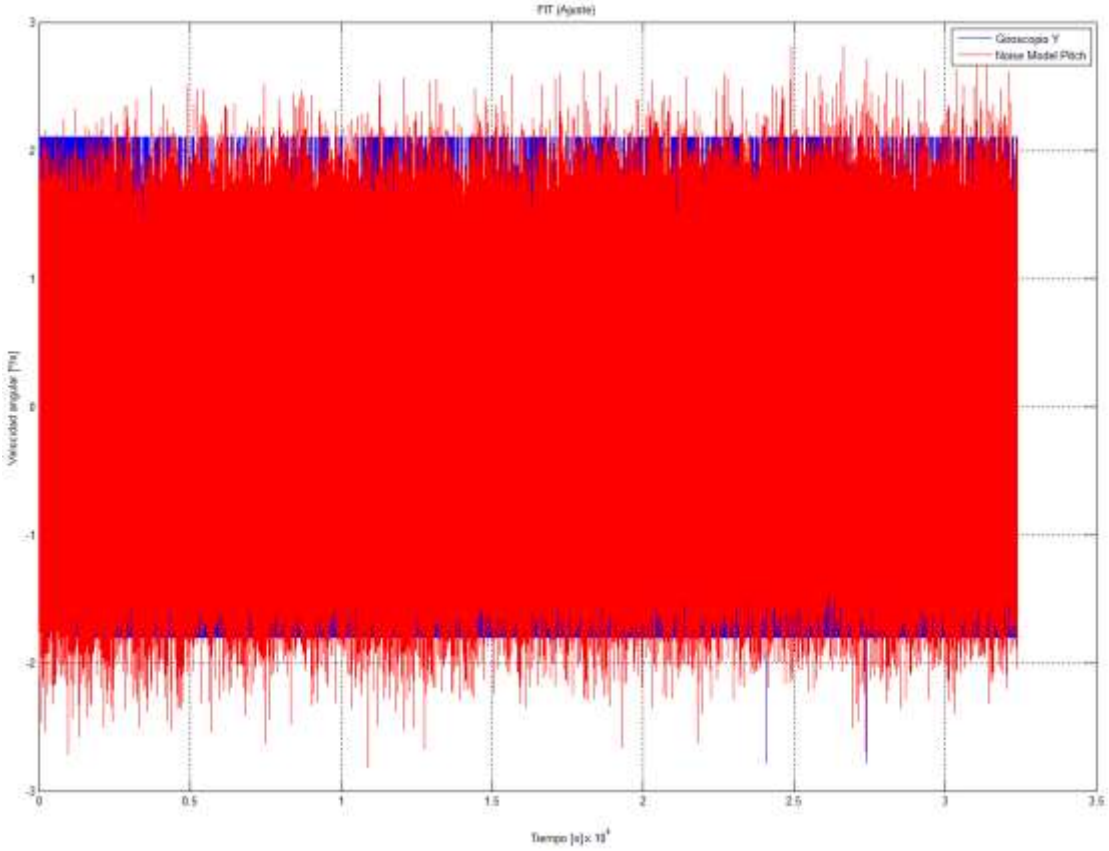
- Modelos FIT obtenidos:

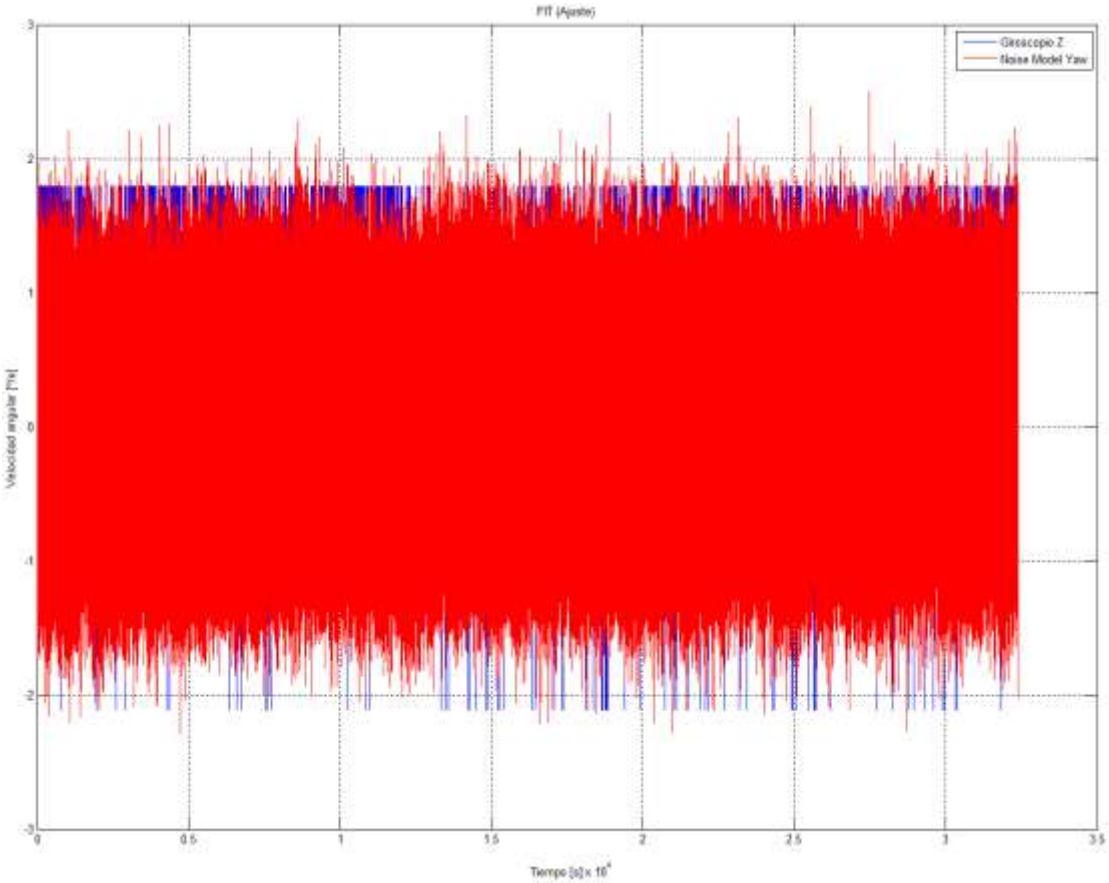





FIT (Ajuste)
Acelerómetro Z
Noise Model Z
Aceleración lineal [m/s2]
Tiempo [s] x 10^4

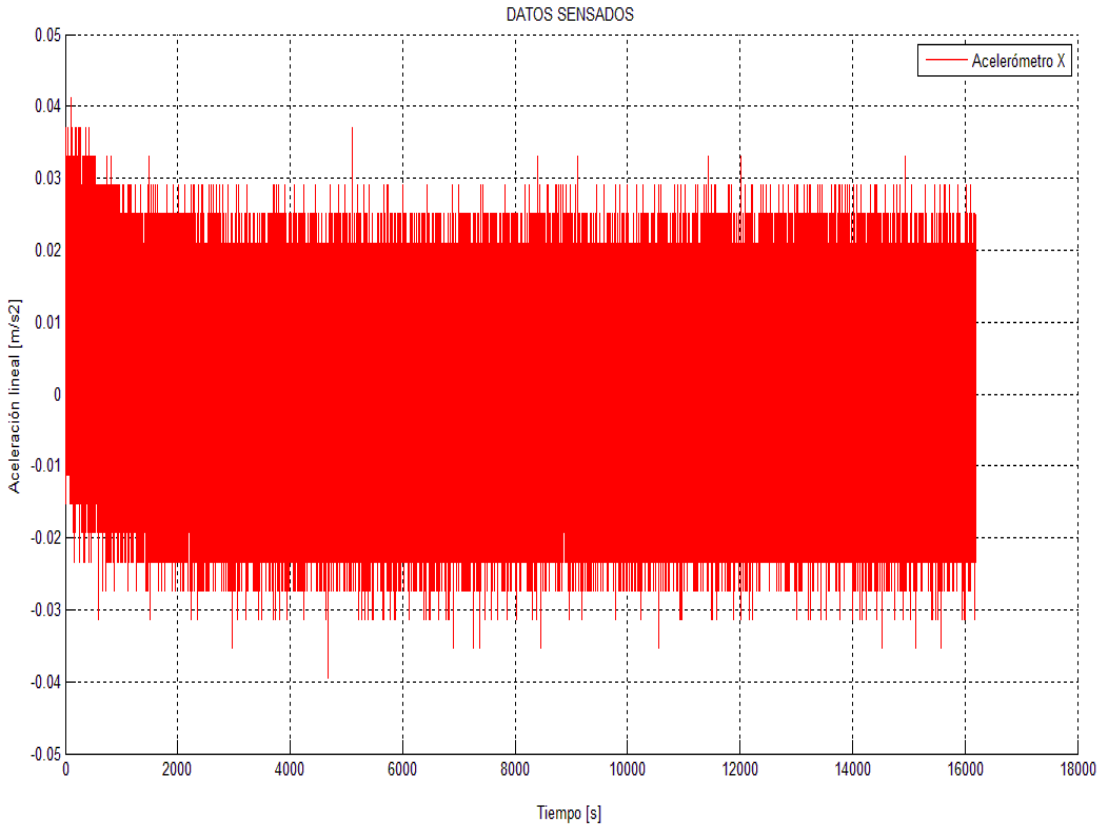

FIT (Ajuste)
Giroscopio X
Noise Model Roll
Velocidad angular [°/s]
Tiempo [s] x 10^4

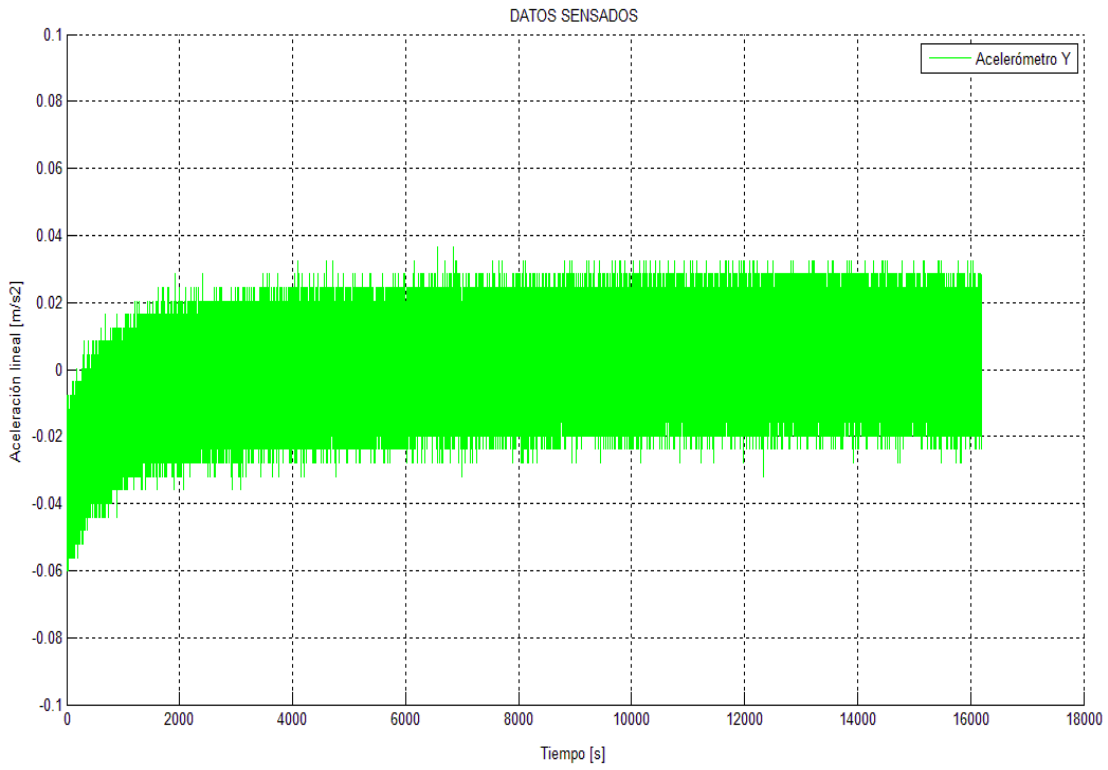

FIT (Ajuste)
Giroscopio Y
Nose Model Pitch
Velocidad angular [°/s]
Tiempo [s] x 10^4

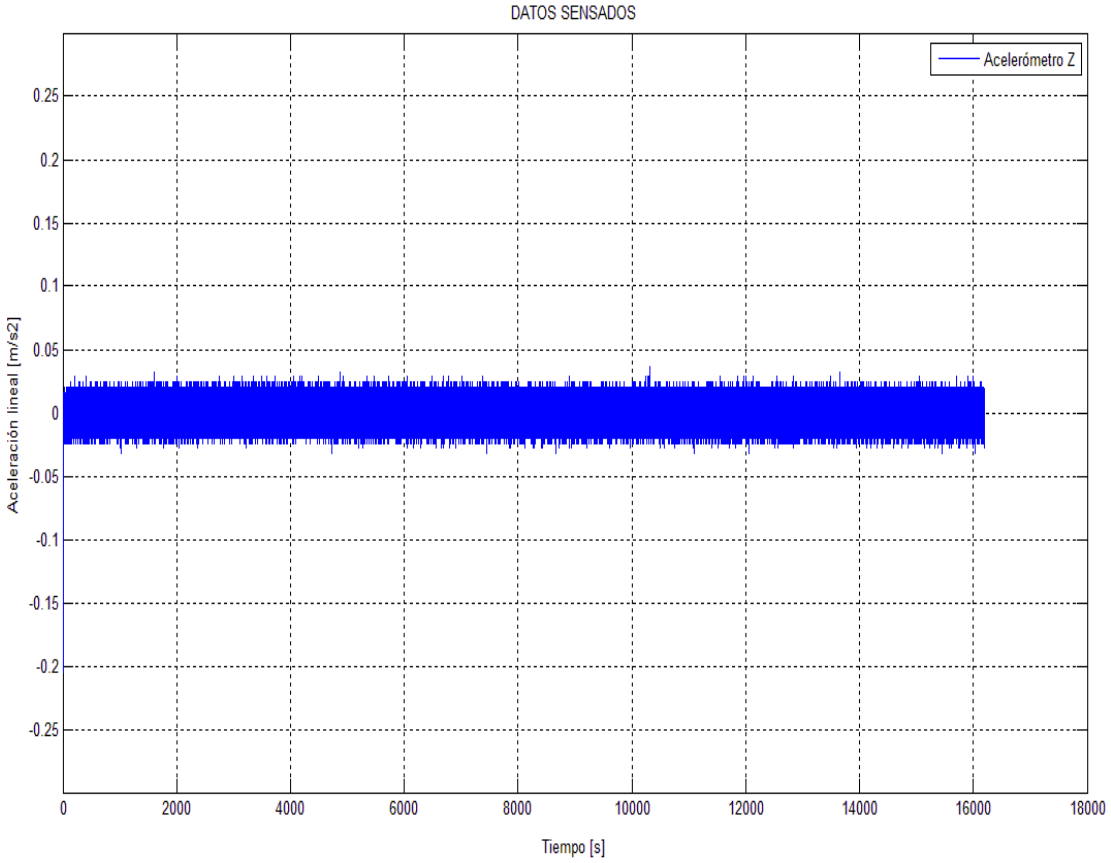

FIT (Ajuste)
Giroscopio Z
Nose Model Yaw
Velocidad angular [°/s]
Tiempo [s] x 10^4

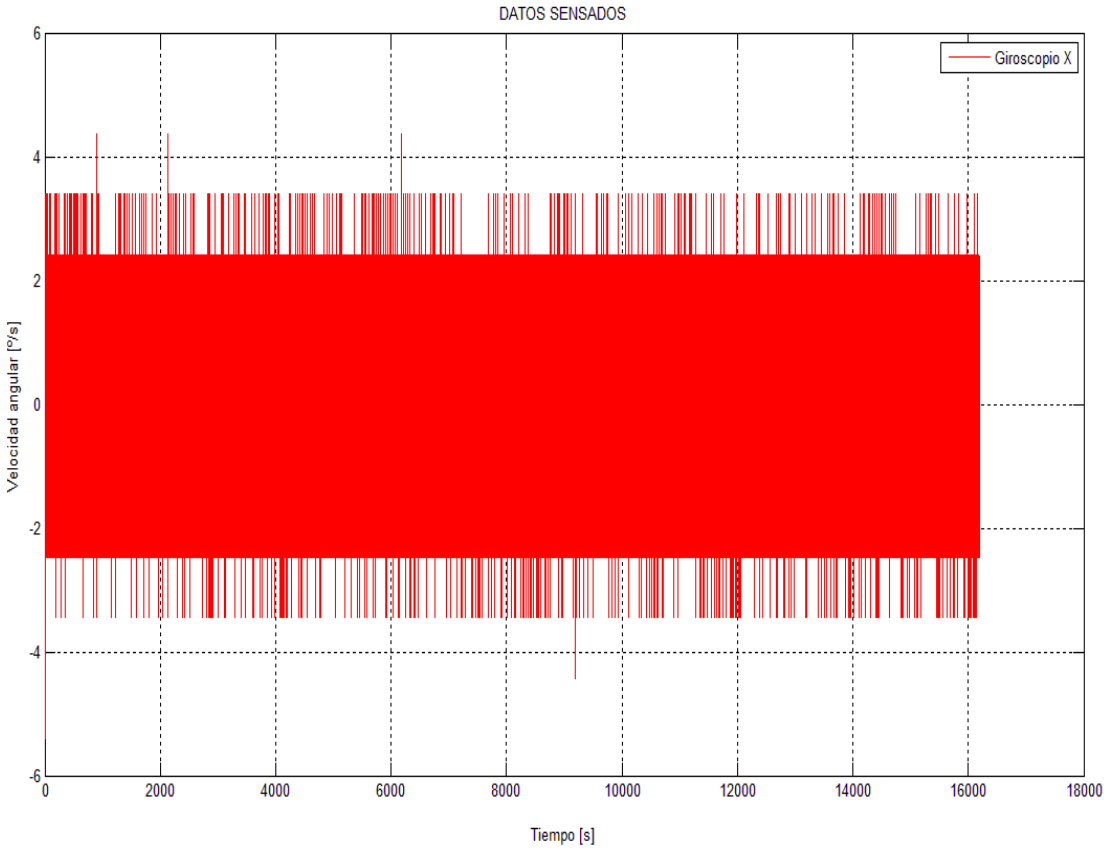
Fs = 200Hz

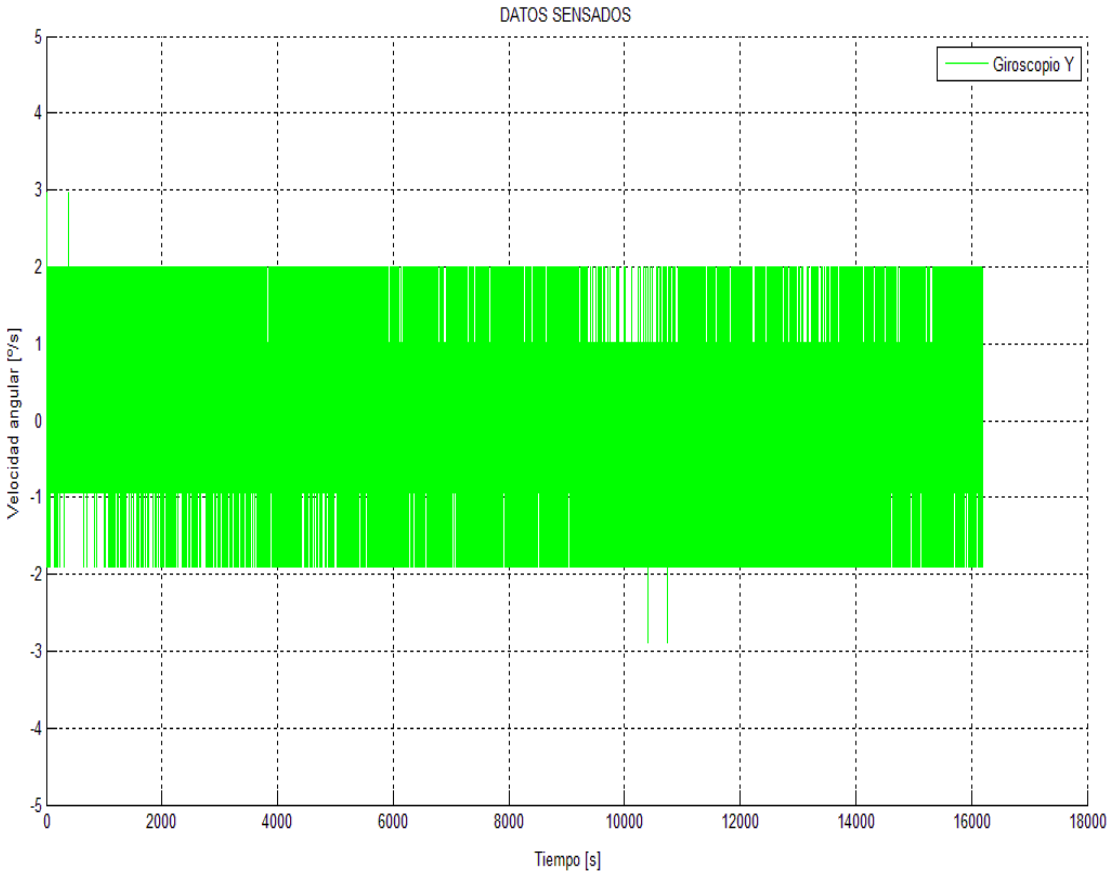
- Lecturas de los sensores:

