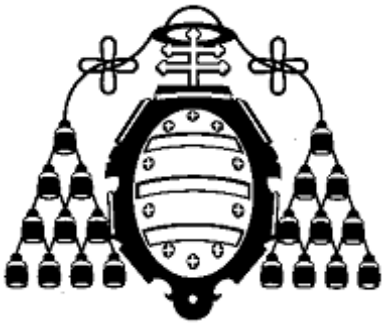


UNIVERSIDAD DE OVIEDO

CENTRO INTERNACIONAL DE POSTGRADO

MASTER EN INGENIERÍA MECATRÓNICA

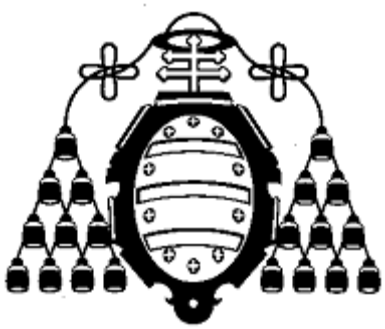
TRABAJO FIN DE MÁSTER

DISEÑO Y VALIDACIÓN DE UN SISTEMA PARA LA ESTIMACIÓN DEL MOVIMIENTO DURANTE EL ENTRENAMIENTO DE MUSCULACIÓN MEDIANTE ACELERÓMETROS

FEBRERO 2016

ALUMNO: ADRIÁN BOURDELANDE GONZÁLEZ

TUTOR: JUAN CARLOS ÁLVAREZ ÁLVAREZ



UNIVERSIDAD DE OVIEDO

CENTRO INTERNACIONAL DE POSTGRADO

MASTER EN INGENIERÍA MECATRÓNICA

TRABAJO FIN DE MÁSTER

DISEÑO Y VALIDACIÓN DE UN SISTEMA PARA LA ESTIMACIÓN DEL MOVIMIENTO DURANTE EL ENTRENAMIENTO DE MUSCULACIÓN MEDIANTE ACELERÓMETROS

FEBRERO 2016

Adrián Bourdelande González **Juan Carlos Álvarez Álvarez**

AGRADECIMIENTOS

Sin duda alguna en primer lugar querría agradecer a mi tutor Juan Carlos Álvarez su ayuda con el desarrollo de este proyecto, por siempre guiarme en la buena dirección pero sin imponerla de forma alguna haciendo que yo recorriese mi propio camino con este proyecto y simplemente siendo un soporte en las cuestiones que fueron necesarias.
En segundo lugar querría agradecer la inestimable ayuda de mi antiguo compañero de máster Alberto Sánchez García que no sólo me ayudó con conocimientos sino que además me prestó hasta algún sensor para la realización de este proyecto.
Una mención muy especial también para mi antiguo compañero de máster Jaime Fernández Alonso sin él tampoco hubiese llegado a poder realizar este trabajo.

Por otro lado agradecer al gimnasio "El Milán" de Oviedo por dejarme hacer las pruebas con el sensor en sus instalaciones durante varios días e incluso permitirme grabar dentro de sus instalaciones.

RESUMEN

Este trabajo final de master consiste principalmente en la captación de datos de algunos movimientos concretos que se realizan en deportes tales como el Powerlifting y su posterior procesado y análisis.
Para la captación del movimiento se usará un sensor del tipo IMU (Unidad de medición inercial), una vez obtenidos los datos se aplicará un algoritmo para poder determinar la variación de la orientación a lo largo del tiempo y con ello se podrá obtener información para la mejora en la preparación física de los atletas.
Obtener cierta información sobre un movimiento determinado en la misma sesión de entrenamiento o en el mismo día puede ayudar en gran medida a personalizar los entrenamientos no sólo para un atleta determinado, sino para cada día de forma individual y única.

PALABRAS CLAVE

IMU, orientación, TRIAD, ARDUINO, musculación, aceleración, campo magnético, press de banca, calibración.