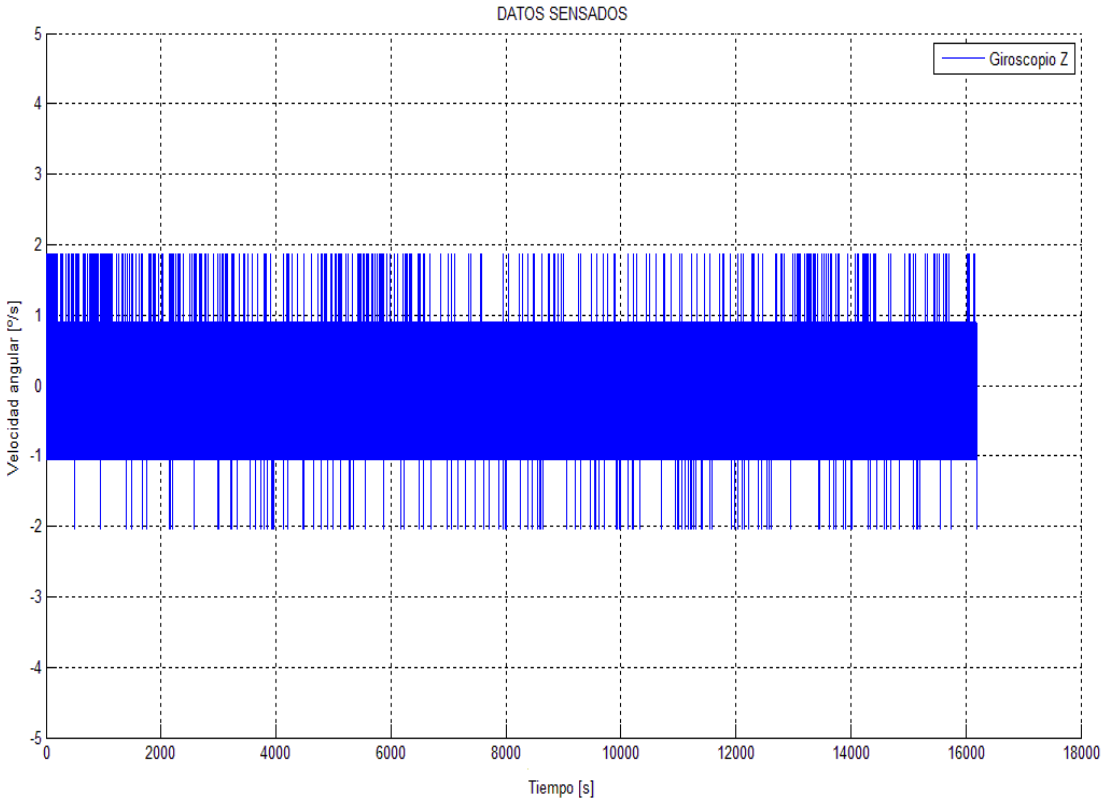




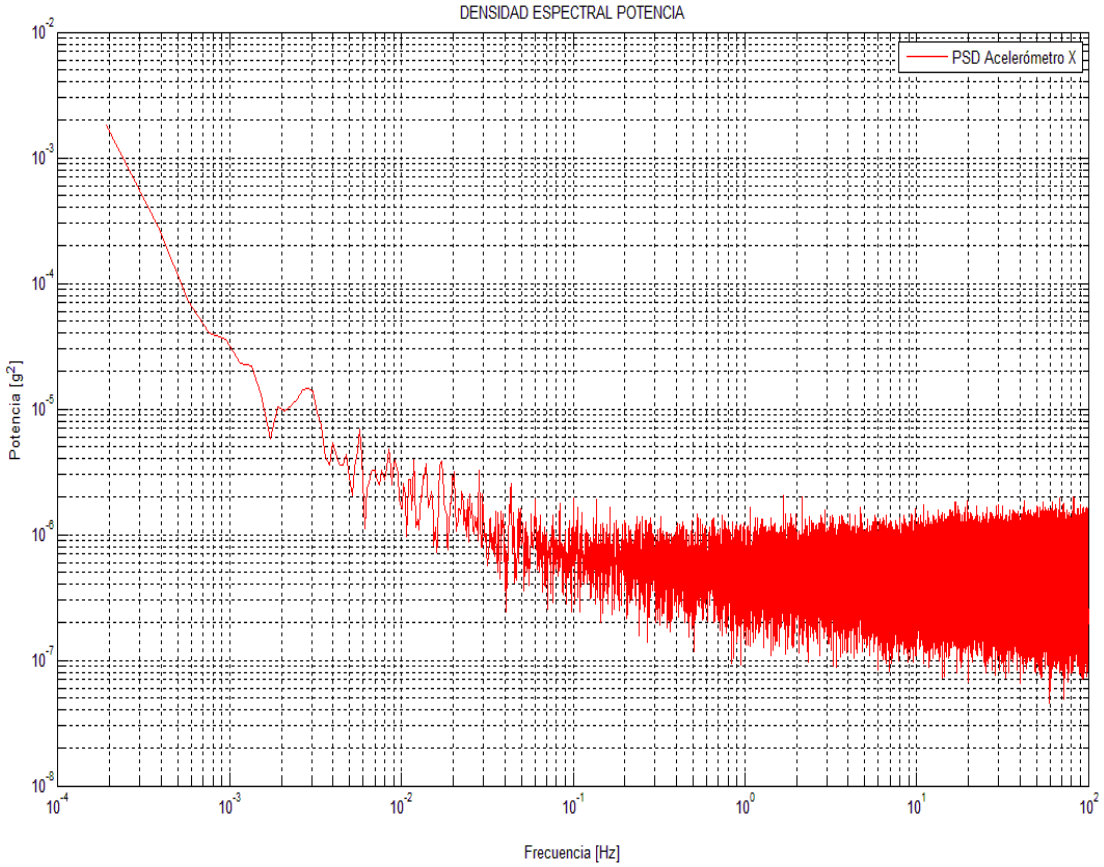

DATOS SENSADOS
Acelerómetro Z
Aceleración lineal [m/s2]
Tiempo [s]
0.25
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
0
2000
4000
6000
8000
10000
12000
14000
16000
18000

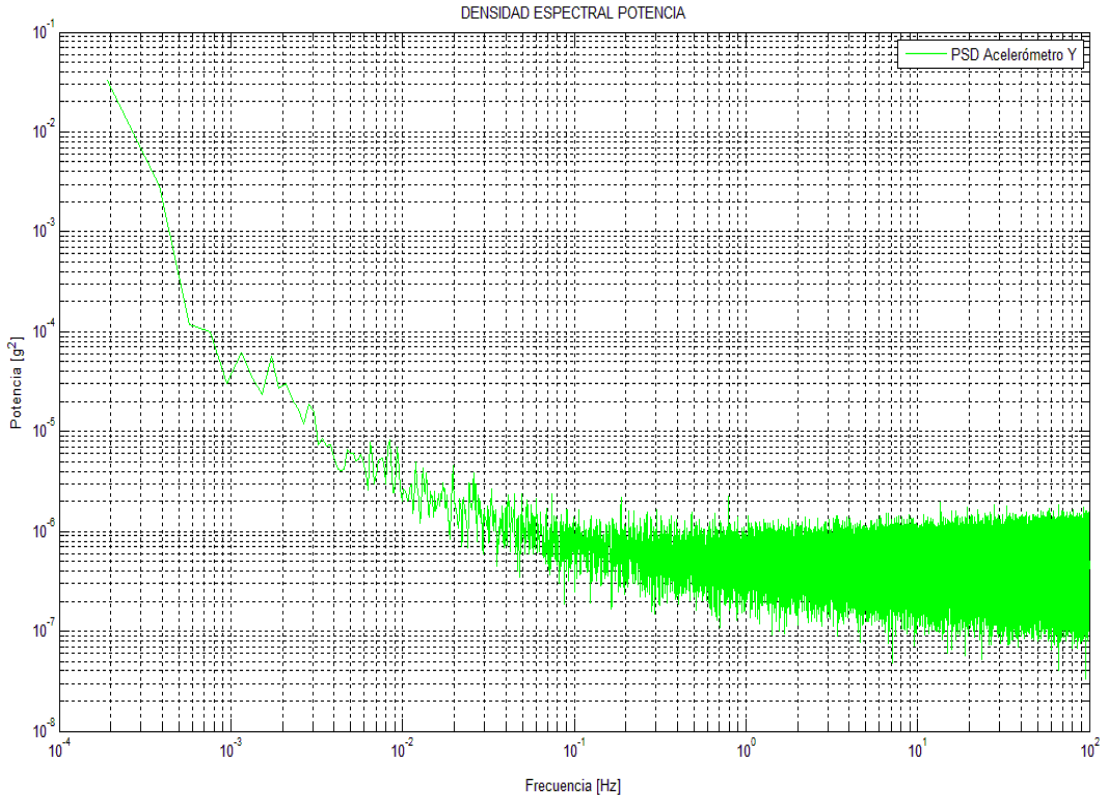

DATOS SENSADOS
Giroscopio X
Velocidad angular [º/s]
Tiempo [s]
6
4
2
0
-2
-4
-6
0
2000
4000
6000
8000
10000
12000
14000
16000
18000

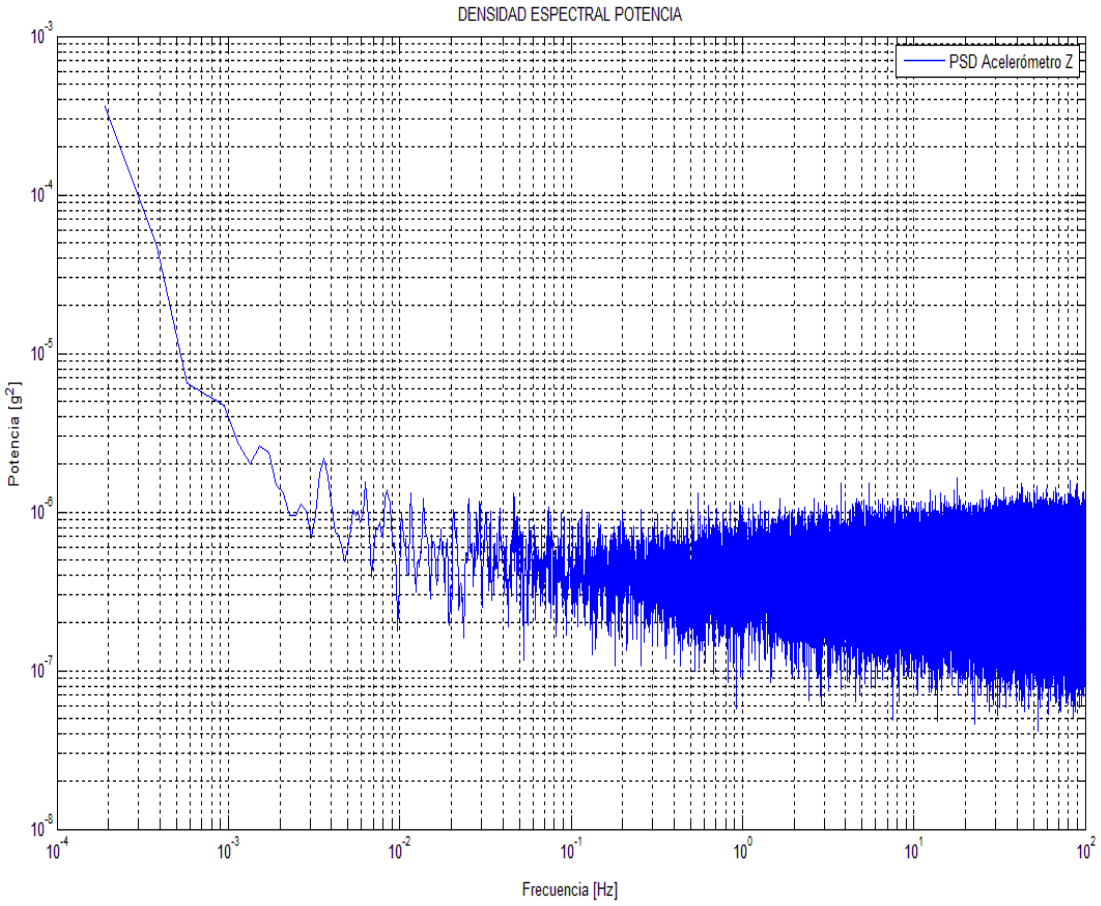

DATOS SENSADOS
Giroscopio Y
Velocidad angular [°/s]
Tiempo [s]
5
4
3
2
1
0
-1
-2
-3
-4
-5
0
2000
4000
6000
8000
10000
12000
14000
16000
18000

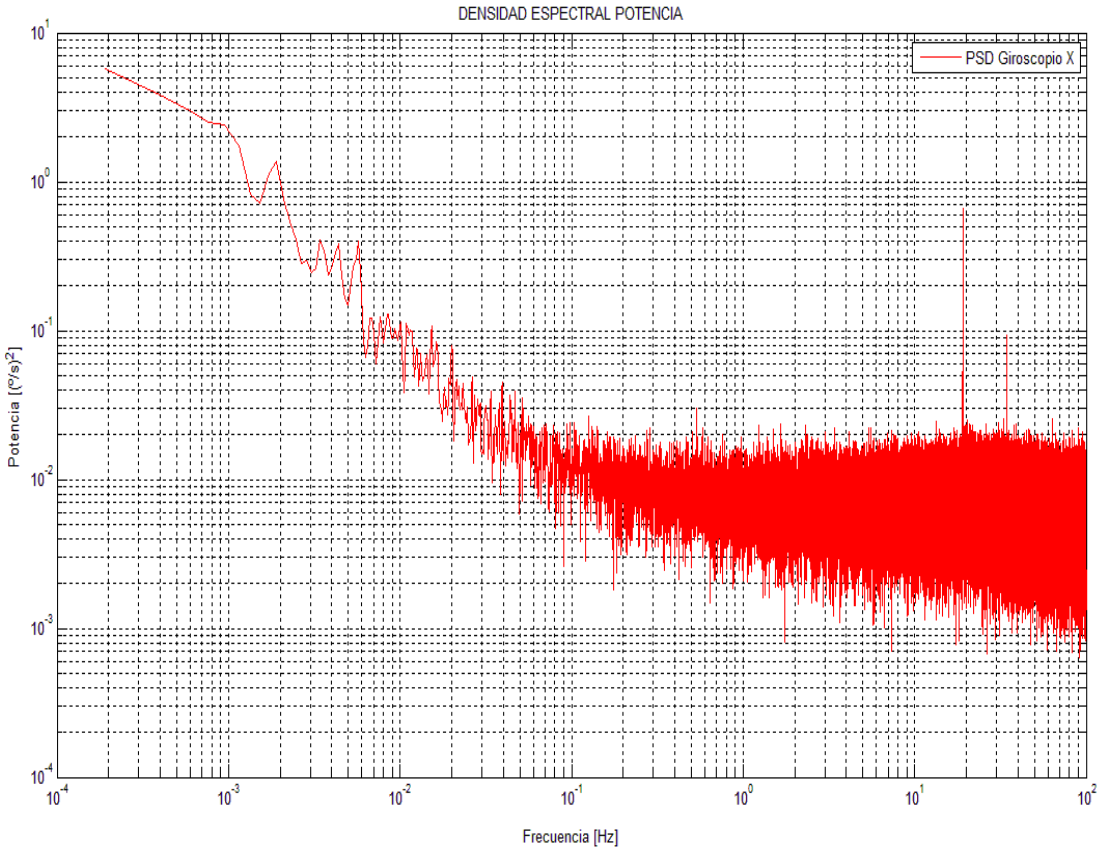

DATOS SENSADOS
Giroscopio Z
Velocidad angular [°/s]
Tiempo [s]
5
4
3
2
1
0
-1
-2
-3
-4
-5
0
2000
4000
6000
8000
10000
12000
14000
16000
18000

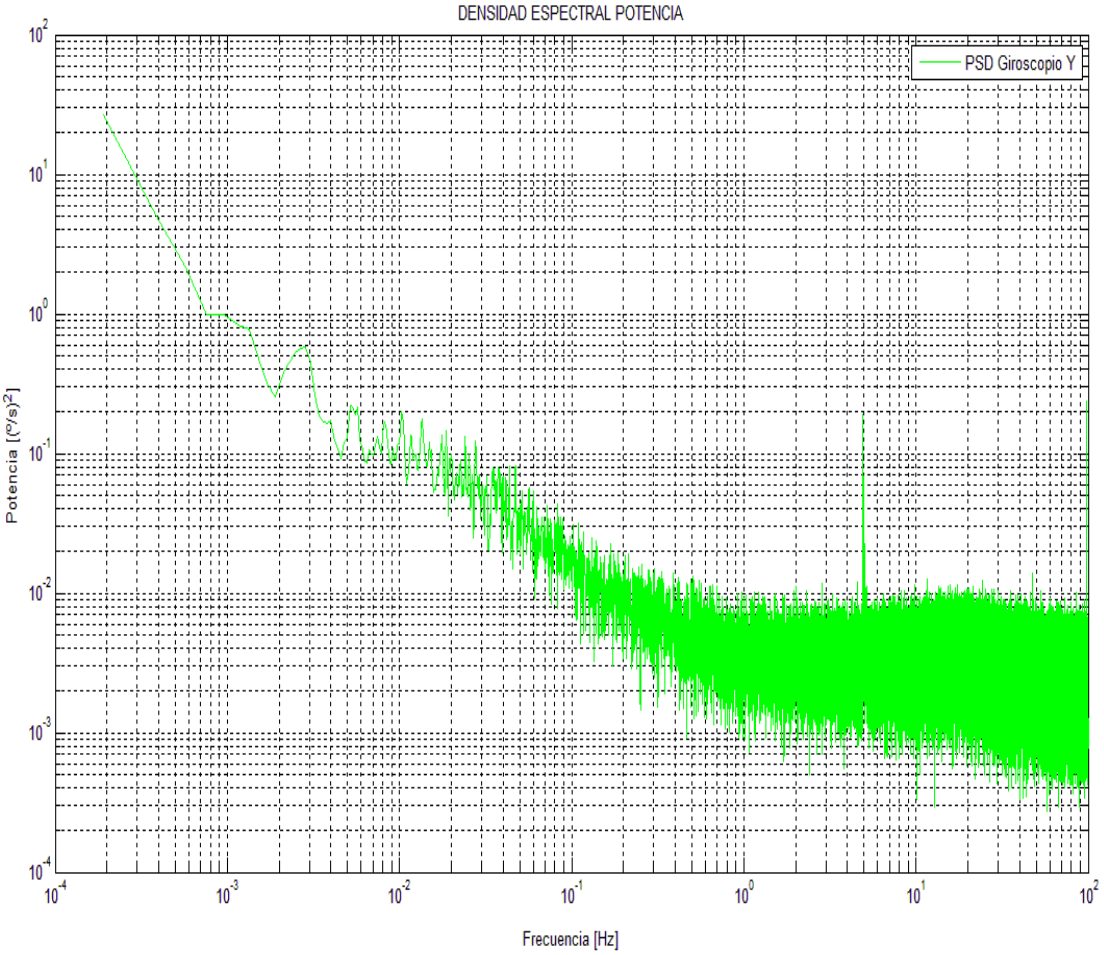
- Densidad espectral de potencia de los sensores:

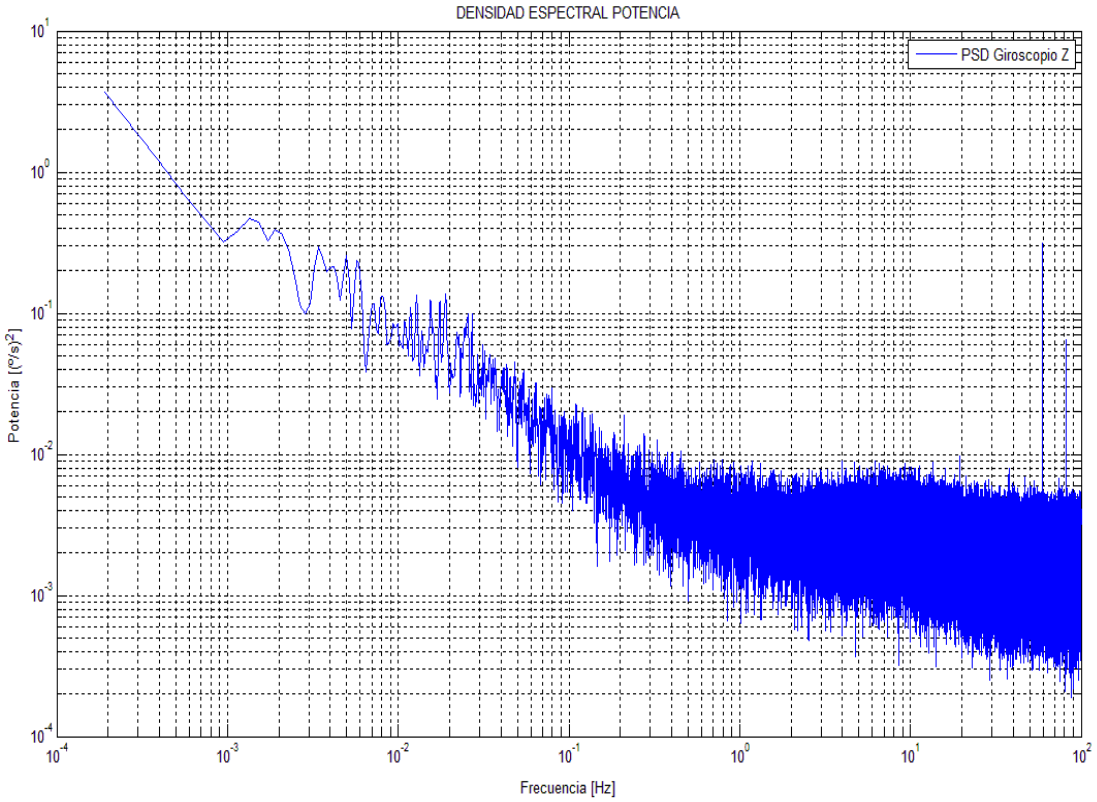




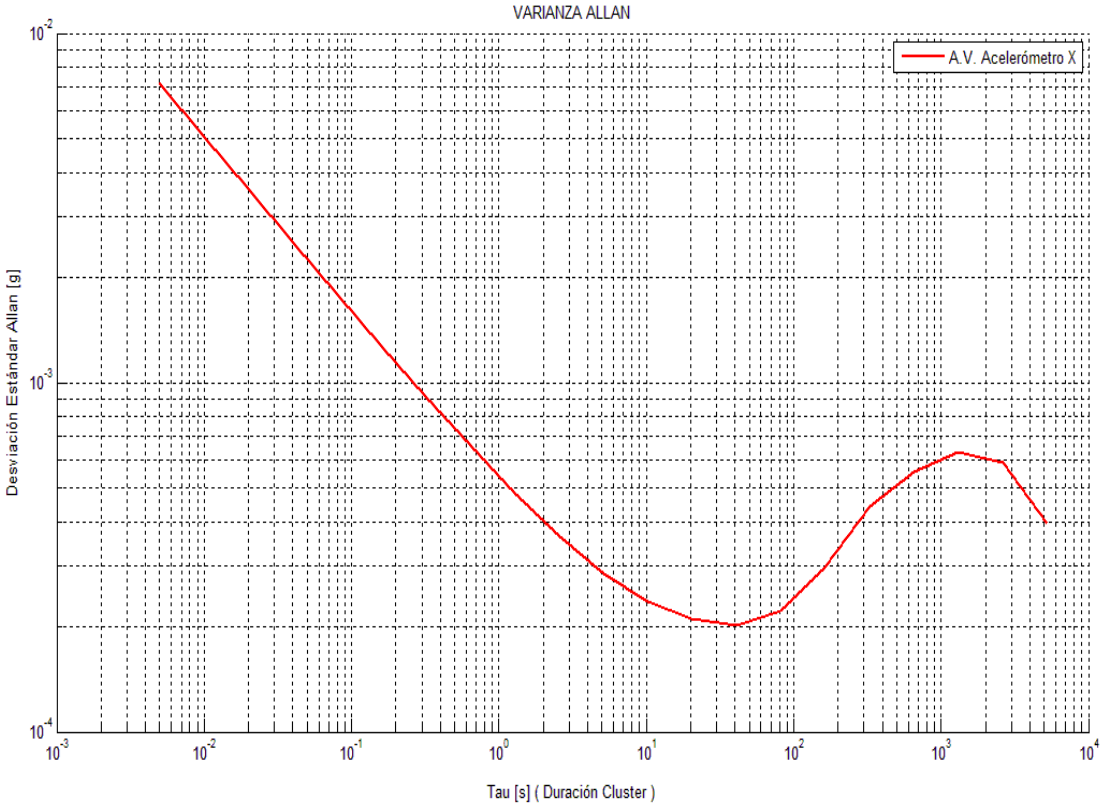

DENSIDAD ESPECTRAL POTENCIA
PSD Acelerómetro Z
Potencia [g²]
Frecuencia [Hz]

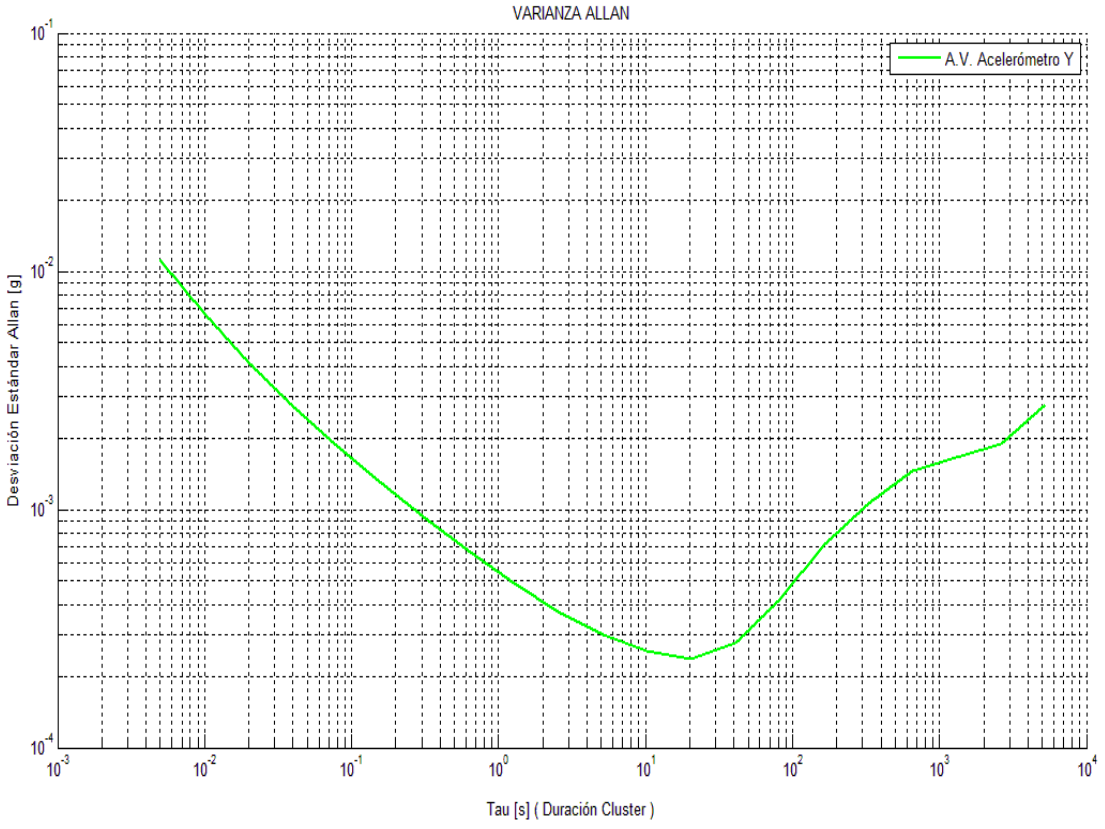

DENSIDAD ESPECTRAL POTENCIA
PSD Giroscopio X
Potencia [(°/s)²]
Frecuencia [Hz]

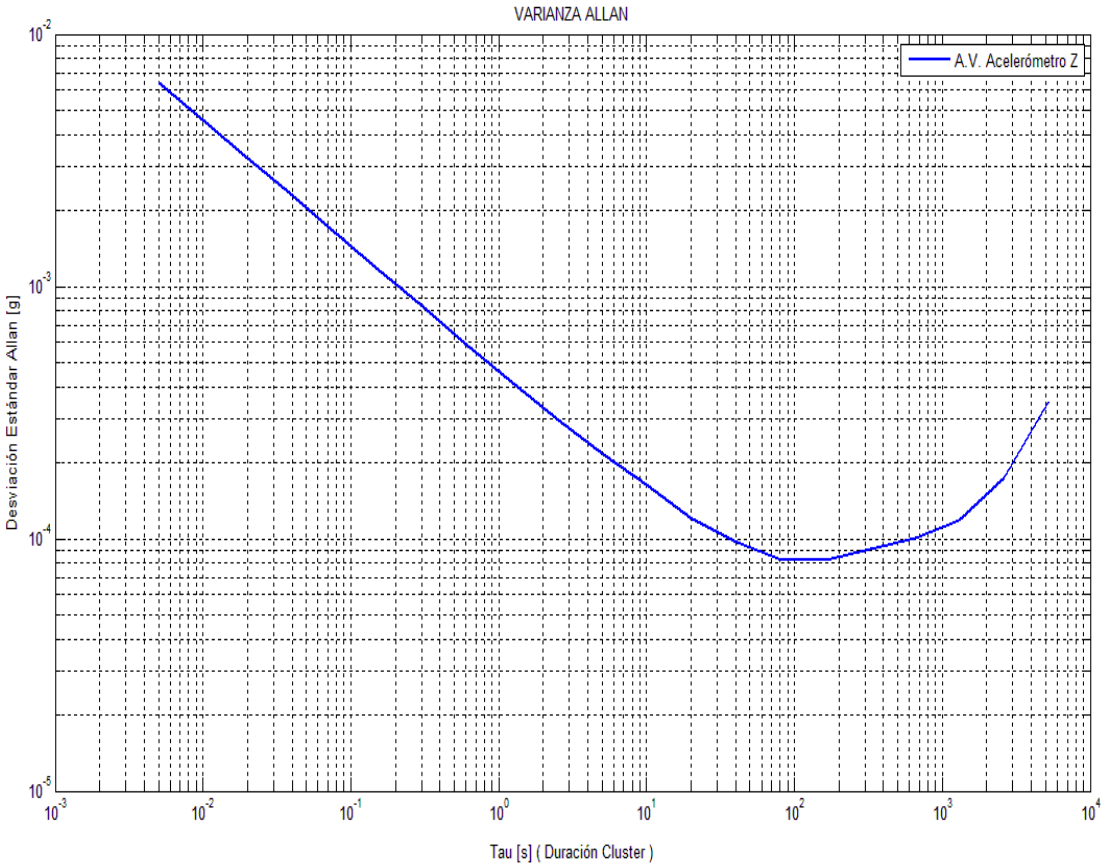

DENSIDAD ESPECTRAL POTENCIA
PSD Giroscopio Y
Potencia [(º/s)²]
Frecuencia [Hz]

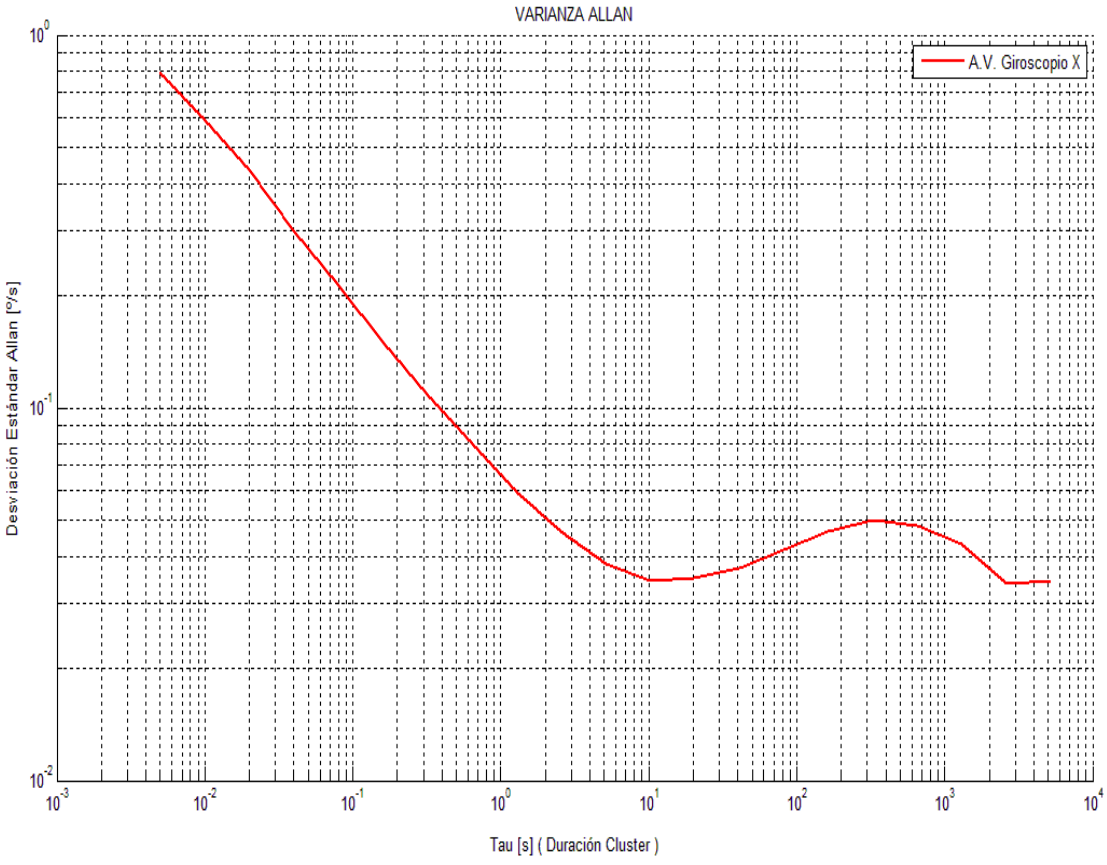

DENSIDAD ESPECTRAL POTENCIA
PSD Giroscopio Z
Potencia [(º/s)²]
Frecuencia [Hz]

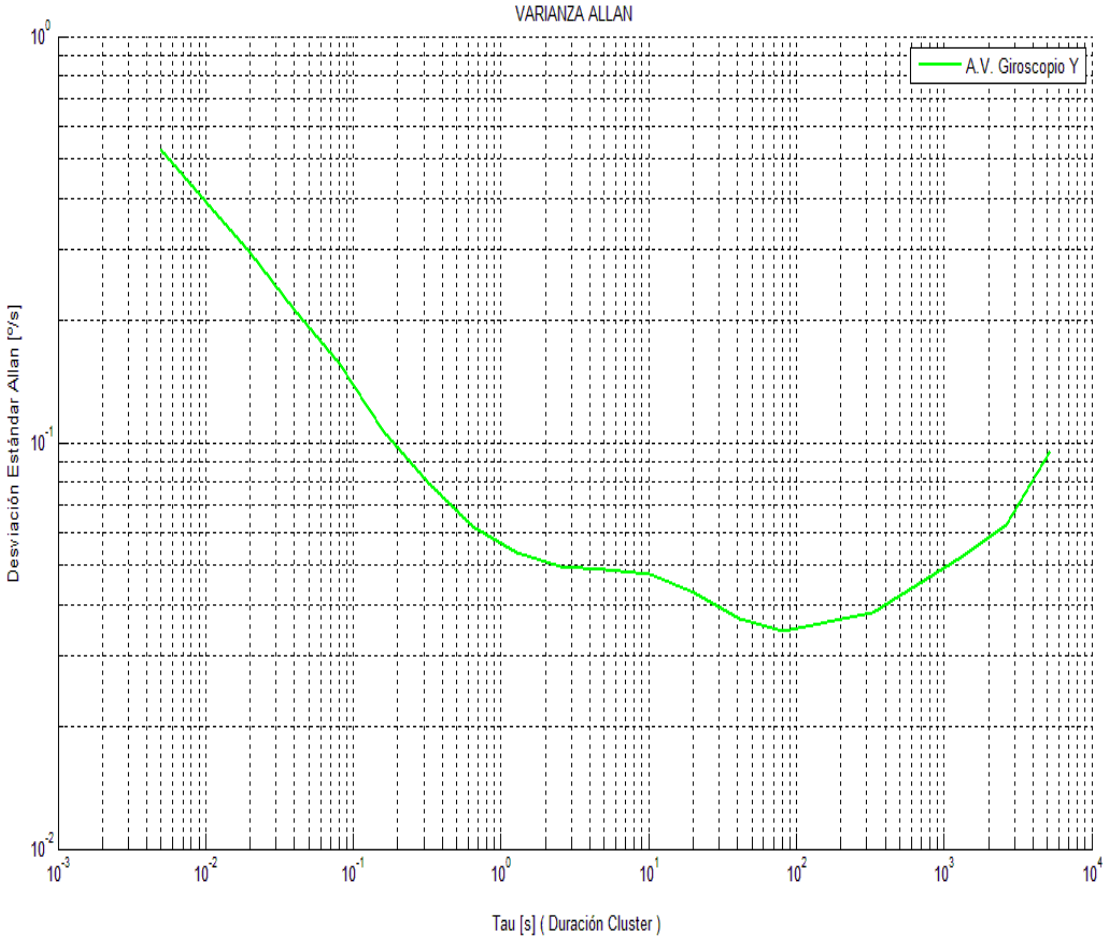
- Resultados del análisis de Allan Variance:

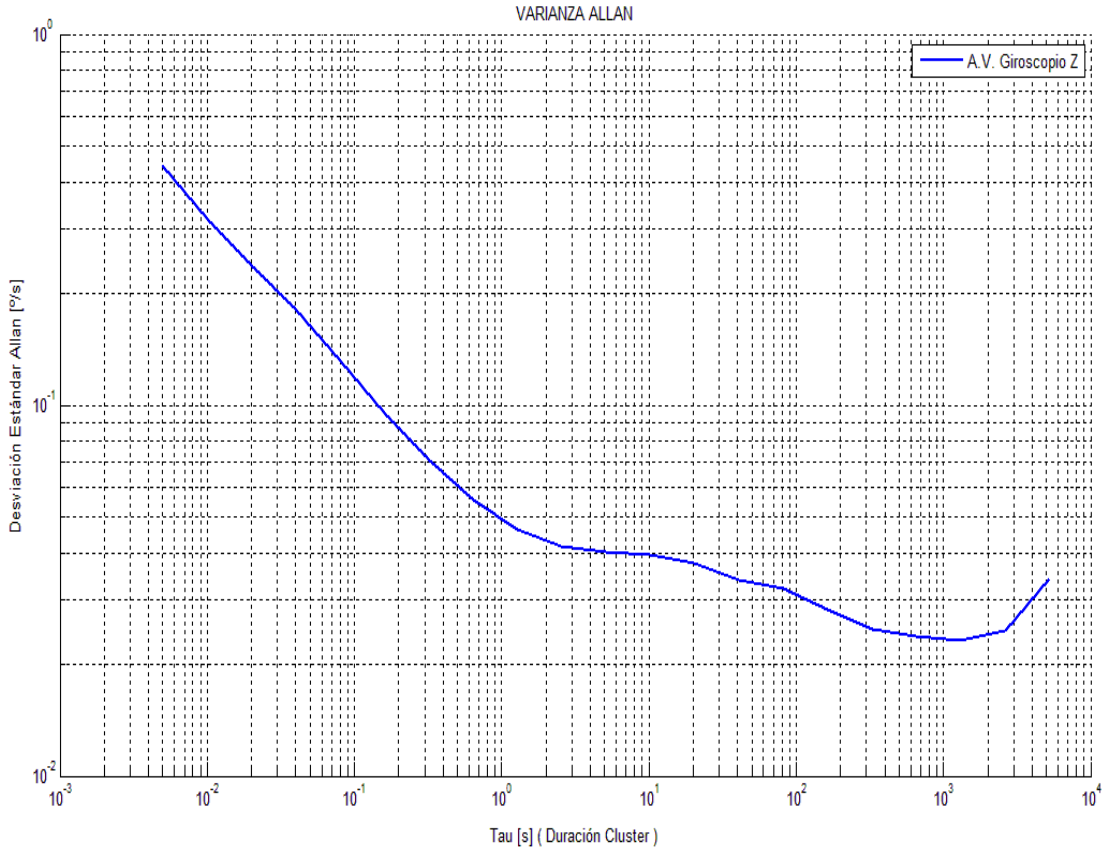




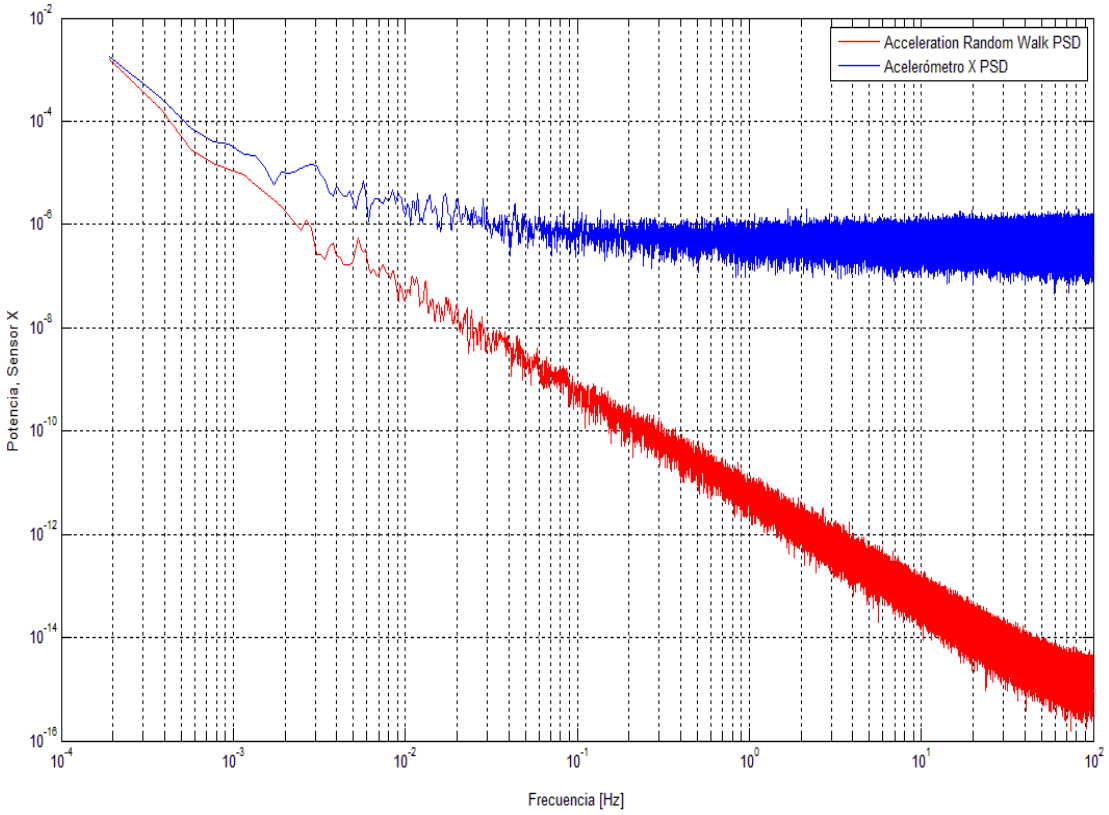

VARIANZA ALLAN
A.V. Acelerómetro Z
Desviación Estándar Allan [g]
Tau [s] (Duración Cluster)