INDICE

1. INTRODUCCIÓN

La estimación de la orientación y posición de un cuerpo es un problema que se lleva décadas intentando resolver con cada vez mayor precisión.
El campo de aplicación más antiguo es sin duda alguna la navegación y posteriormente la aeronáutica, siendo en estos campos en los que se hicieron grandes avances dada la importancia de los mismos y la cantidad de recursos económicos que se dispusieron en tales materias históricamente. Otros campos de aplicación son la medicina y los videojuegos, aunque gracias a los avances tecnológicos y el bajo coste que tiene actualmente la tecnología necesaria para estimar la orientación y posición sin un gran grado de exactitud han aparecido miles de posibles aplicaciones.
Antiguamente se utilizaban giróscopos y acelerómetros mecánicos aunque en este proyecto la atención se centrará en los denominados IMUs que no son más que la integración en una misma unidad de tres giróscopos y tres acelerómetros dispuestos de forma ortogonal.
Algunos ejemplos de productos y aplicaciones donde se utilizaron o se utilizan aun giróscopos y acelerómetros son los siguientes:

-Cohete V2 fabricado en la segunda guerra mundial por un equipo de técnicos del ejército nazi. Su primer lanzamiento exitoso se produjo en 1942 y funcionaba de la siguiente forma, después del lanzamiento dos giróscopos y acelerómetros integrados inclinaban el cuerpo del misil en el ángulo necesario y controlaban el motor principal a la velocidad necesaria, de forma que su trayectoria balística asegurase alcanzar al objetivo.

Ilustración 1: Cohete V2

-Horizonte artificial empleado en la aviación, se utilizó por primera vez en 1929 y fue inventado por Elmer Sperry. El horizonte artificial está formado por un giróscopo de rotación horizontal, que está montado sobre un sistema de ejes que le permiten tres grados de libertad, todo esto está en el interior de una caja hermética. El giróscopo ha de girar a gran velocidad, mediante una corriente de aire o mediante un pequeño motor eléctrico, este está unido a una esfera visible, dividida en dos hemisferios por la línea del horizonte (el superior de color azul, representando el cielo, y el inferior de color negro, simulando la tierra), los cuales están graduados con líneas

horizontales cada 5° por encima y por debajo de la línea del horizonte. Por delante de la esfera se coloca una representación de avión en miniatura (en forma de W o de omega invertida) que sirve para marcar sobre la esfera los grados de cabeceo de la aeronave.

Ilustración 2: Horizonte artificial

-Función de giro automático de la pantalla en cualquier Smartphone o Tablet. Cualquiera de estos dispositivos poseen actualmente varios acelerómetros con los cuales el dispositivo es capaz de saber en todo momento, mientras está en reposo, su orientación respecto al suelo.

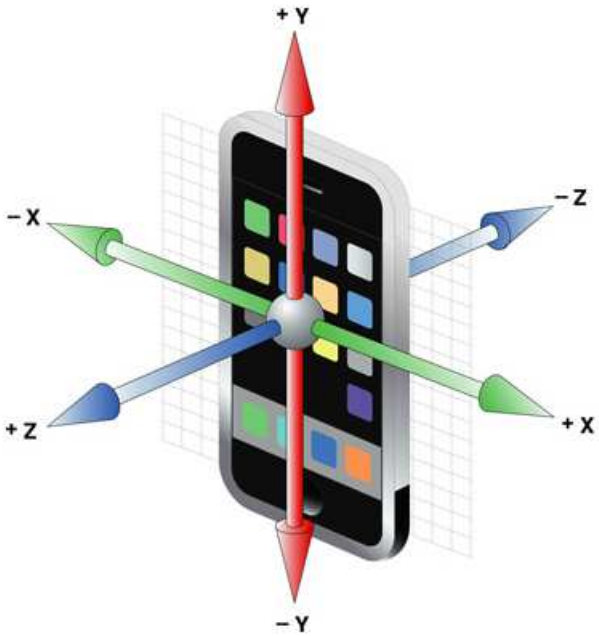


Ilustración 3:Acelerómetros en un Smartphone

-Mando de la videoconsola Wii de Nintendo, este mando en su día fue una revolución y su funcionamiento se basaba simplemente en la combinación de un acelerómetro y una cámara de vídeo junto con una barra de LEDs. El acelerómetro se indica con la flecha roja en la ilustración X.