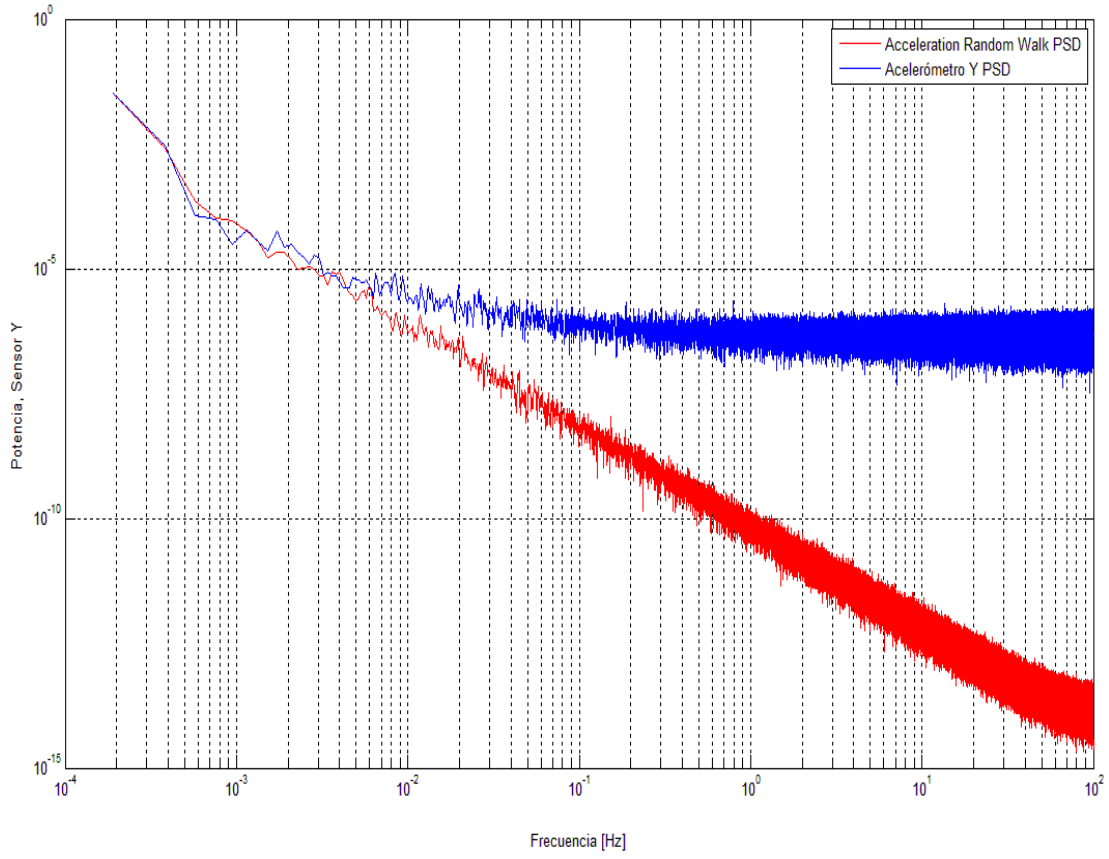

VARIANZA ALLAN
A.V. Giroscopio X
Desviación Estándar Allan [°/s]
Tau [s] (Duración Cluster)

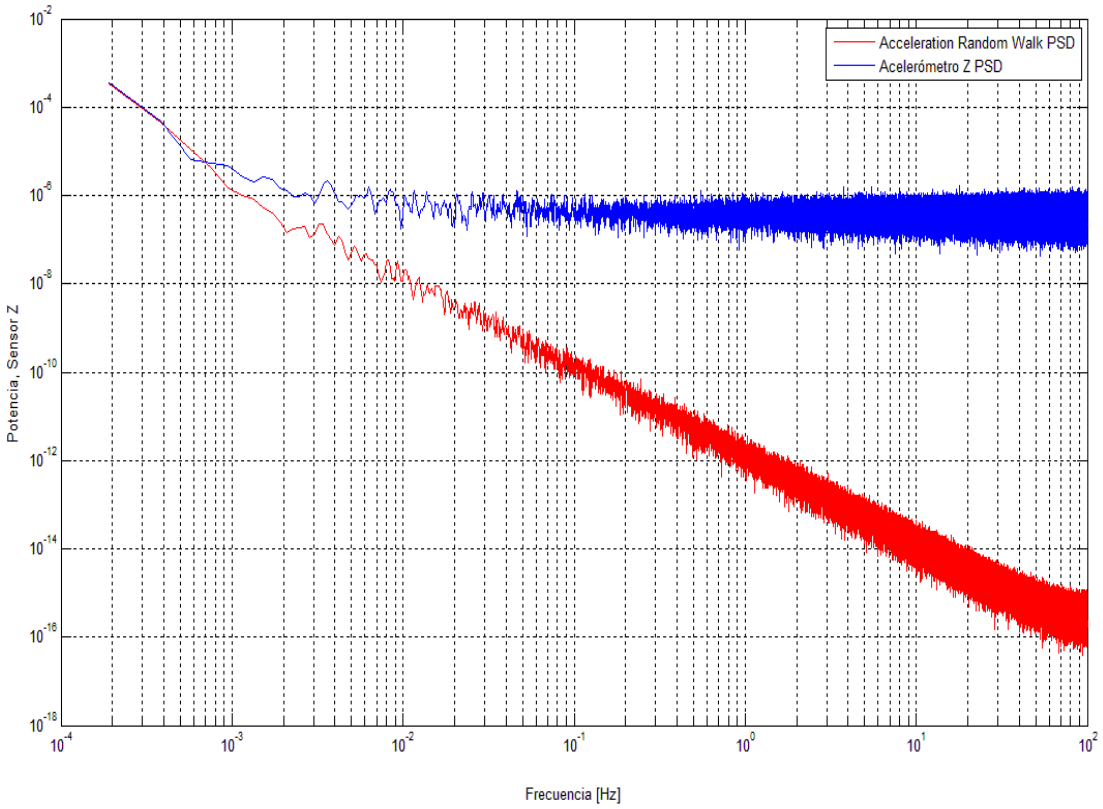

VARIANZA ALLAN
A.V. Giroscopio Y
Desviación Estándar Allan [°/s]
Tau [s] (Duración Cluster)


VARIANZA ALLAN
A.V. Giroscopio Z
Desviación Estándar Allan [°/s]
Tau [s] (Duración Cluster)

- Comparativa PSD de los sensores y PDS del modelo ARW:

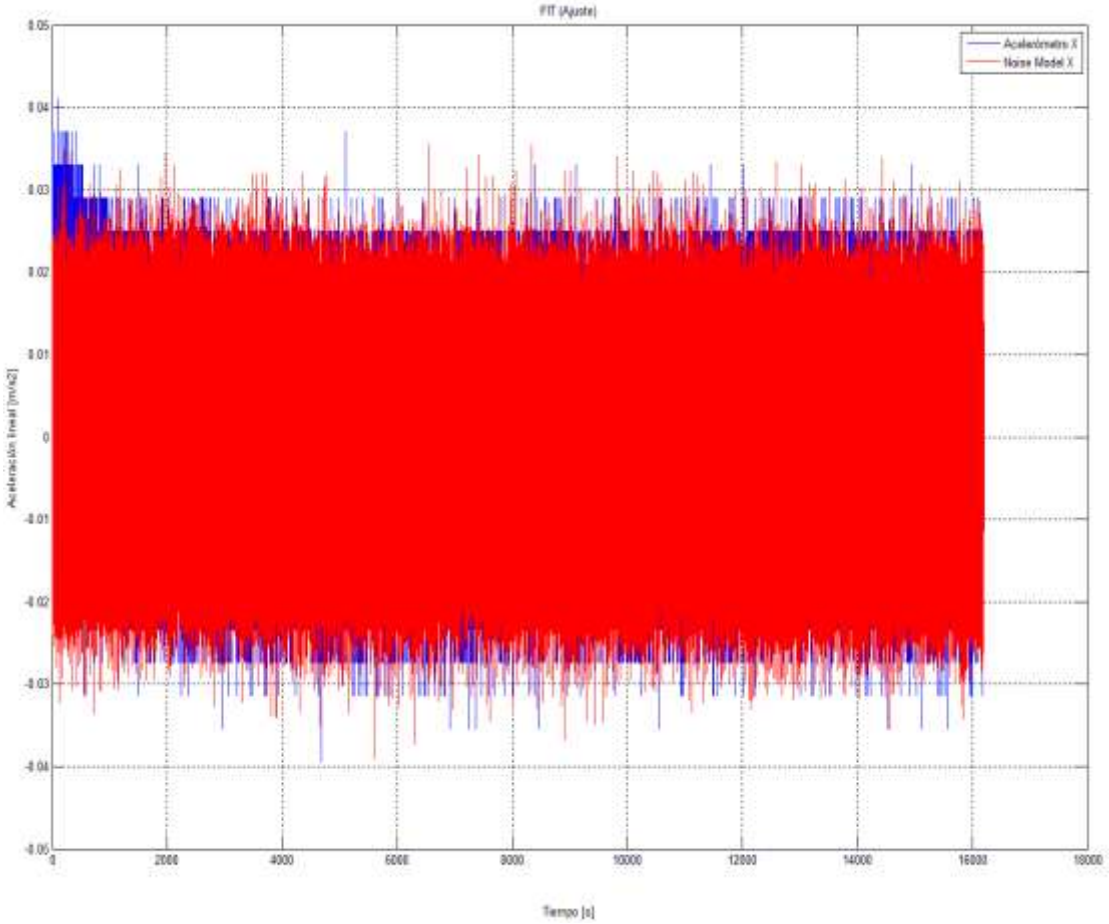


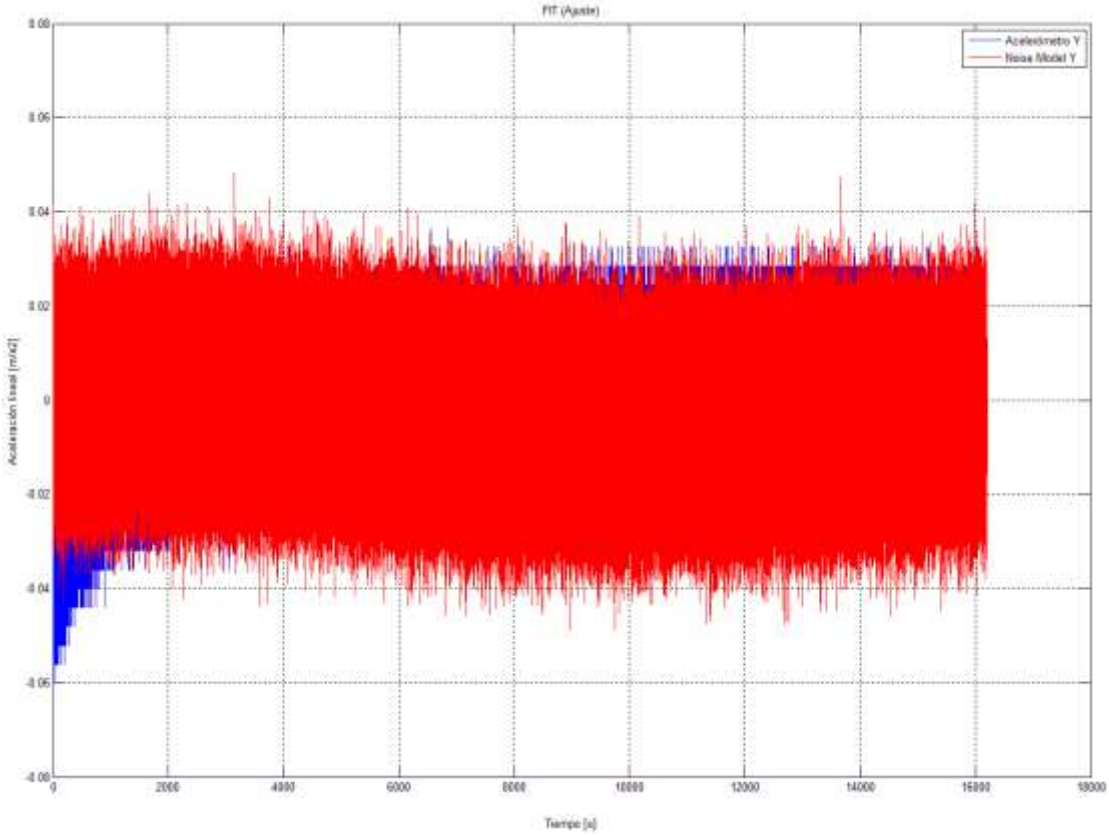


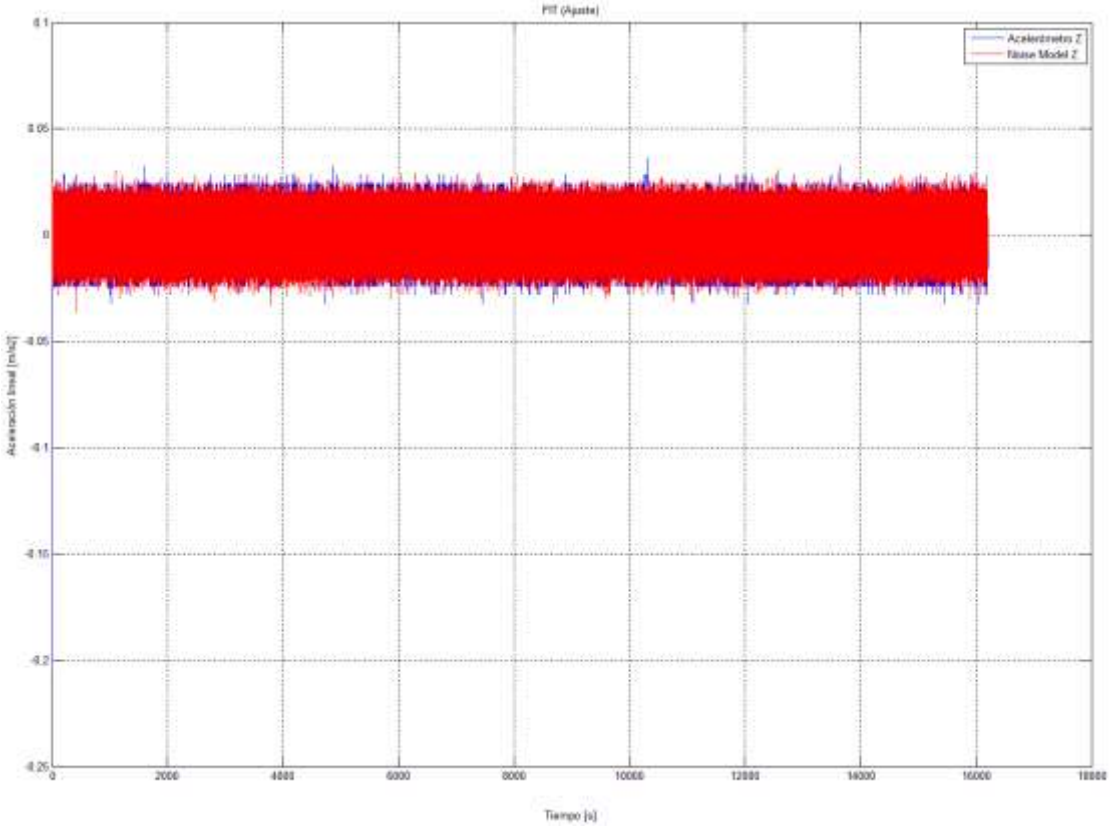


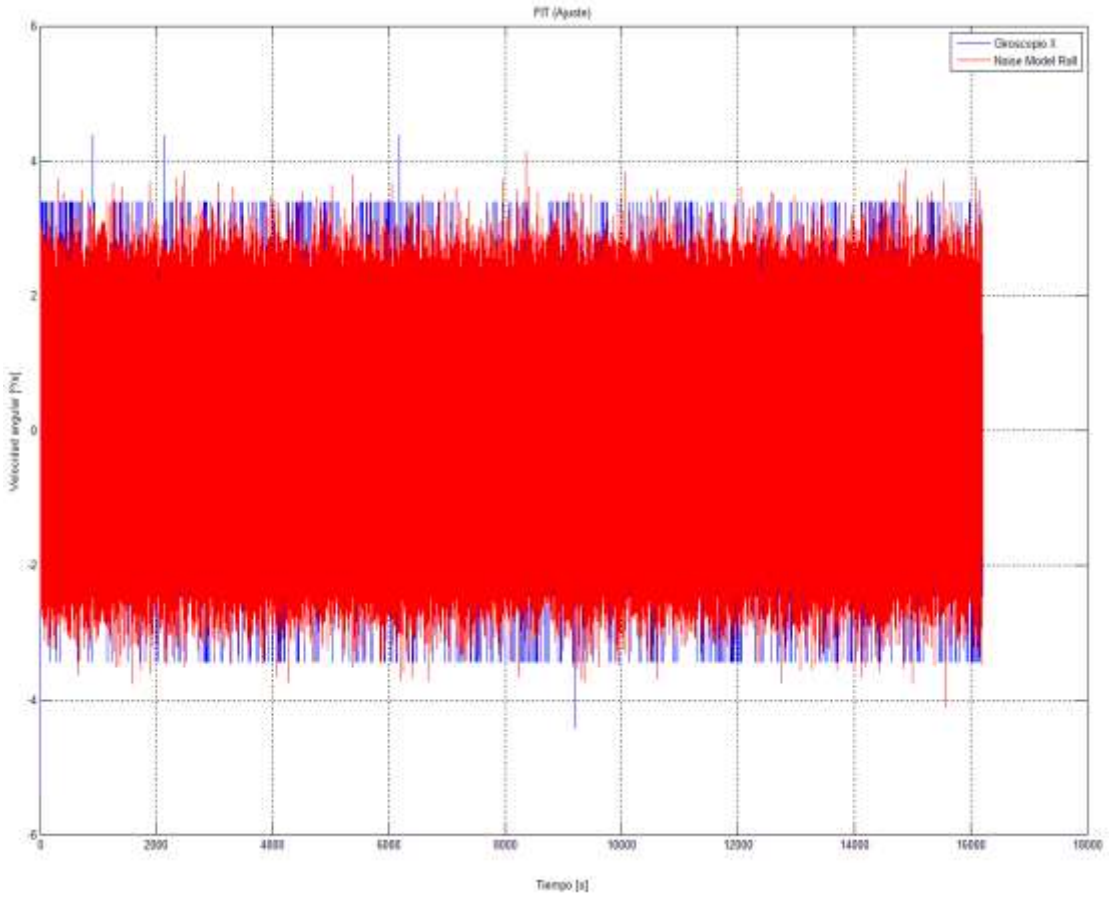
No aparece la contribución de K, Acceleration/Rate Random Walk, para los giroscopios.

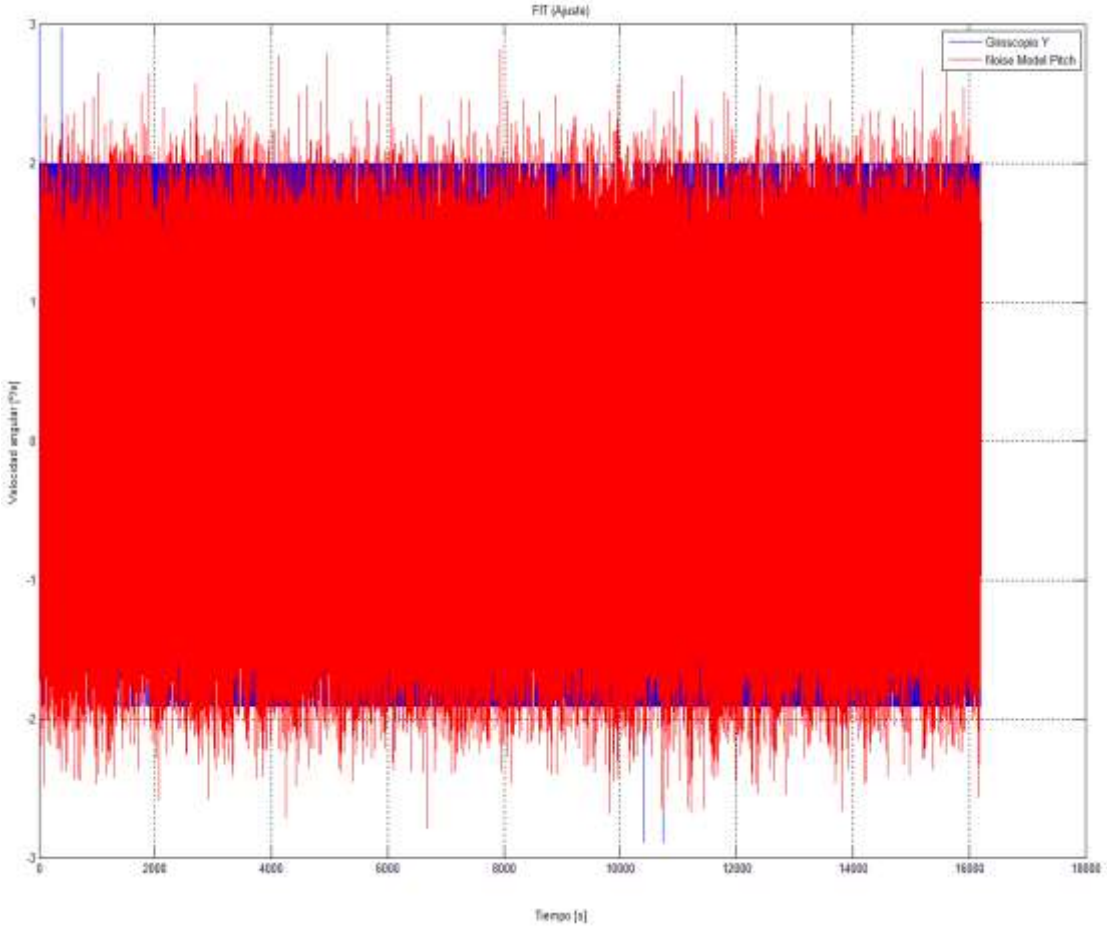
- Modelos FIT obtenidos:

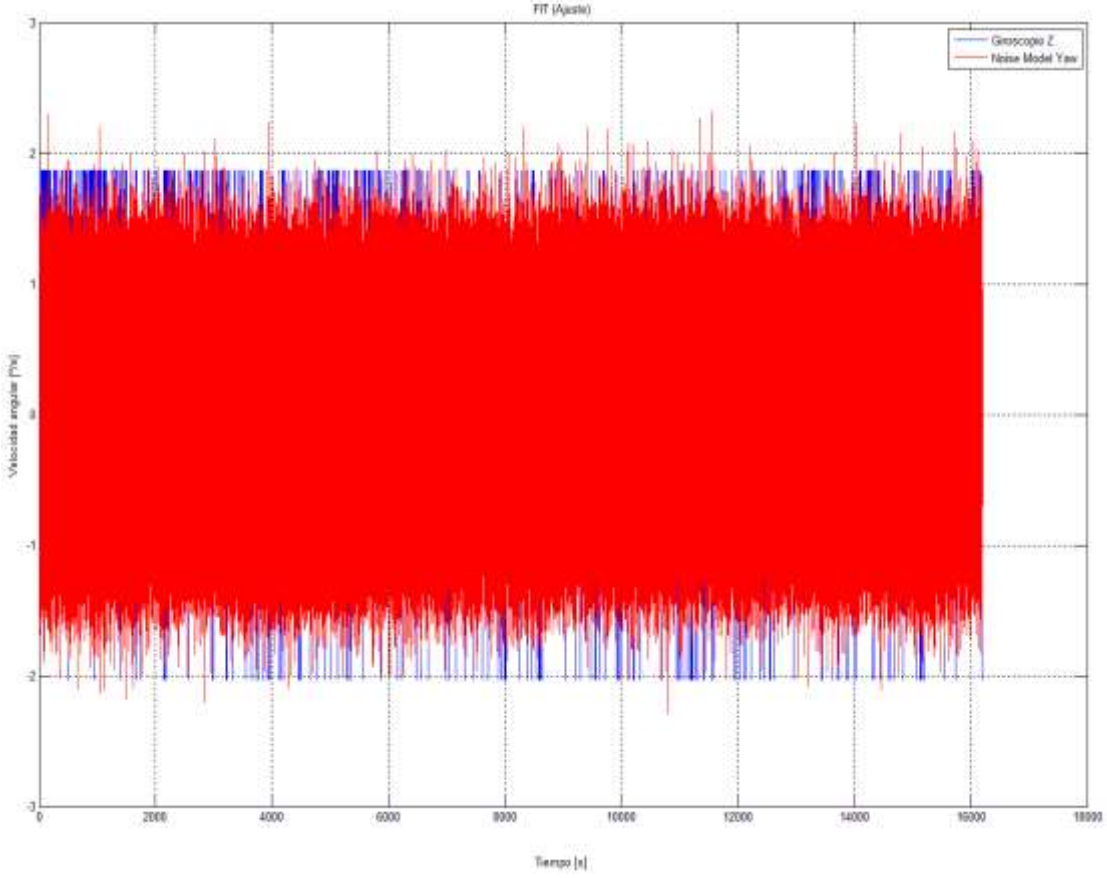





FIT (Ajuste)
Acelerómetro Z
Noise Model Z
Aceleración lineal [m/s2]
Tiempo [s]

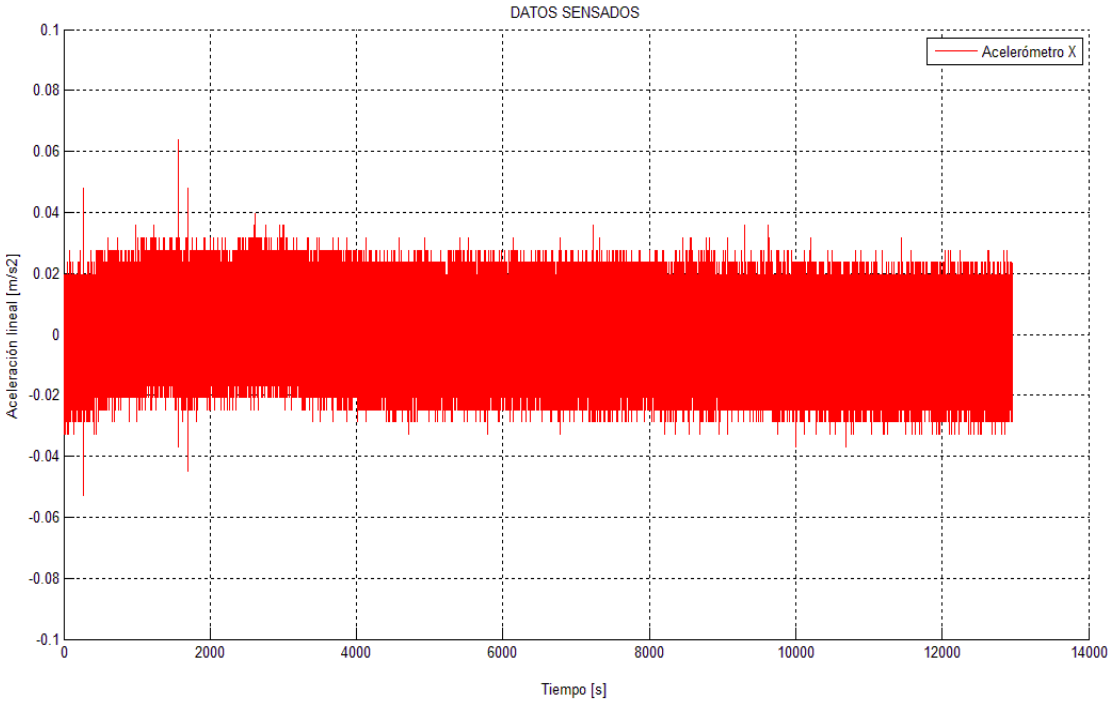

FIT (Ajuste)
Giroscopio X
Noise Model Roll
Velocidad angular [º/s]
Tiempo [s]

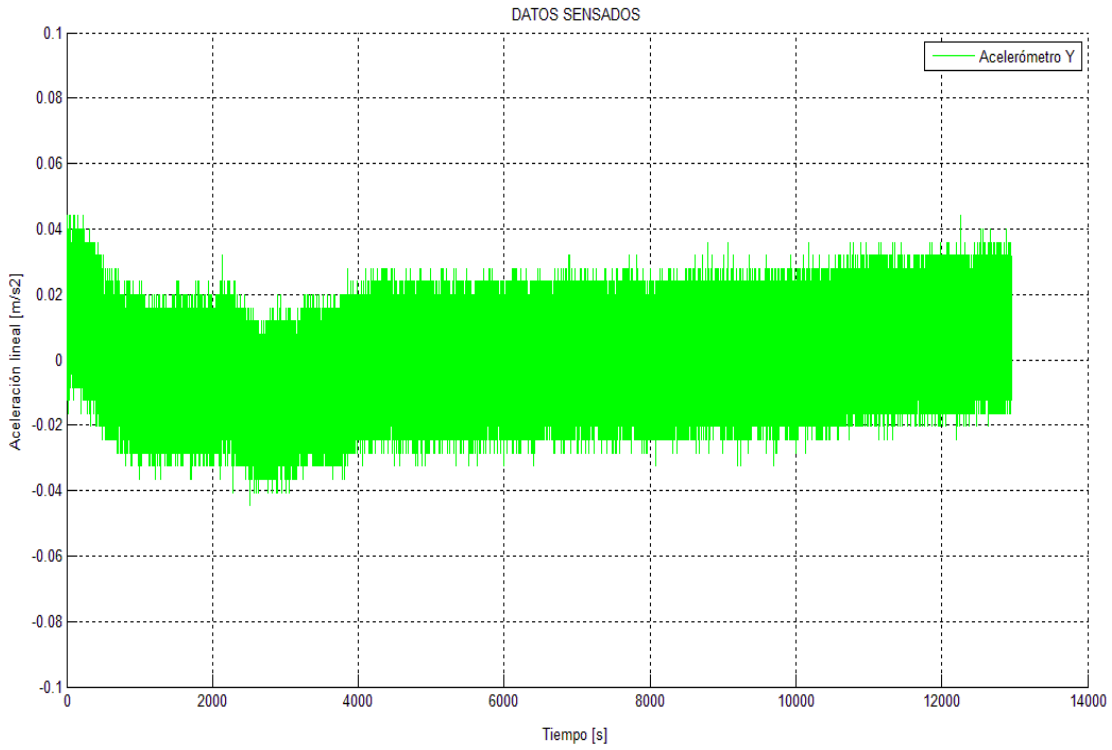

FIT (Ajuste)
Giroscopio Y
Noise Model Pitch
Velocidad angular [°/s]
Tiempo [s]

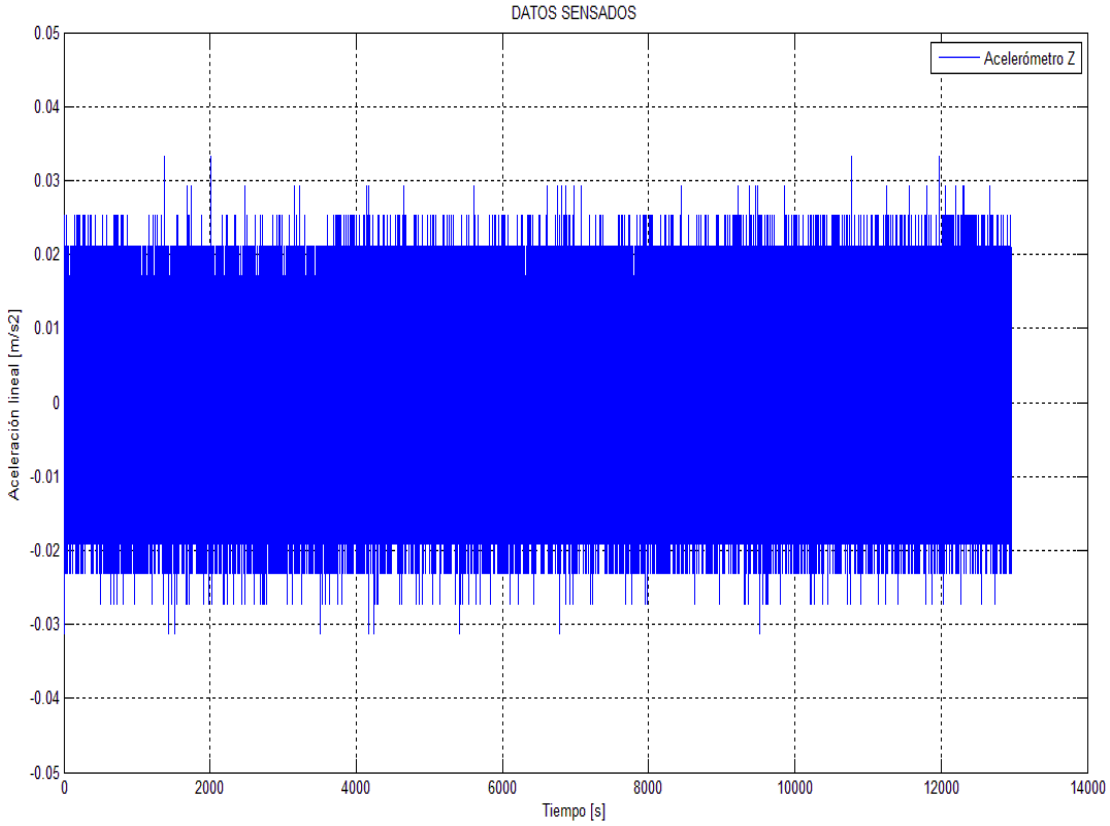

FIT (Ajuste)
Giroscopio Z
Noise Model Yaw
Velocidad angular [°/s]
Tiempo [s]

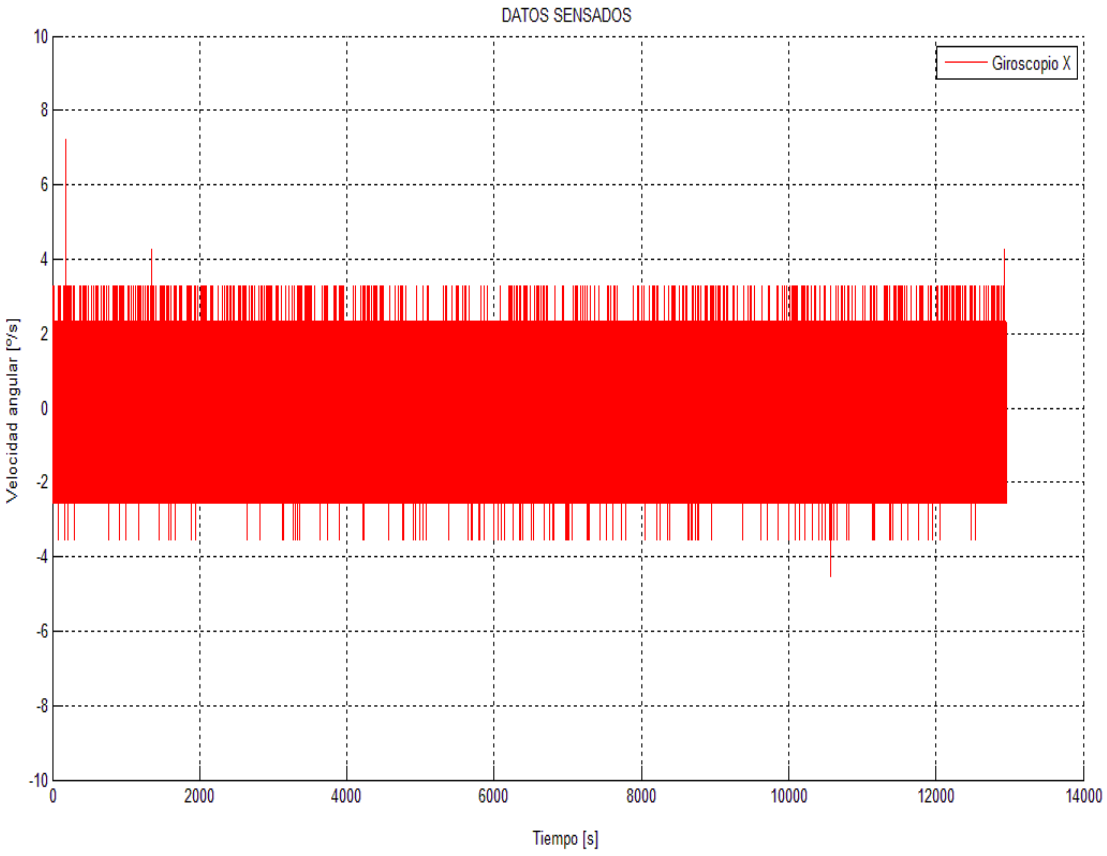
Fs = 250Hz

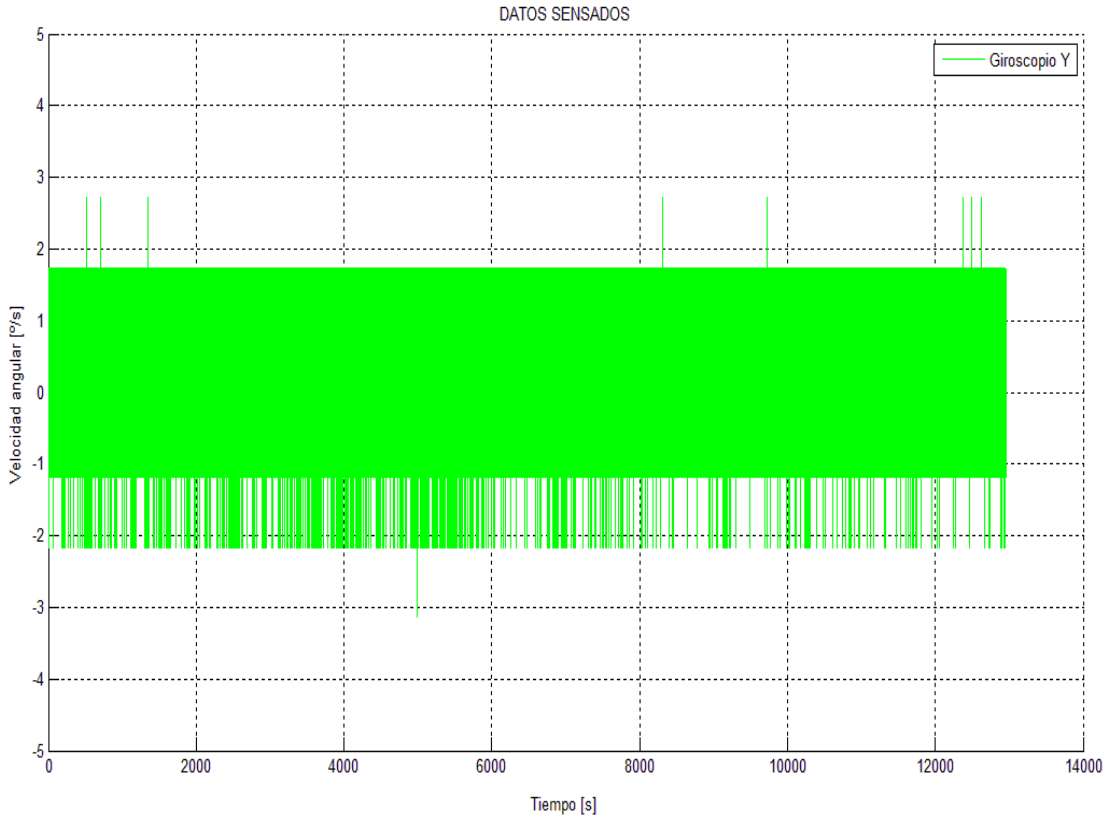
- Lecturas de los sensores:

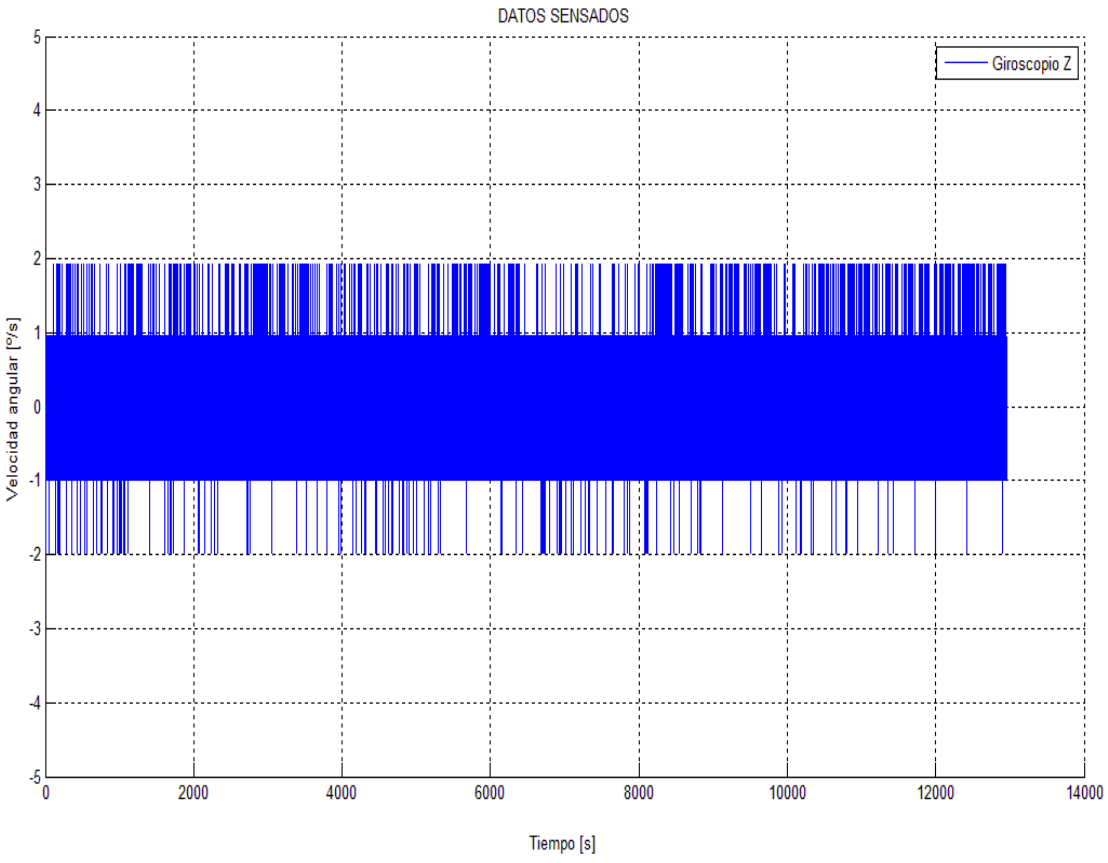




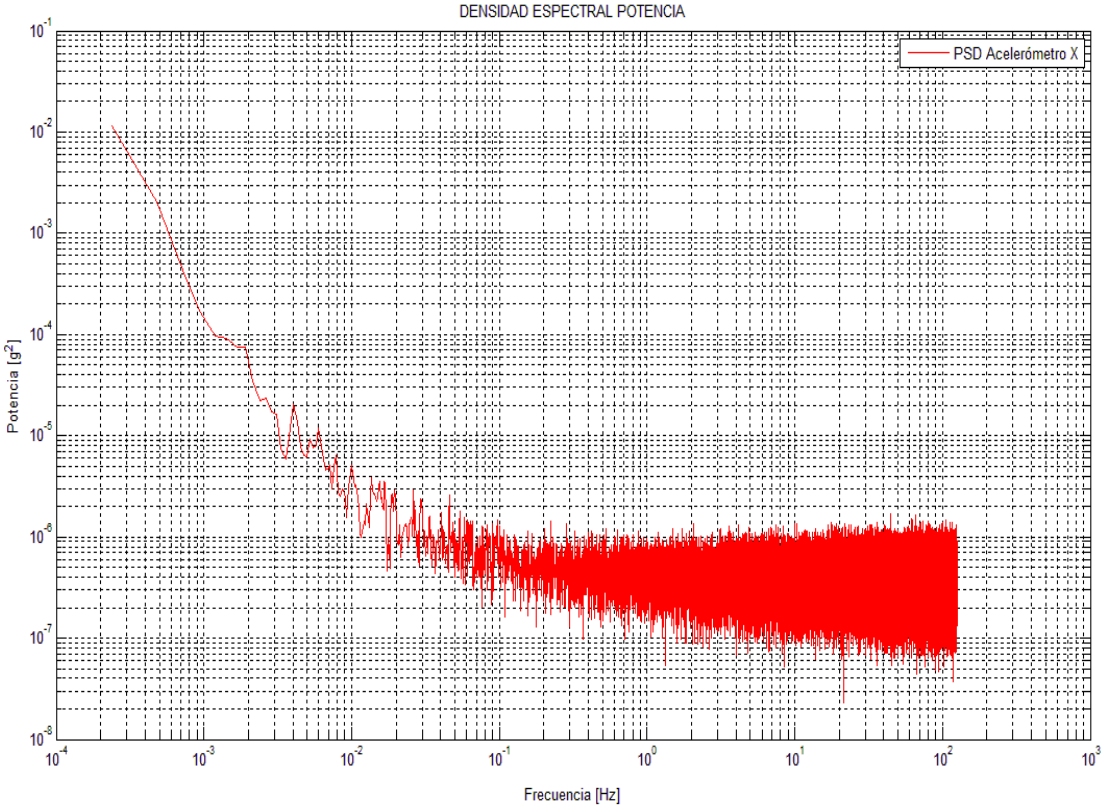

DATOS SENSADOS
Acelerómetro Z
Aceleración lineal [m/s2]
Tiempo [s]
0.05
0.04
0.03
0.02
0.01
0
-0.01
-0.02
-0.03
-0.04
-0.05
0
2000
4000
6000
8000
10000
12000
14000