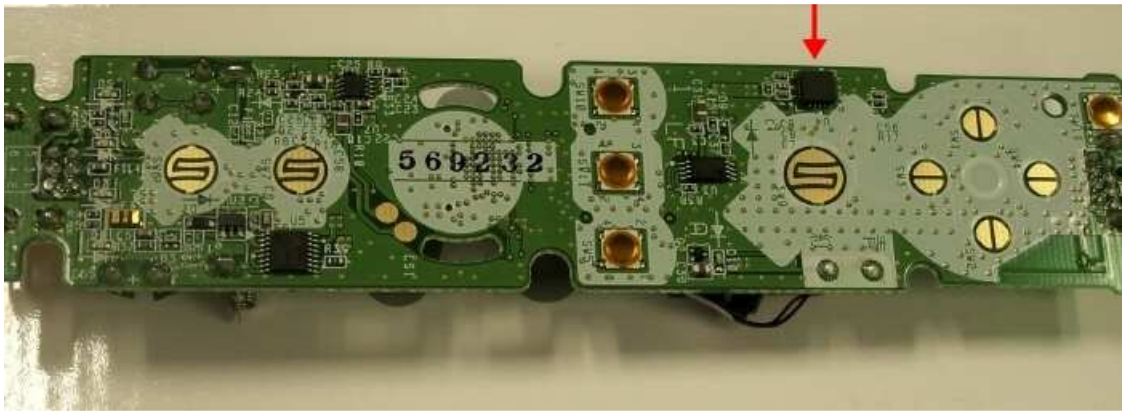


Ilustración 4: Mando de la Nintendo Wii

Como se ha visto los giróscopos y los acelerómetros se han utilizado desde campos tales como la aviación y el armamento hasta para productos de consumo en masa como son los Smartphones y las videoconsolas. Esto se debe a que con el avance de la tecnología han pasado a ser pequeños, robustos y económicos.
Además en la actualidad para ganar precisión y exactitud los giróscopos y acelerómetros se combinan con otros dispositivos como por ejemplo magnetómetros, sistemas barométricos, unidades GPS o sensores de velocidad externos.

1.1.Objetivo del proyecto

El objetivo de este proyecto es el diseño de un prototipo capaz de capturar y analizar ciertos movimientos que se realizan durante el entrenamiento de musculación. Al saber la velocidad de algunos movimientos se pueden hacer cálculos muy exactos e inmediatos de cuáles son los pesos más adecuados a usar en el entrenamiento.
A lo largo del transcurso del proyecto para lograr alcanzar el objetivo se han llegado a adquirir conocimientos en diversos campos, como los diversos métodos para calcular la orientación, la programación en Arduino, la implementación de sensores y módulos bluetooth, el prototipado rápido, la comunicación en serie, la calibración de sensores, etc
Siendo así el proyecto el escenario perfecto para tocar varios campos y fusionarlos finalmente obteniendo un resultado final que es una mezcla de todo, lo cual se considera muy apropiado para la filosofía de este máster en mecatrónica.

1.2.Fases del proyecto

La realización de este trabajo se podría dividir en varias fases. Inicialmente hubo una etapa muy importante y bastante extensa de documentación e investigación de que tipos de sensores se podían implementar y que métodos había para tratar los datos obtenidos por los sensores.
Posteriormente se empezó a trabajar con una unidad de Xsens con la que viene incluido su propio software y la cual tiene gran precisión. Con este sensor se empezó a gestar todo lo que sería a posteriori el código de procesamiento de datos, teniendo en cuenta que este sensor era algo muy refinado y al usar a posteriori sensores más económicos los resultados podrían no ser los mismos. Con el Xsens se siguieron todos los pasos principales que luego habría que aplicar con el sensor definitivo, pasos tales como obtención de datos, calibración del sensor, manejo de la unidad PAN-TILT, la cual describiremos posteriomente.
En definitiva el Xsens fue una plataforma estupenda de aprendizaje y permitió que se pudiera avanzar en el conocimiento del funcionamiento de los sensores al mismo tiempo que se empezaba a trabajar en la integración del sensor definitivo con la plataforma ARDUINO, y su conexión por bluetooth a un smartphone.
La tercera fase del proyecto fue implementar todo lo aprendido con el Xsens utilizando ya el montaje definitivo con ARDUINO y el sensor elegido.
Finalmente se diseño el soporte mecánico que permite integrar de forma cómoda todos los componentes del proyecto y que se adapta a las medidas de las barras olímpicas que es el lugar de trabajo del sensor.

2. HERRAMIENTAS UTILIZADAS

En este capítulo se presentan todos los medios de los que se ha dispuesto para la realización de este proyecto. Con un conocimiento previo de los medios utilizados se podrá comprender mejor el desarrollo de todo el proyecto. Se dividirán los medios en dos partes, software y hardware.

2.1. Software

Se han utilizado programas de muy diverso origen, desde programas para el diseño 3D de piezas como programas para interconectar diversas máquinas, pasando por programas de programación.

2.1.1. MT Manager

Es el programa que el propio fabricante del sensor Xsens suministra con sus sensores. Es un programa muy completo con múltiples opciones pero básicamente tiene dos funciones principales. Por un lado visualiza la posición del sensor a tiempo real tanto con gráficas como con una representación 3D del propio sensor y por otro es capaz de registrar los datos que suministra el sensor pudiéndose exportar para su posterior procesado con otro programa si así se desea.

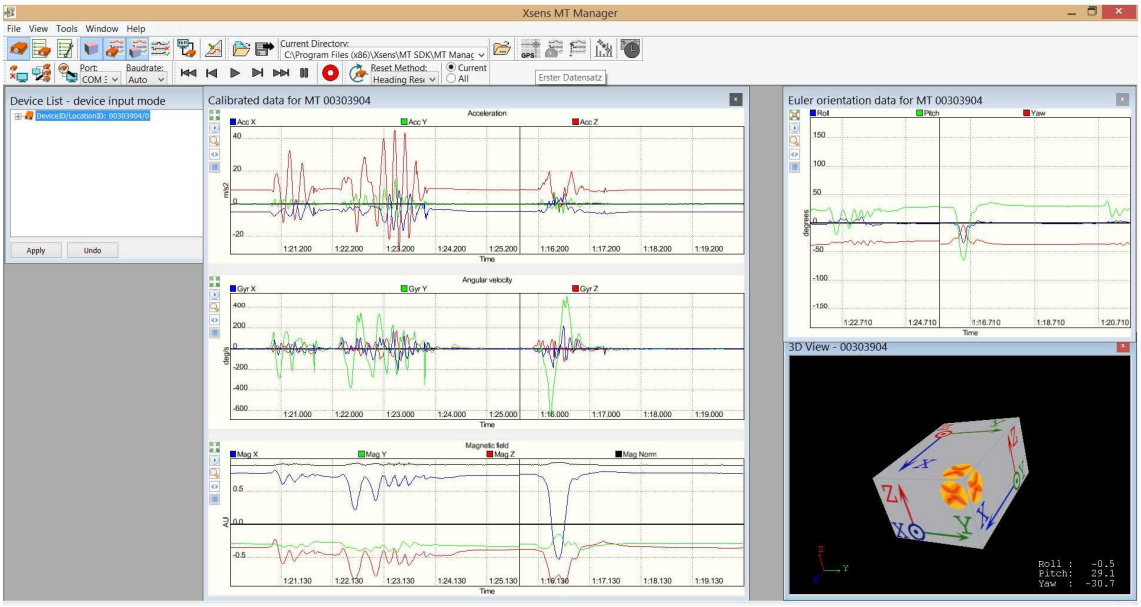


Ilustración 5: MT Manager

2.1.2. Matlab

es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M).Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware.
Se ha utilizado para tratar los datos obtenidos con el sensor. Se ejecuta el algoritmo y se calcula así la orientación del sensor, además con Matlab se pueden hacer gráficas que no sólo son importantes para representar los datos sino que son vitales a la hora de corregir el código, mejorar el algoritmo y en definitiva desarrollar el procesamiento de los datos.