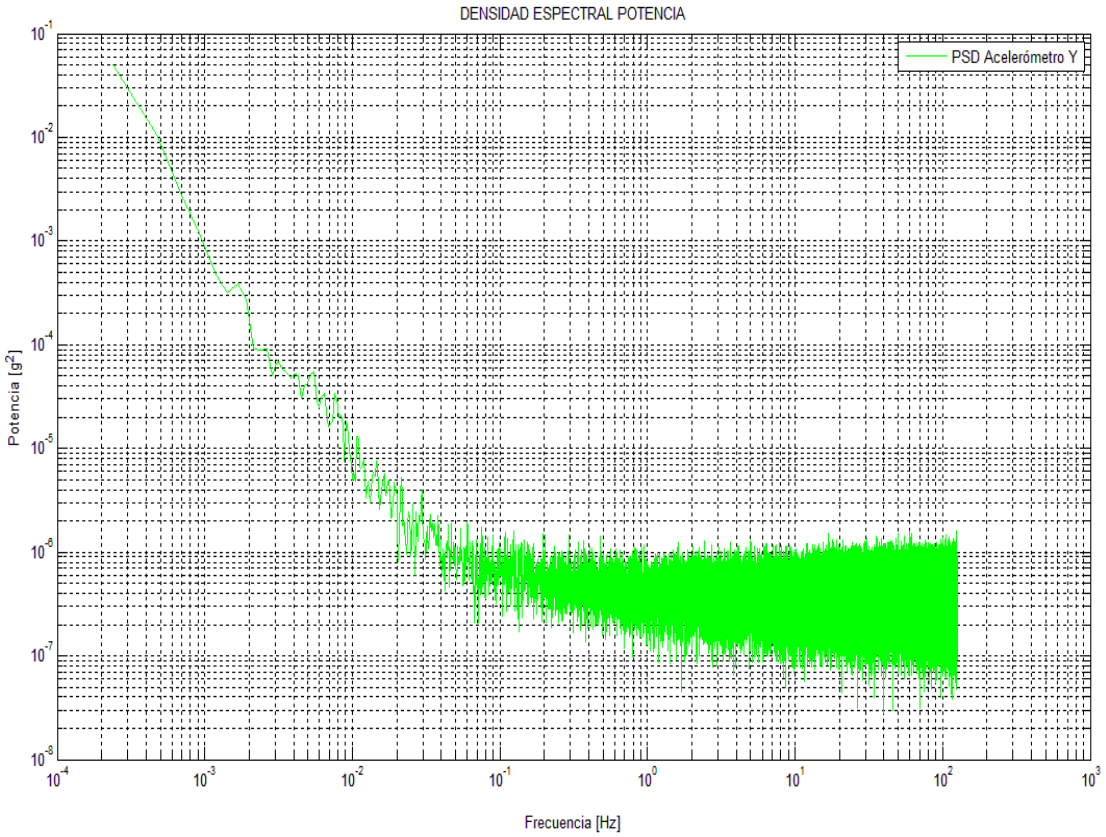

DATOS SENSADOS
Giroscopio X
Velocidad angular [°/s]
Tiempo [s]
10
8
6
4
2
0
-2
-4
-6
-8
-10
0
2000
4000
6000
8000
10000
12000
14000

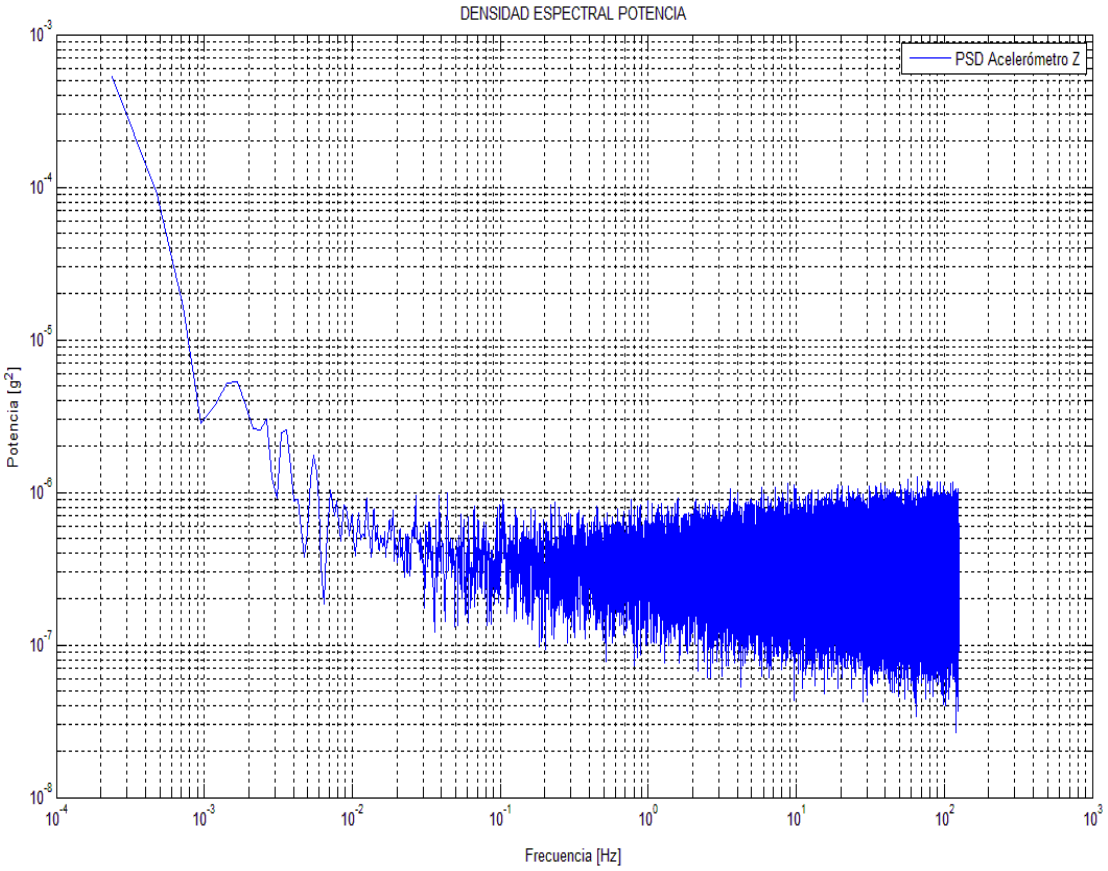

DATOS SENSADOS
Giroscopio Y
Velocidad angular [°/s]
Tiempo [s]
5
4
3
2
1
0
-1
-2
-3
-4
-5
0
2000
4000
6000
8000
10000
12000
14000

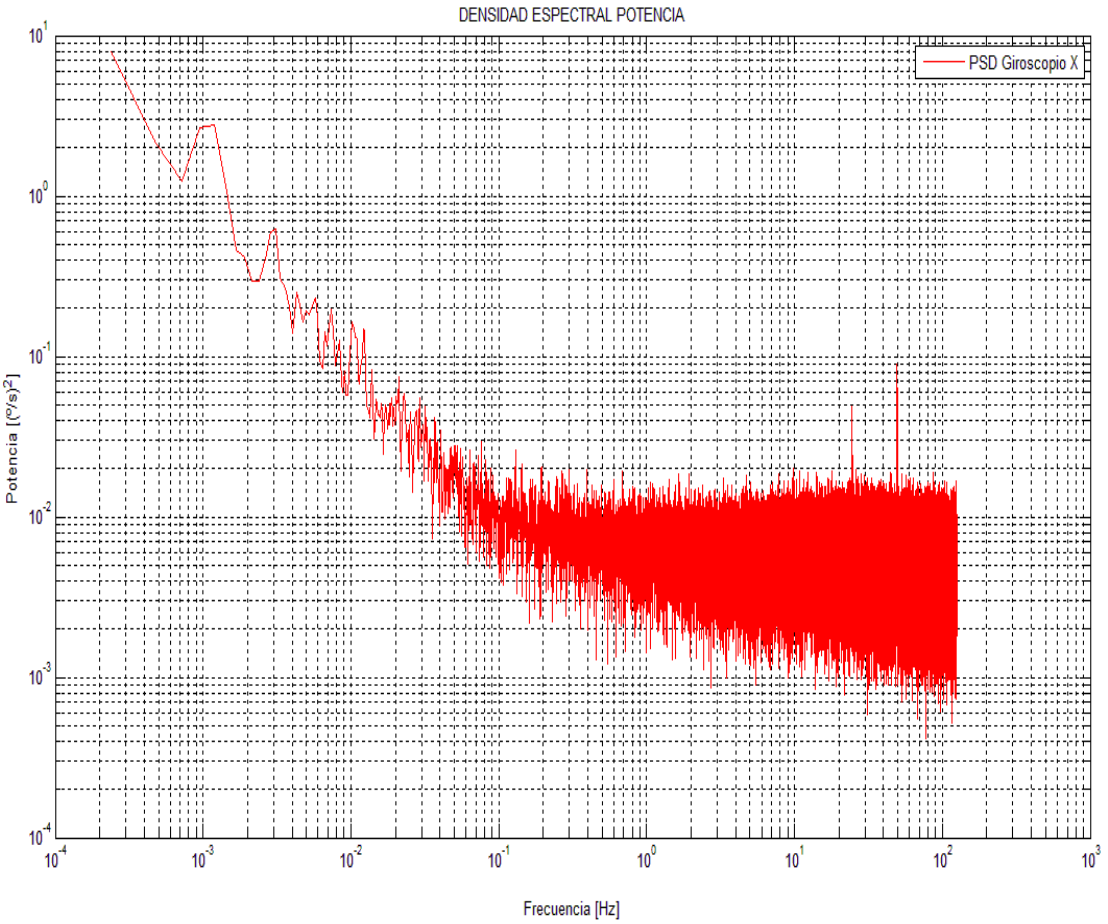

DATOS SENSADOS
Giroscopio Z
Velocidad angular [°/s]
Tiempo [s]
5
4
3
2
1
0
-1
-2
-3
-4
-5
0
2000
4000
6000
8000
10000
12000
14000

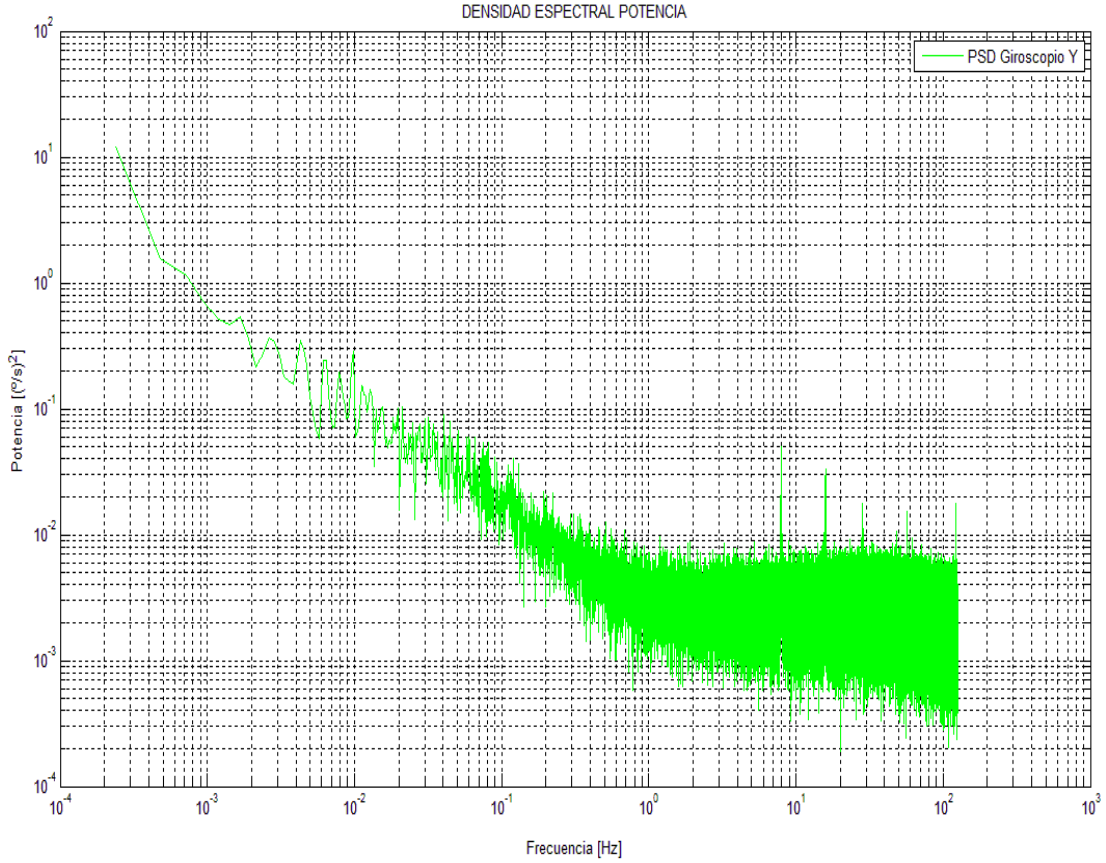
- Densidad espectral de potencia de los sensores:

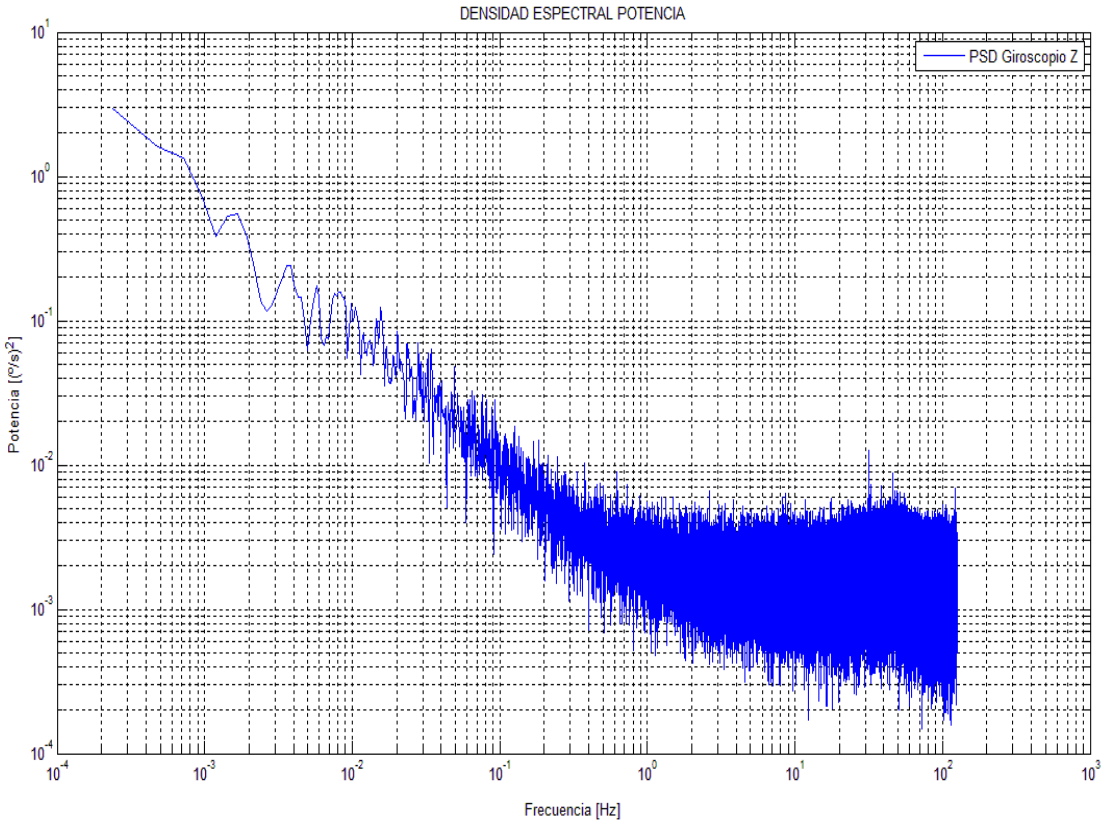




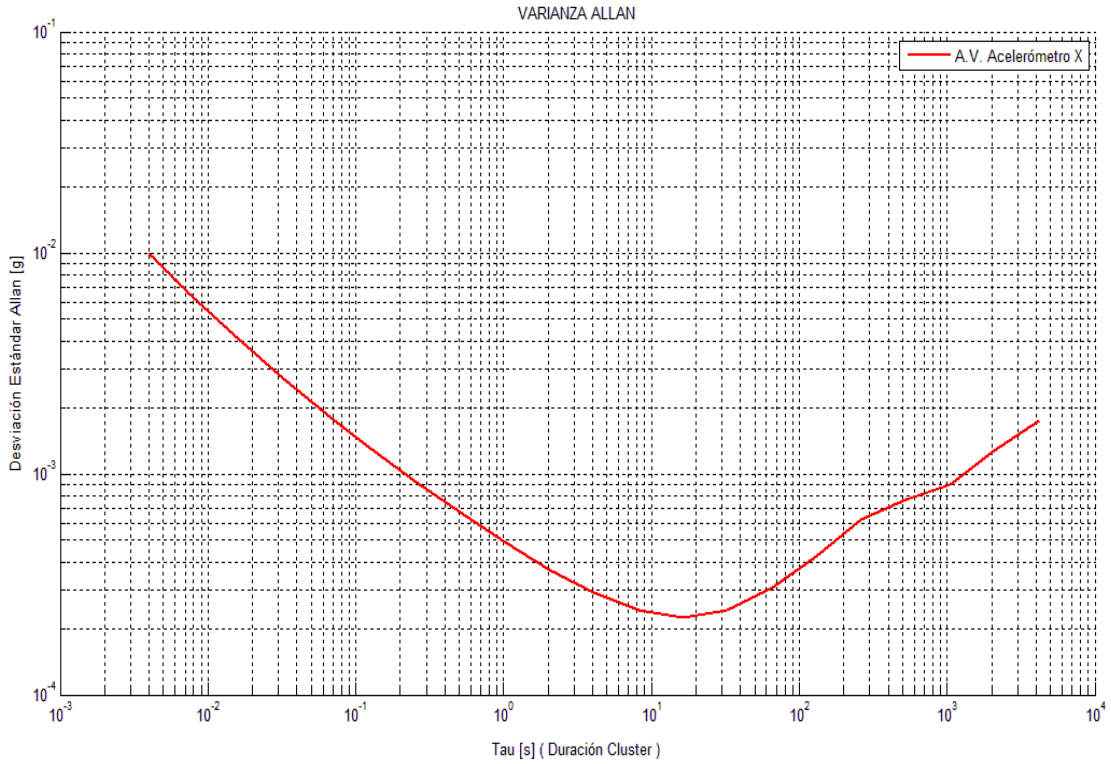

DENSIDAD ESPECTRAL POTENCIA
PSD Acelerómetro Z
Potencia [g²]
Frecuencia [Hz]

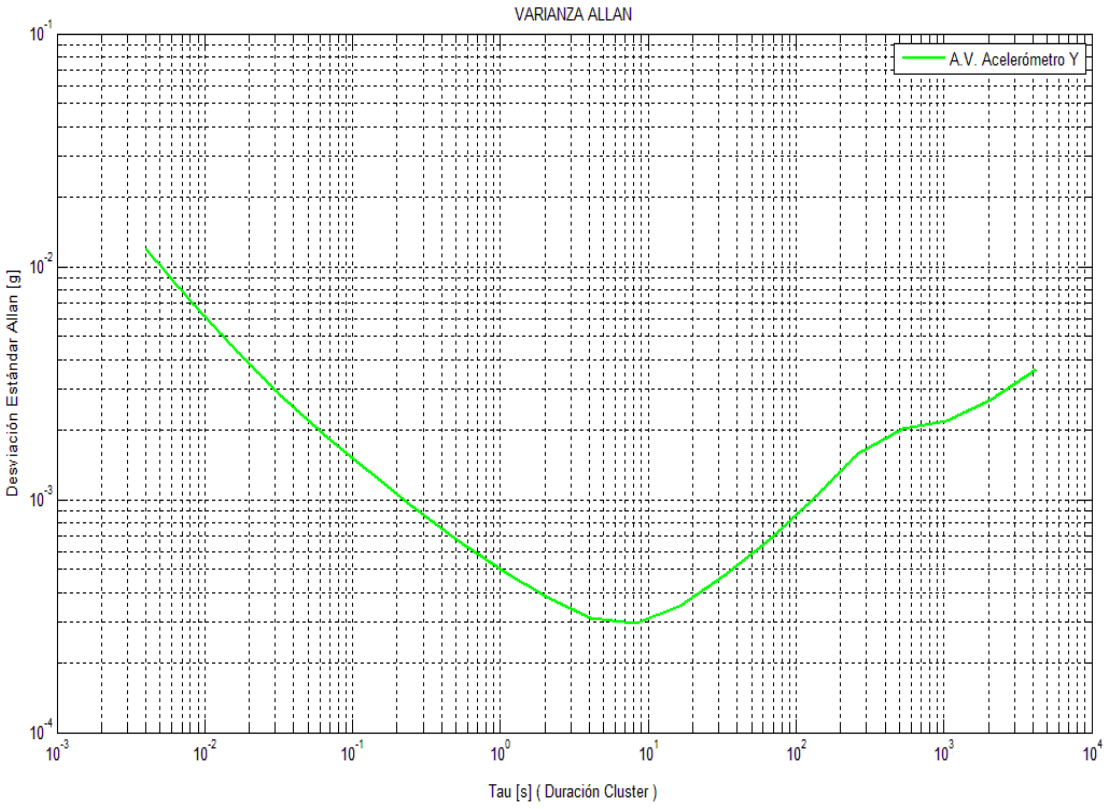

DENSIDAD ESPECTRAL POTENCIA
PSD Giroscopio X
Potencia [(º/s)²]
Frecuencia [Hz]

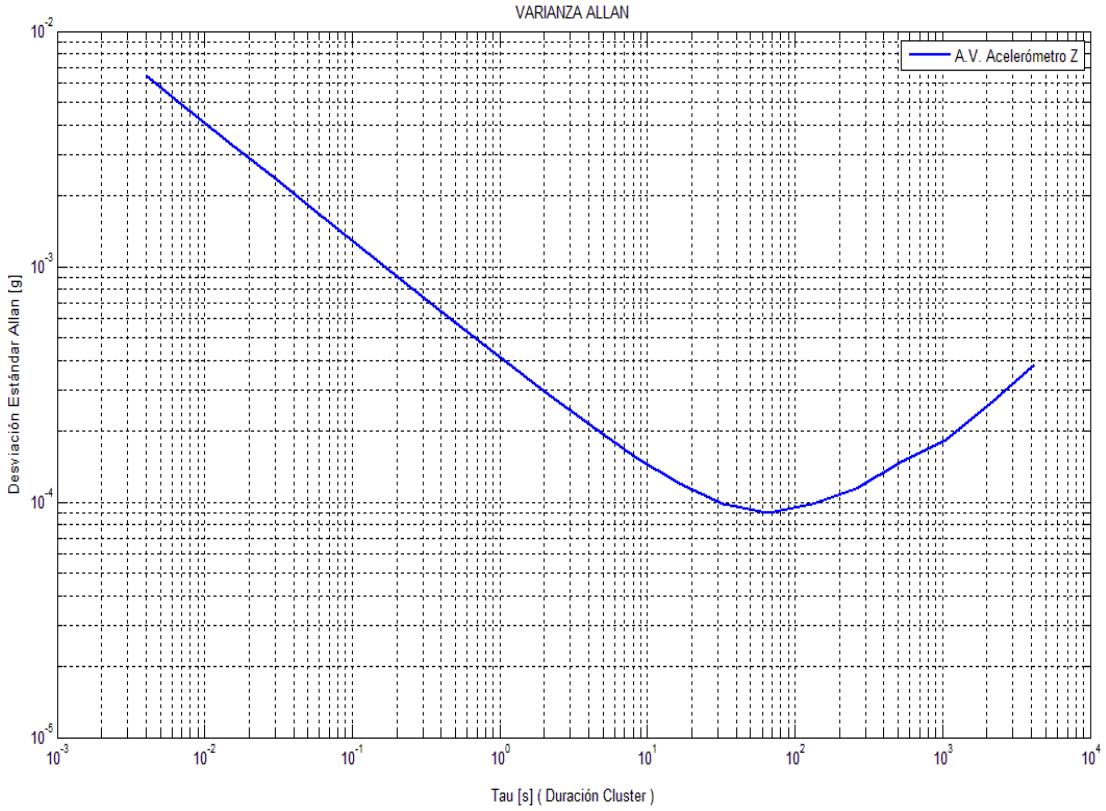

DENSIDAD ESPECTRAL POTENCIA
PSD Giroscopio Y
Potencia [(º/s)²]
Frecuencia [Hz]

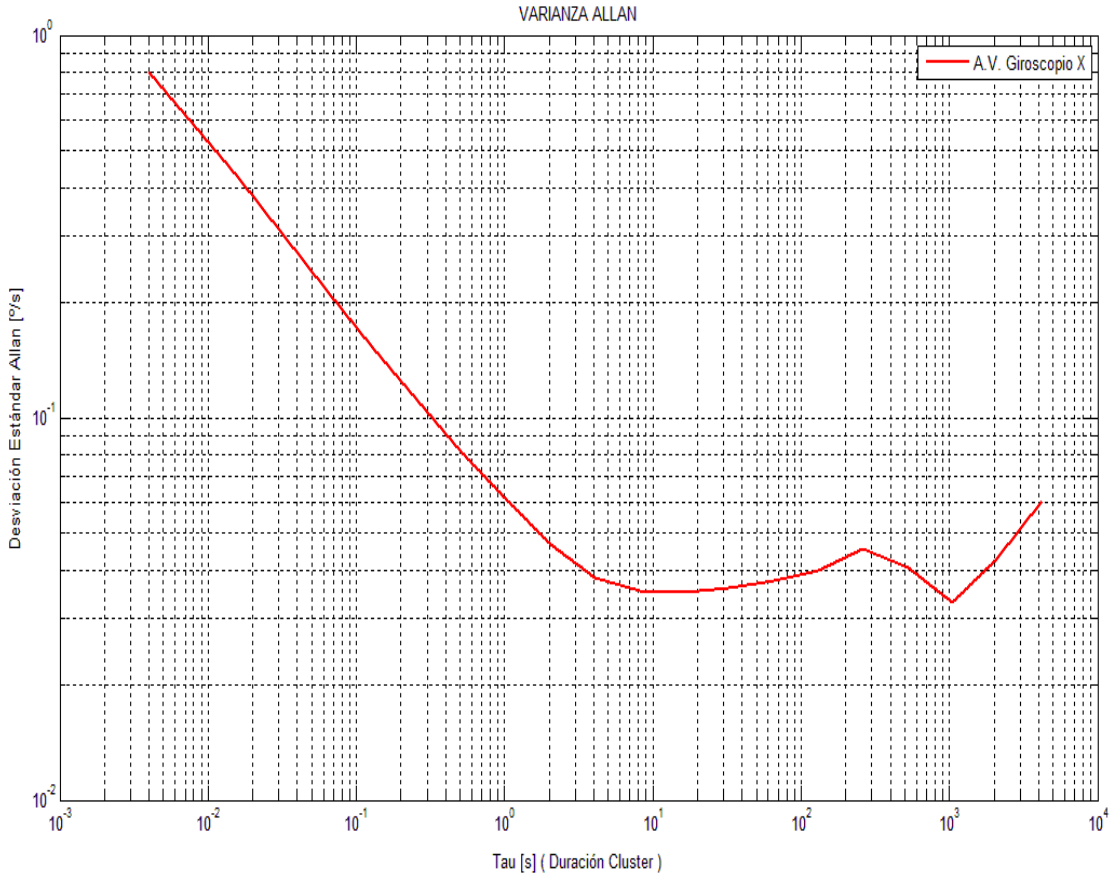

DENSIDAD ESPECTRAL POTENCIA
PSD Giroscopio Z
Potencia [(º/s)²]
Frecuencia [Hz]

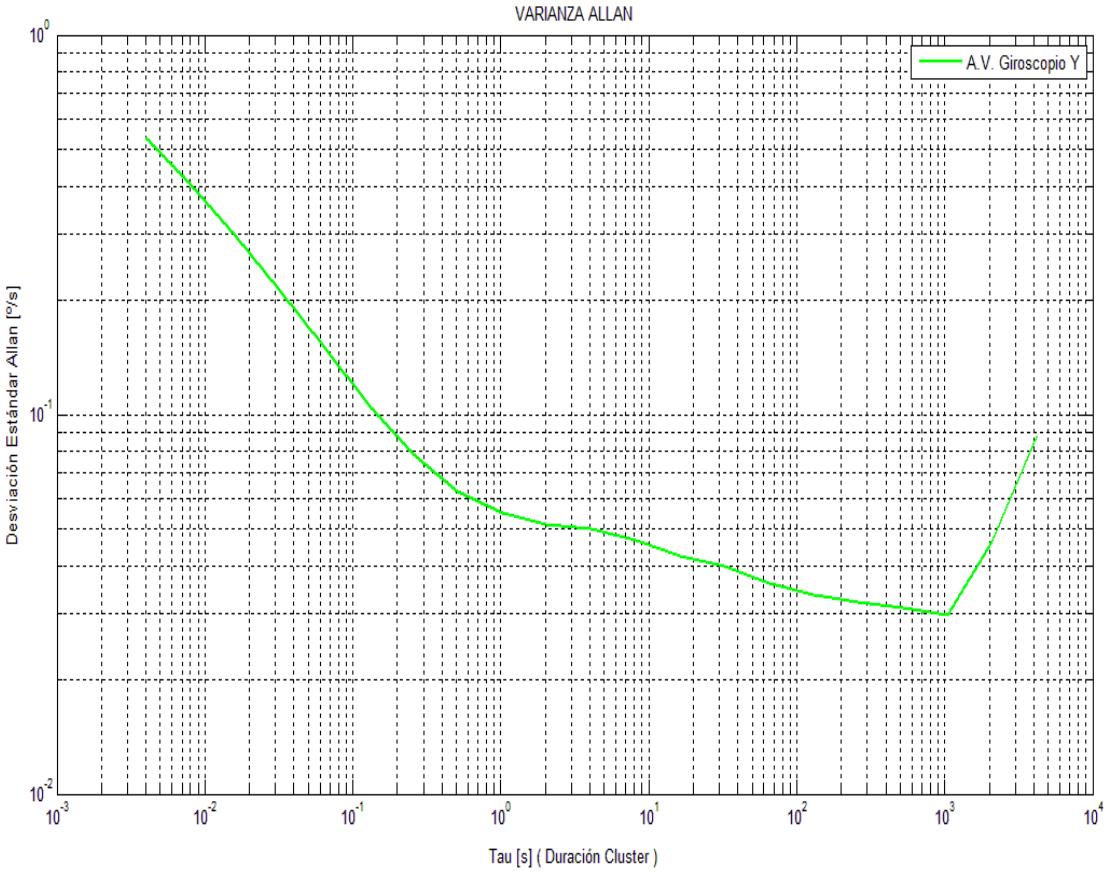
- Resultados del análisis de Allan Variance:

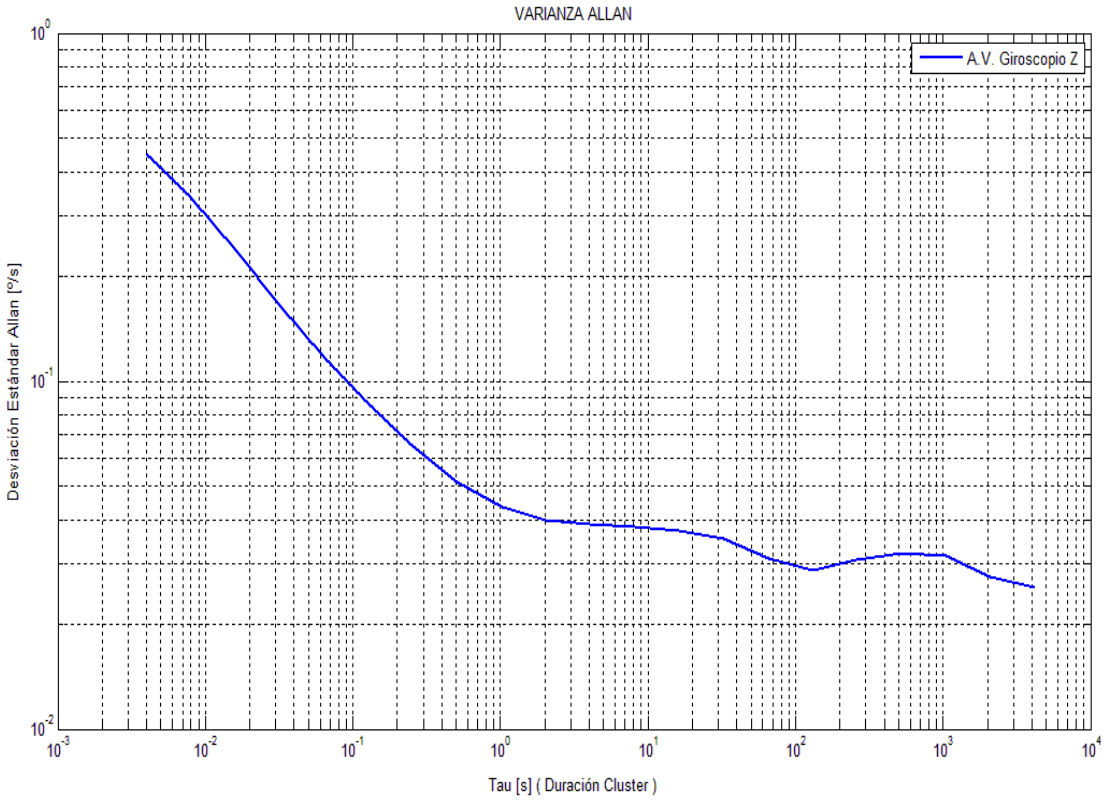




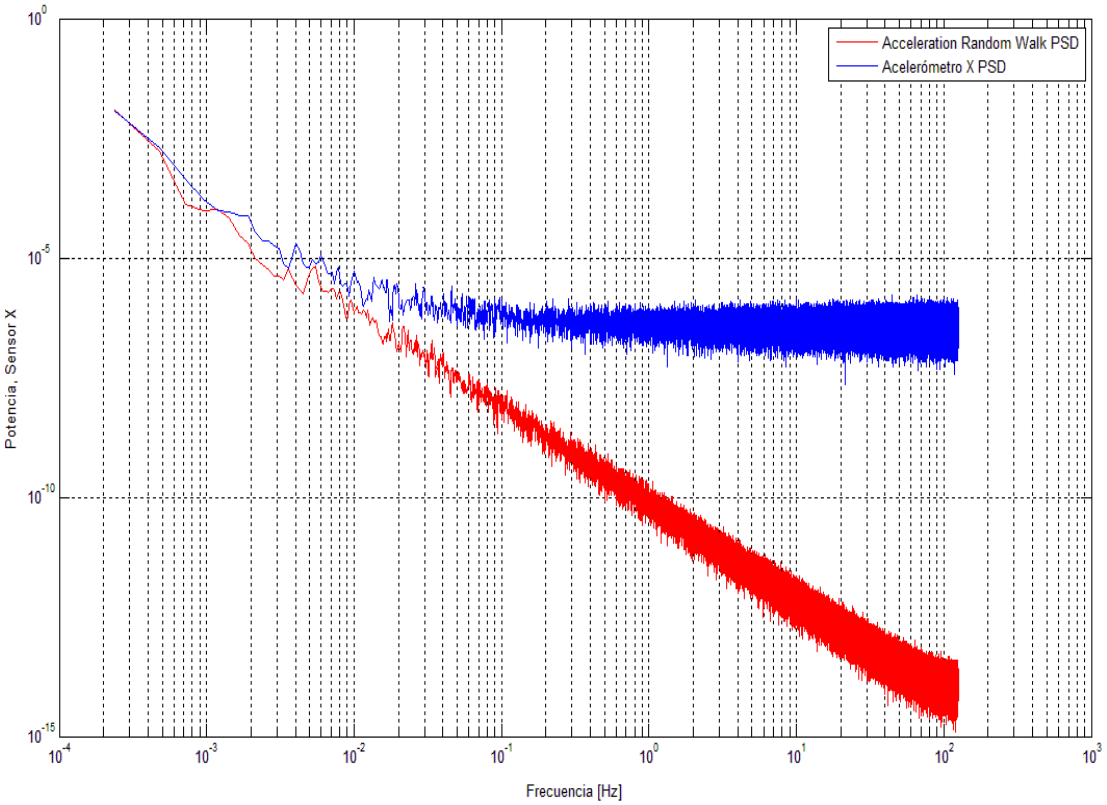

VARIANZA ALLAN
A.V. Acelerómetro Z
Desviación Estándar Allan [g]
Tau [s] (Duración Cluster)

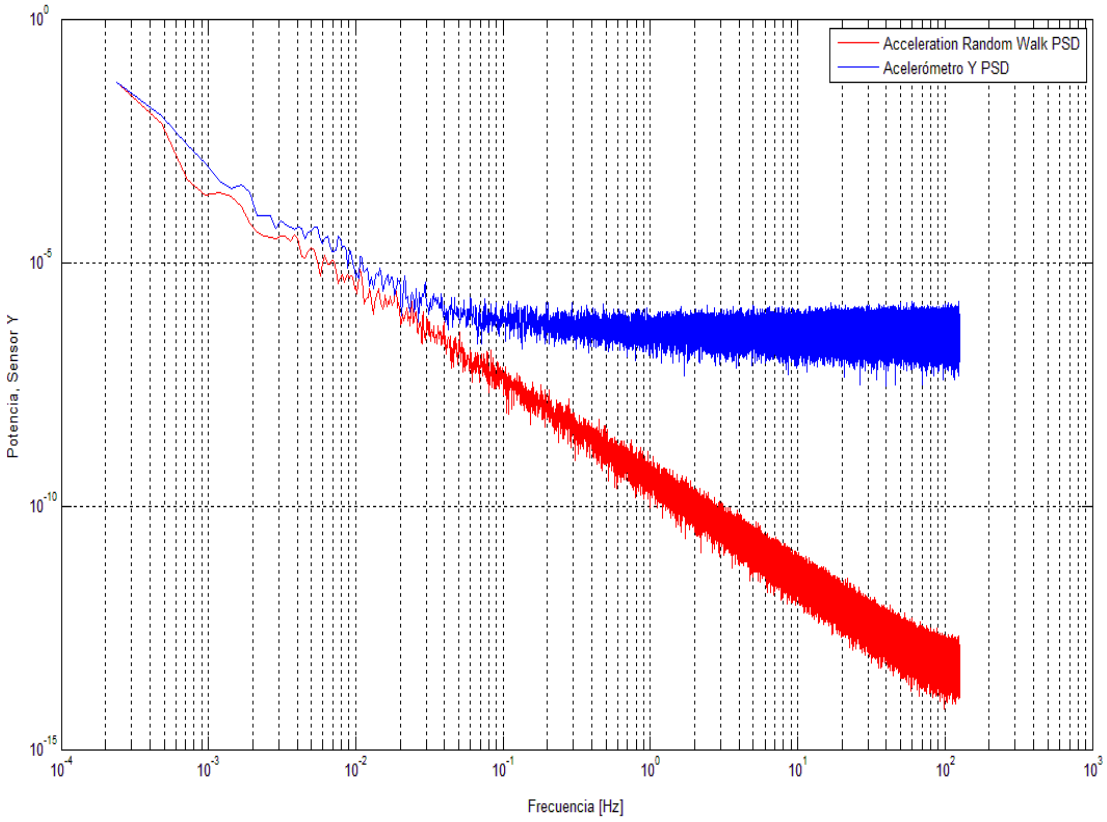

VARIANZA ALLAN
A.V. Giroscopio X
Desviación Estándar Allan [°/s]
Tau [s] (Duración Cluster)

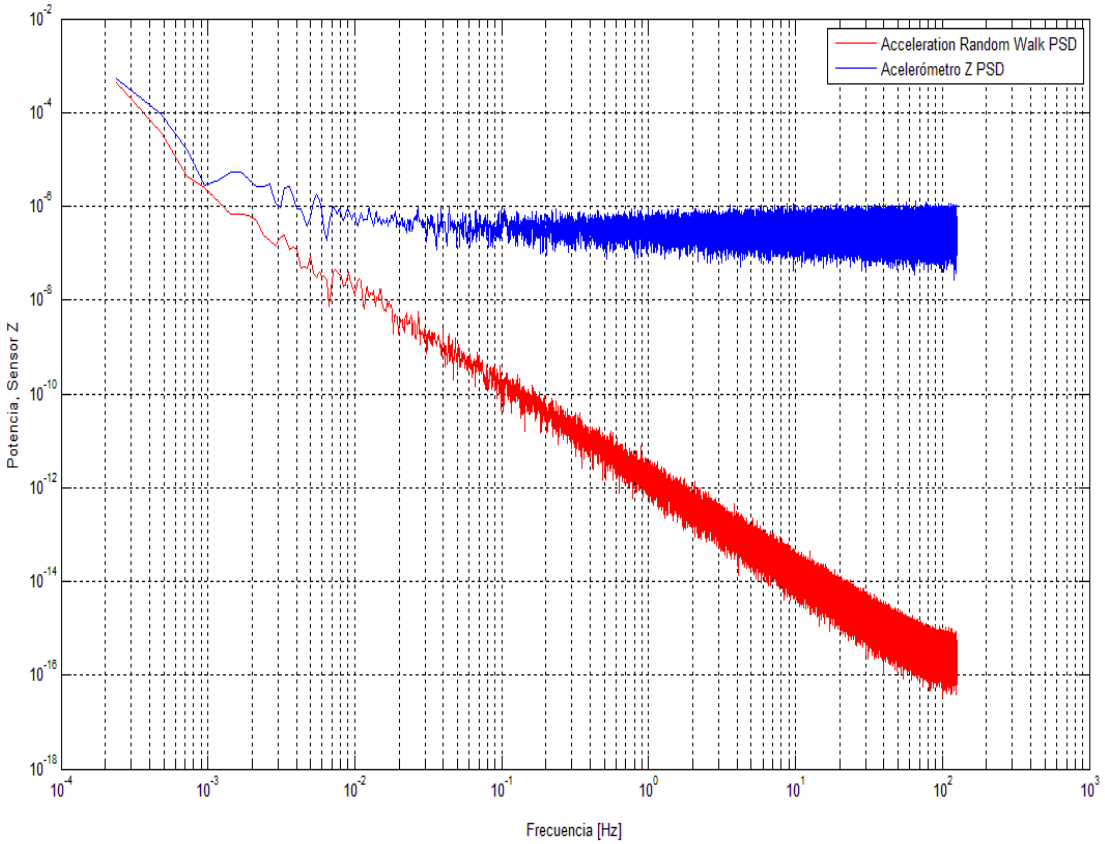

VARIANZA ALLAN
A.V. Giroscopio Y
Desviación Estándar Allan [°/s]
Tau [s] (Duración Cluster)


VARIANZA ALLAN
A.V. Giroscopio Z
Desviación Estándar Allan [°/s]
Tau [s] (Duración Cluster)

- Comparativa PSD de los sensores y PDS del modelo ARW:

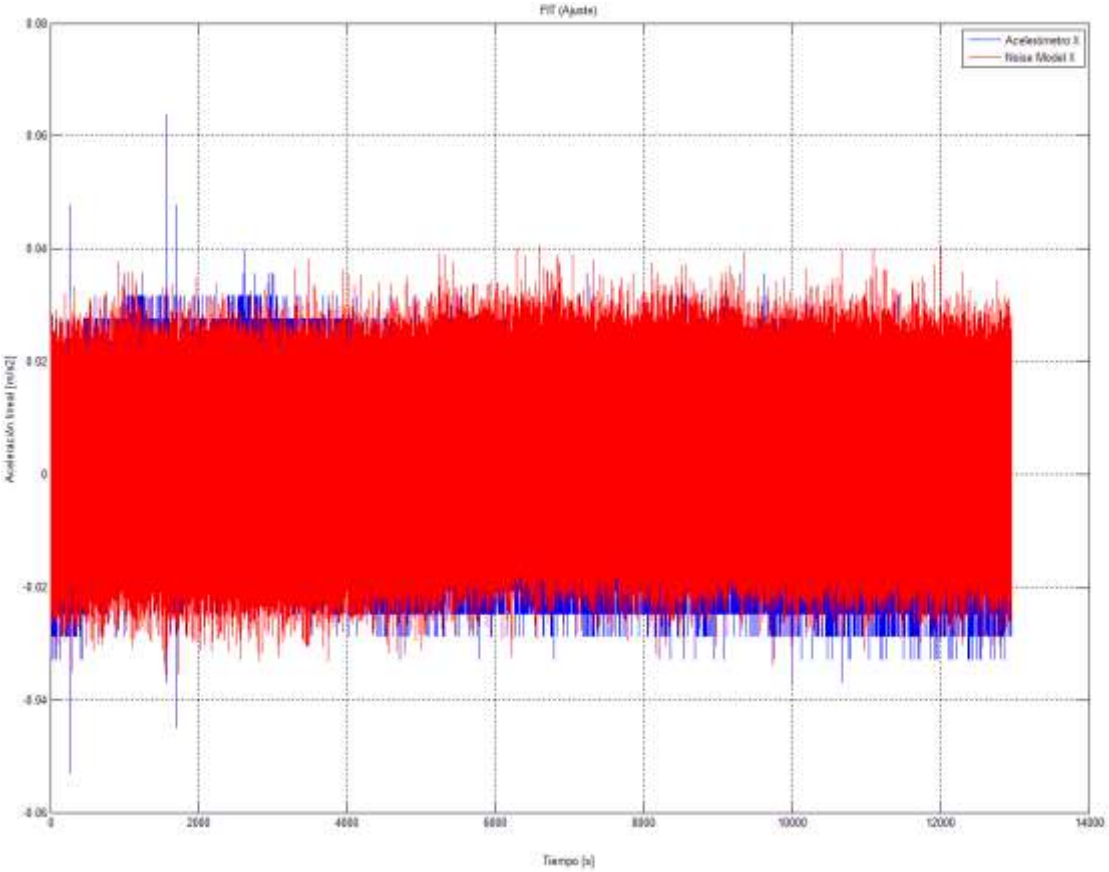


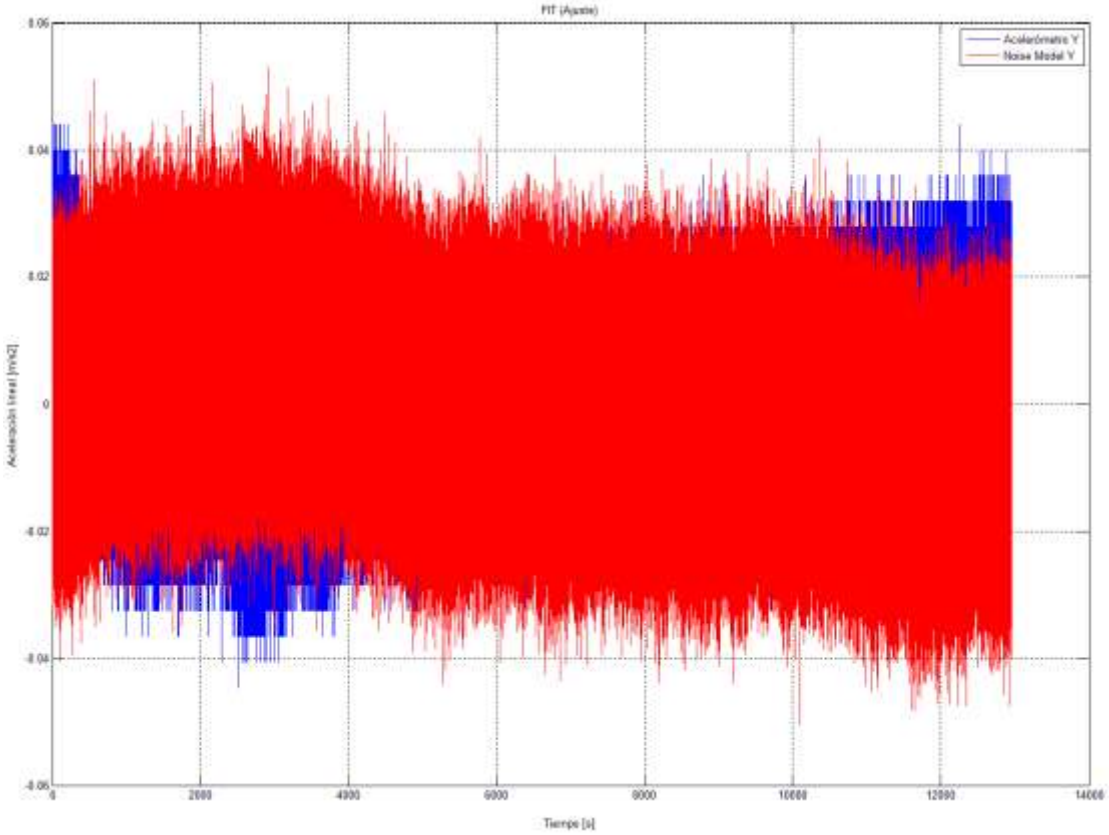


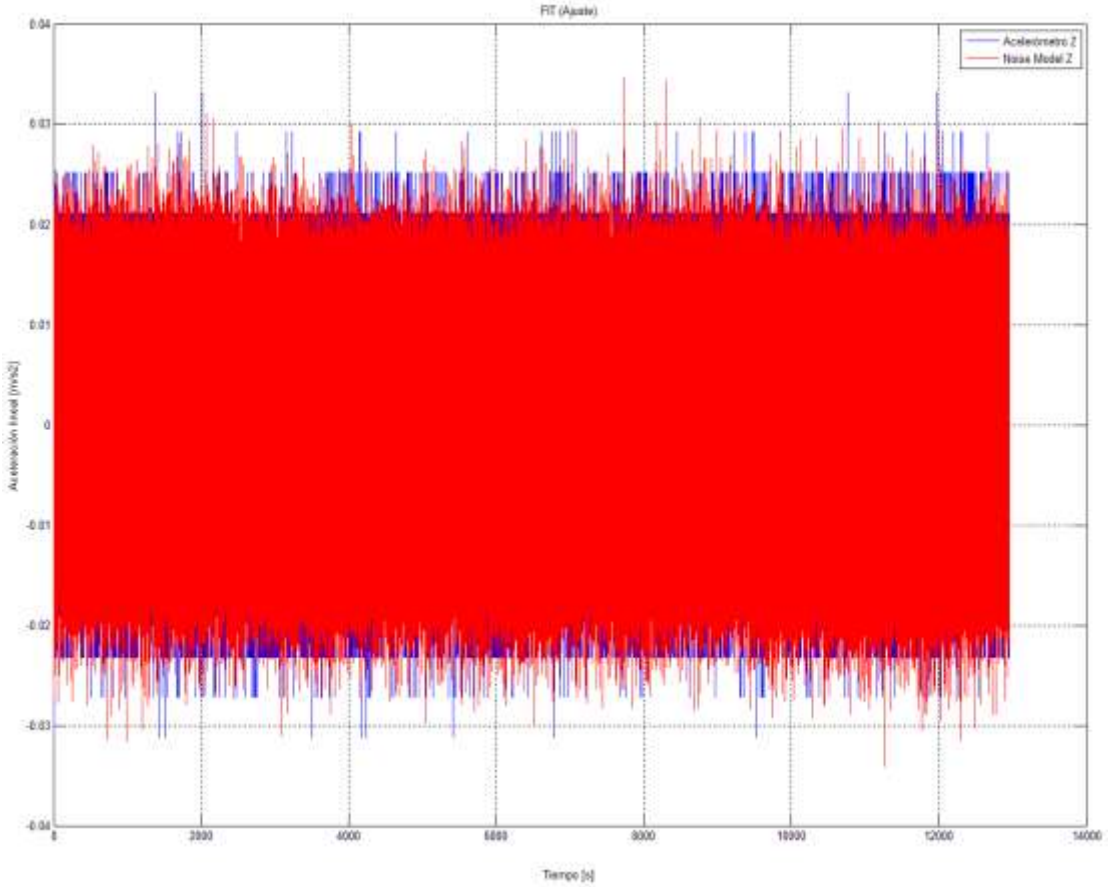


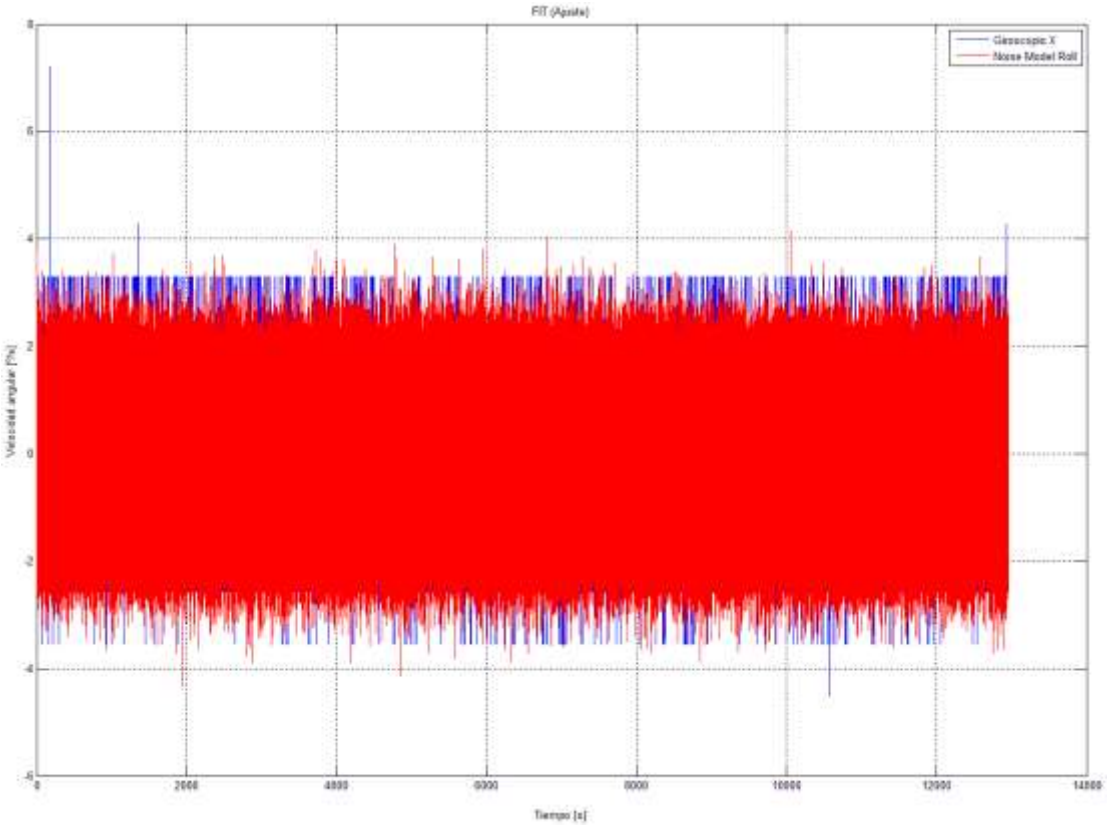
No aparece la contribución de K, Acceleration/Rate Random Walk, para los giroscopios.

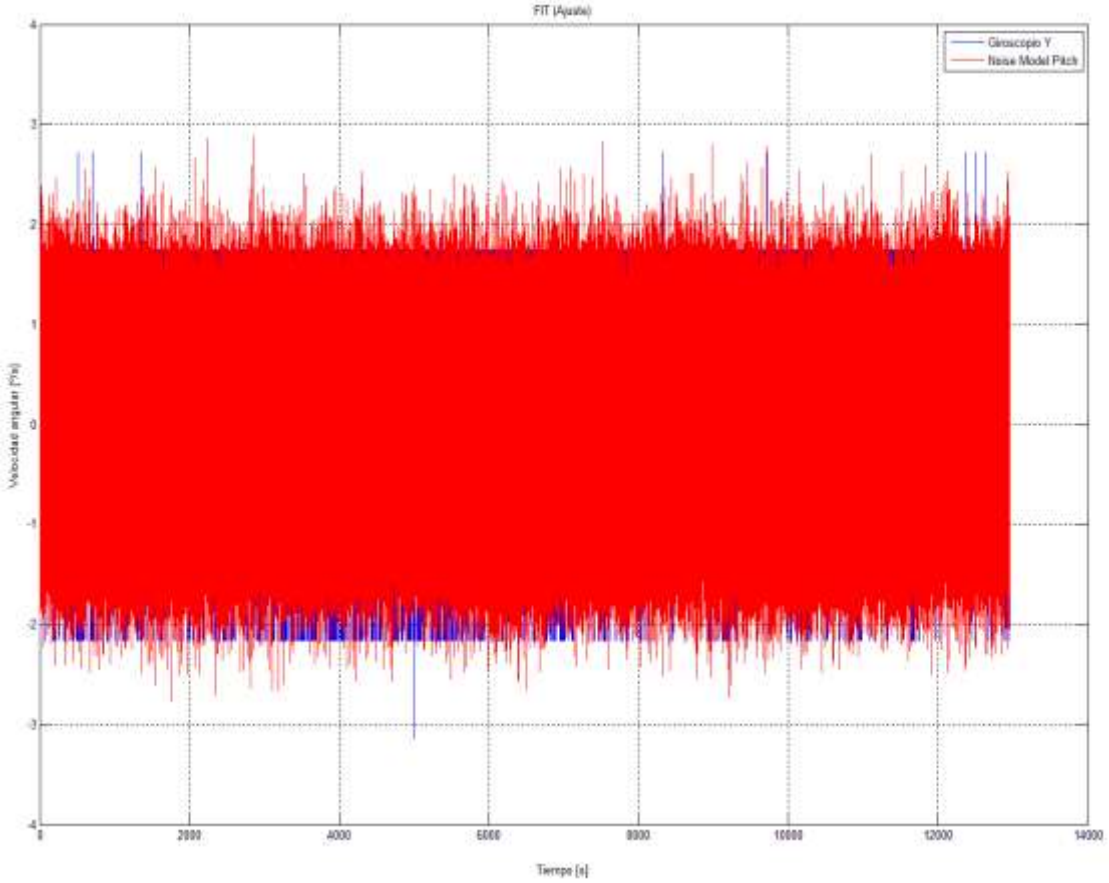
- Modelos FIT obtenidos:

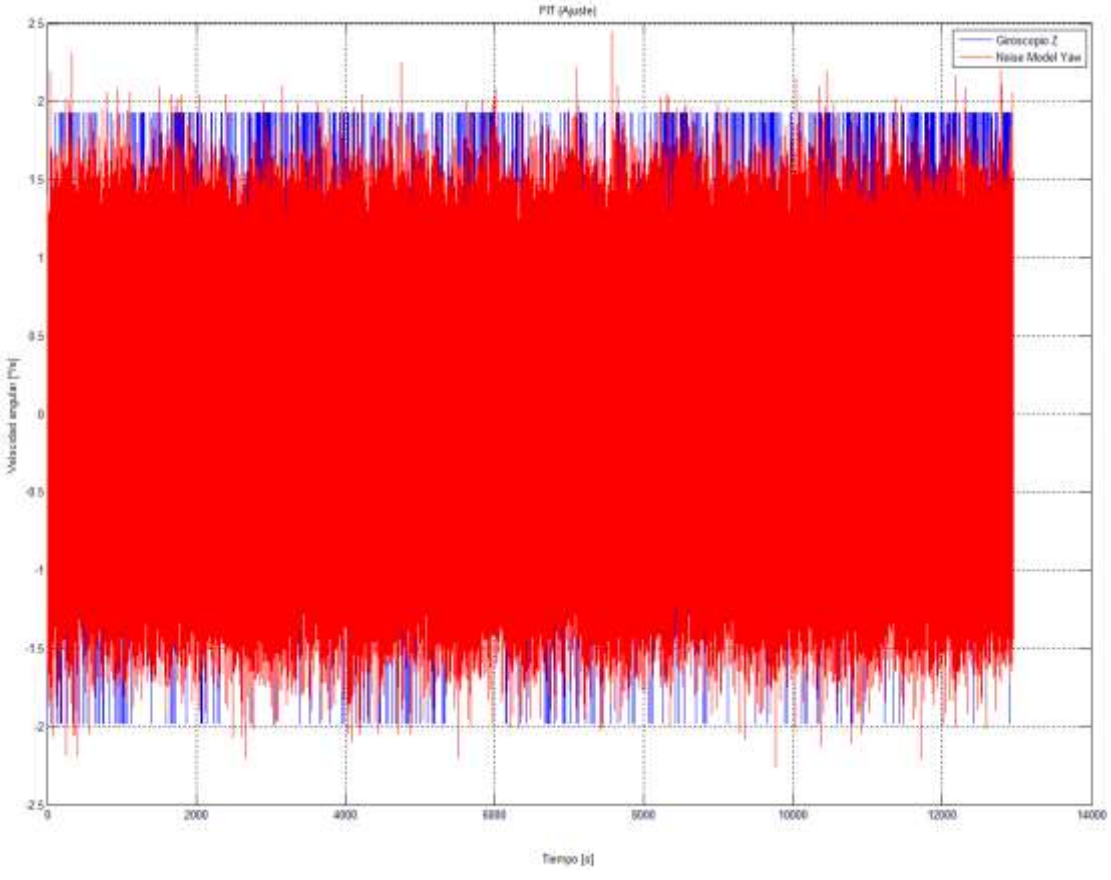





FIT (Ajuste)
Acelerómetro Z
Noise Model Z
Aceleración lineal [m/s2]
Tiempo [s]


FIT (Ajuste)
Giroscopio X
Noise Model Roll
Velocidad angular [?/s]
Tiempo [s]


FIT (Ajuste)
Giroscopio Y
Noise Model Pitch
Velocidad angular [º/s]
Tiempo [s]


FIT (Ajuste)
Giroscopio Z
Noise Model Yaw
Velocidad angular [º/s]
Tiempo [s]

Resum:

Actualment l'ús de sistemes d'ajut a la navegació està molt estès. Aquets sistemes permeten fer un seguiment acurat dels moviments del vehicle on estan integrats i inclús permeten fer-los servir com a sistemes de guiat en vehicles no tripulats.

Aquests sistemes incorporen sensors GPS, brúixoles i IMU. Amb els sensors GPS obtenim l'origen de la navegació i una trajectòria, mentre les condicions ambientals i geogràfiques ho permetin. Amb la brúixola obtenim vectors de referència magnètica. Amb la IMU podem extrapolar vectors de desplaçament, velocitat i acceleracions; sense estar condicionats per els factors ambientals i geogràfics.

No obstant les unitats de mesura inercial son afectades per errors acumulatius en les mesures; efecte que fa que s'hagin de generar models que compensin aquests errors.

En aquest treball crearem una plataforma hardware per integrar una IMU de molt baix cost amb un mòdul GPS i un sensor tèrmic. Dotarem la plataforma amb connectivitat bluetooth i estudiarem, identificarem i modelarem els errors estocàstics que afecten els sensors de la IMU mitjançant l'algorisme de modelat estocàstic de Variància d'Allan. A la fi extraurem els models d'error de la IMU per poder incrementar-ne el rendiment.

Resumen:

Actualmente el uso de sistemas de ayuda a la navegación está muy extendido. Estos sistemas permiten realizar el seguimiento preciso del vehiculo en el que se integran e incluso se pueden utilizar como sistemas de guiado en vehículos no tripulados.

Estos sistemas incorporan sensores GPS, brújulas e IMU. Con los sensores GPS obtenemos un punto origen en la navegación y una trayectoria, siempre que las condiciones ambientales y geográficas lo permitan. Con la brújula obtenemos vectores de referencia magnética. Con la IMU podemos extrapolar vectores de desplazamiento, velocidad y aceleración, sin estar condicionados por los factores ambientales y geográficos.

No obstante las unidades de medida inercial se ven afectadas por errores acumulativos en la medidas; efecto que obliga a generar modelos que compensen y corrijan estos errores.

En este trabajo crearemos una plataforma hardware para integrar una IMU de muy bajo coste con un módulo GPS y un sensor térmico. Dotaremos la plataforma de conectividad bluetooth y estudiaremos, identificaremos y modelaremos los errores estocásticos que afectan a los sensores de la IMU mediante el algoritmo de modelado estocástico de Varianza de Allan Finalmente extraeremos los modelos de error de la IMU, pudiendo asi incrementar su rendimiento.

Summary:

Currently, systems that use navigational aid are widespread. These systems allow precise tracking of the vehicle in which they are integrated and can even be used as guidance systems in unmanned vehicles.

These systems incorporate GPS sensors, compass and IMU. With the GPS sensors we can get a start point and a path for navigation, while the environmental and geographical conditions allow it. With the compass we obtain magnetic reference vectors. We can extrapolate from the IMU displacement, velocity and acceleration vectors, without being constrained by environmental and geographical factors.

Notwithstanding the inertial measurement units are affected by cumulative errors in the measurements; effect requiring generate models to compensate and correct these errors.

In this work we will create a hardware platform to integrate a very low cost IMU with GPS module and a thermal sensor. We will endow the platform with bluetooth connectivity. We will study, identify and will model the stochastic errors affecting the IMU sensors using Allan Variance stochastic modeling algorithm. Finally we extract the error models of the IMU and can thus increase his performance.