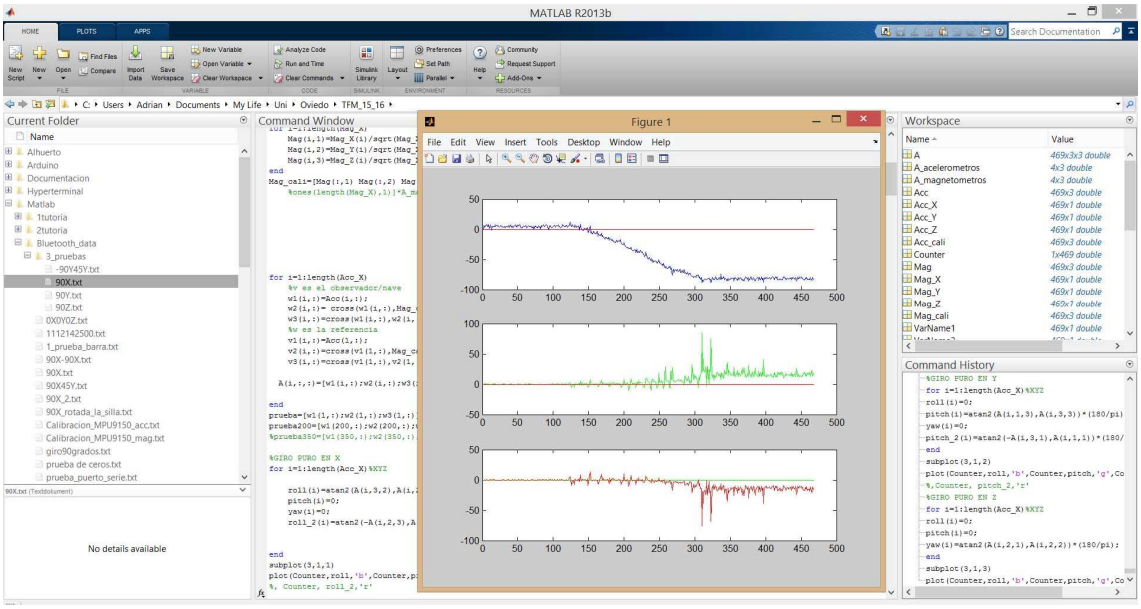


Ilustración 6: Matlab

2.1.3. ***SolidWorks***

Es un software CAD para modelado mecánico en 3D, desarrollado en la actualidad por SolidWorks Corp. El programa permite modelar piezas y conjuntos y extraer de ellos tanto planos técnicos como otro tipo de informaciones necesarias para la producción. Es un programa que funciona con base en las nuevas técnicas de modelado con sistemas CAD.
Con SolidWorks se diseñó el soporte para el sensor y el resto de componentes. El soporte consta de dos piezas las cuales se unen con 4 tornillos.
SolidWorks no sólo ofrece unas herramientas de edición simples y rápidas para el diseño en tres dimensiones de las piezas, sino que además ofrece la posibilidad de exportar los archivos de las piezas en formato .STL el cual es el necesario para poder imprimir las piezas con una impresora 3D mediante impresión aditiva.

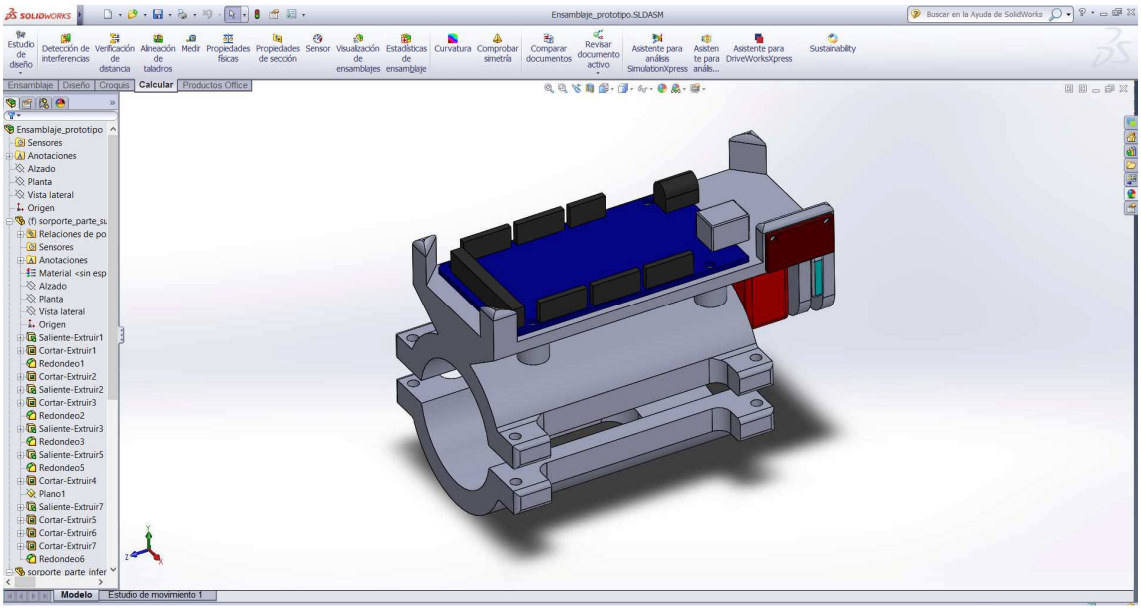


Ilustración 7: SolidWorks

2.1.4. Arduino (IDE)

Es un software open-source creado por los diseñadores de Arduino. Con este software se puede escribir el código para Arduino, depurarlo y compilarlo. En el caso de este proyecto ha sido utilizado con para escribir el código y compilarlo en un Arduino Mega 2560.
Cabe mencionar que este software ofrece la herramienta de un terminal serie lo cual fue de gran ayuda durante la fase de pruebas con el prototipo ya que se pudo obtener los datos del sensor directamente en el ordenador desde el terminal serie de Arduino.

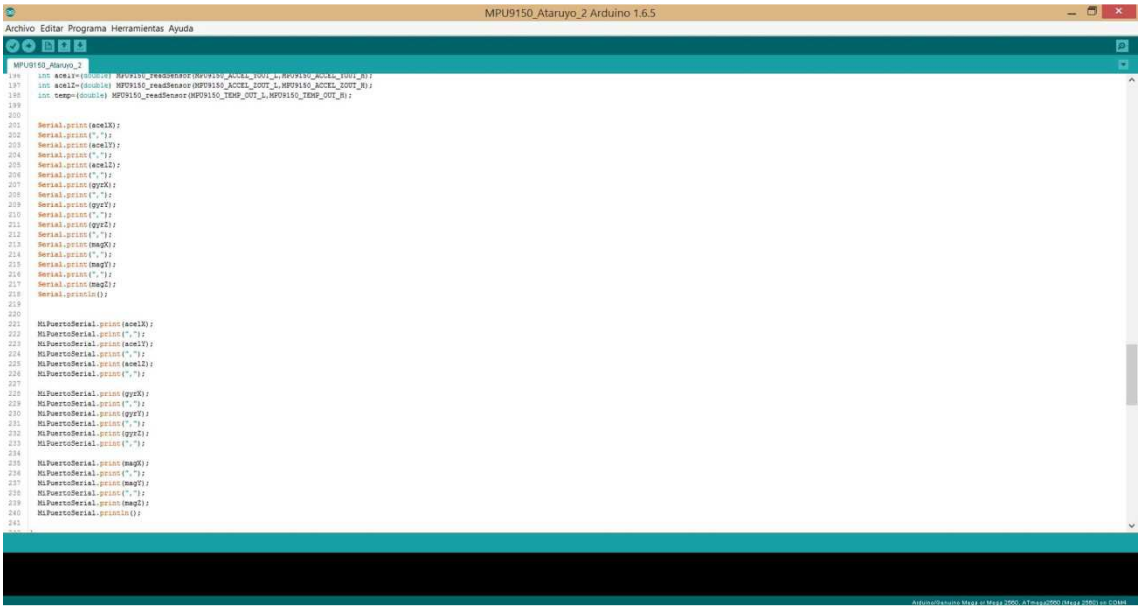


Ilustración 8: Arduino IDE

2.1.5. Bluetooth SPP

Software para la comunicación bluetooth en Android, con esta aplicación se logra conectar un Smartphone con Arduino y enviar así los datos obtenidos por los sensores. Esta aplicación es totalmente gratuita.

2.1.6. Hyperterminal

Es un programa de comunicación serie para Windows el cual permite conectar entre si varios ordenadores o ordenadores con otros dispositivos.

2.2. Hardware

Se han utilizado diversos dispositivos tanto para el desarrollo del prototipo como en su construcción final. A continuación se describen brevemente.

2.2.1. PanTilt PTU 46-17

El dispositivo Pan-Tilt se basa en una estructura con un par de motores paso a paso que permiten rotar una plataforma respecto a dos de sus ejes. Su uso más habitual es el de dar movimiento a cámaras de vigilancia aunque también se puede usar para posicionar objetos en el espacio con gran precisión, siendo este el uso dado para este proyecto.
Al montar el sensor en la unidad Pan-Tilt se puede situar el sensor en posiciones conocidas y moverlo a velocidades constantes conocidas.

El modelo Pan-Tilt utilizado para este proyecto ha sido el siguiente y tiene las siguientes especificaciones:

Ilustración 9: Pan-Tilt PTU 46-17

-capacidad de carga de 1,8Kg
-velocidad máxima de giro de 300°/s
-resolución de 0.514°

2.2.2. Giróscopos

Son dispositivos mecánicos que sirven para medir, mantener o cambiar la orientación en el espacio de algún aparato o vehículo.
Su principio de funcionamiento se basa en el principio de conservación del momento angular. Un giróscopo mecánico se constituye básicamente de un disco montado en un soporte cardánico de tal manera que pueda el disco girar libremente en cualquier dirección. Al estar el disco girando se produce que el mismo giro del disco hace que el disco no quiera modificar su plano de giro.
El giróscopo utilizado en este proyecto poco tiene que ver con lo descrito anteriormente ya que se han empleado giróscopos electrónicos y no mecánicos.
Lo giróscopos MEMS emplean una tecnología microelectrónica con un sensor de silicio que gracias a unas placas especiales que actúan a modo de condensadores son capaces de transformar cambios de intensidad en datos digitales y eso es lo que lee el microprocesador.

2.2.3. Acelerómetros

Un acelerómetro tal como su nombre indica mide la aceleración a la que se ve sometido. Puede medir aceleraciones estáticas tal como la de la gravedad o aceleraciones dinámicas como por ejemplo las del giro de una peonza.
Los acelerómetros empleados en este proyecto son del tipo MEMS, emplean tecnología microelectrónica. Están constituidos por unos dispositivos electromecánicos construidos usualmente con una base de silicio policristalino modelado que se miden en micrómetros y su arquitectura es bastante simple.

Un acelerómetro se encuentra constituido por una serie de estructuras similares a agujas, que detectan el movimiento y pueden transmitir estos datos a un circuito mayor que las utiliza y las registra. El coste de fabricar estos acelerómetros actualmente es inferior a un euro por lo que cada vez se están utilizando más y empleando en mayor número de aplicaciones. Se puede ver el tamaño al microscopio de un acelerómetro comparado con el tamaño de un ácaro.

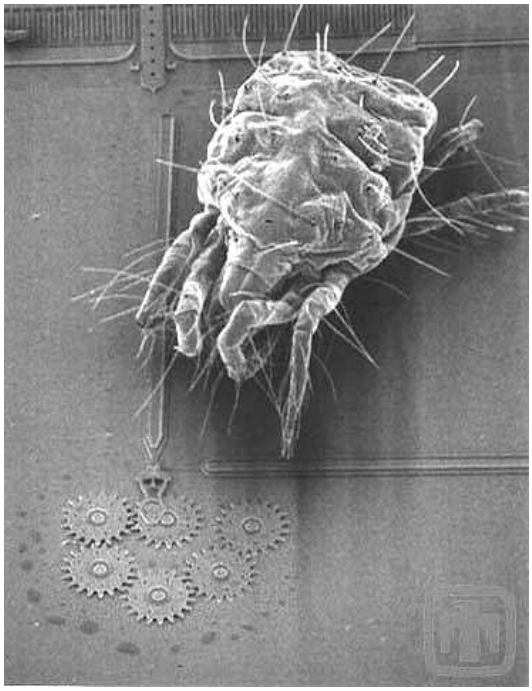


Ilustración 10: Acelerómetro visto en un microscopio

2.2.4. Magnetómetros

Los magnetómetros son dispositivos que sirven para medir el módulo y la dirección de la señal magnética de una muestra. Ejemplos antiguos son la balanza de Gouy o la balanza de Evans que basan su principio en medir el cambio en peso aparente que se produce en una muestra al aplicar un campo magnético.
Los magnetómetros utilizados en este proyecto poco que ver tienen con los descritos anteriormente. Los utilizados por el prototipo incorporan tecnología microelectrónica y su funcionamiento consiste en medir el campo que atraviesa unas magnetos-resistencias las cuales están alineadas con los ejes.

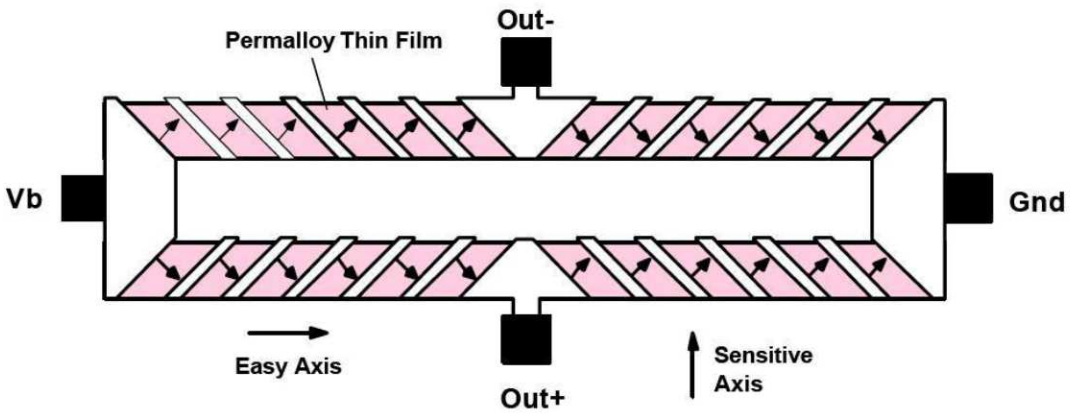


Ilustración 11: Esquema del funcionamiento de un magnetómetro

2.2.5. ***Xsens MTI***

Este dispositivo está formado por un IMU, 3 magnetómetros y un sensor de temperatura.
Es un dispositivo bastante caro ya que detrás de su hardware hay un software con muchas horas de trabajo, teniendo Xsens un nivel de precisión en la orientación muy alto.
Este dispositivo se conecta al ordenador a través de un cable con conexión USB y se maneja con su propio software descrito ya anteriormente, el MT Manager.

Ilustración 12 Sensor Xsens MTI

2.2.6. ***MPU 9150***

Un MPU 9150 consta de un IMU, es decir de tres acelerómetros y tres giróscopos, además de tres magnetómetros y un sensor de temperatura. Es básicamente lo mismo que el Xsens sólo que no mucho más barato y menos preciso además de no tener ningún software detrás ni ningún algoritmo que nos permita interpretar los datos en crudo que el conjunto de todos sus sensores nos da.
Algunas de las características más notables son las siguientes:
-El giróscopo de tres ejes tiene una sensibilidad de hasta 131 LSB/dps y una gama completa de ±250, ±500, ±1000 y ±2000 °/s.
-El acelerómetro de tres ejes tiene un rango de escala completa y programable de ±2g, ±4g, ±8g y ±16g.
-Magnetómetro de tres ejes con un rango de escala completa de ±1200 µT

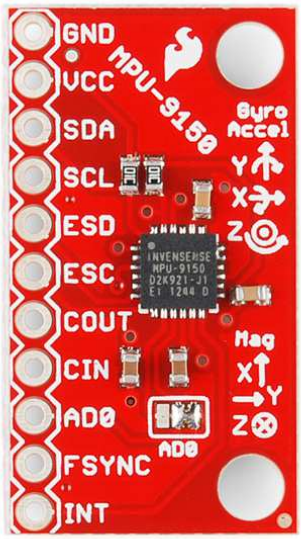


Ilustración 13: Sensor MPU 9150

2.2.7. ***ARDUINO MEGA2560***

El Arduino Mega2560 tiene el microcontrolador más potente de todos los Arduinos y como el resto de Arduinos cuenta con todo lo necesario para llevar a cabo comunicaciones y poder controlar todo tipo de dispositivos tales como motores, LEDs, sensores, etc
La comunicación entre el ordenador y Arduino se produce a través del puerto serie, sin embargo posee un convertidor USB-serie, por lo que sólo se necesita conectar el dispositivo al ordenador utilizando un cable USB como el que utilizan las impresoras.
El Arduino Mega2560 tiene las siguientes especificaciones:
Microcontrolador: ATmega2560
Voltaje Operativo: 5V
Voltaje de Entrada: 7-12V
Voltaje de Entrada(límites): 6-20V
Pines digitales de Entrada/Salida: 54 (de los cuales 15 proveen salida PWM)
Pines análogos de entrada: 16
Corriente DC por cada Pin Entrada/Salida: 40 mA
Corriente DC entregada en el Pin 3.3V: 50 mA
Memoria Flash: 256 KB (8KB usados por el bootloader)
SRAM: 8KB
EEPROM: 4KB
Clock Speed: 16 MHz

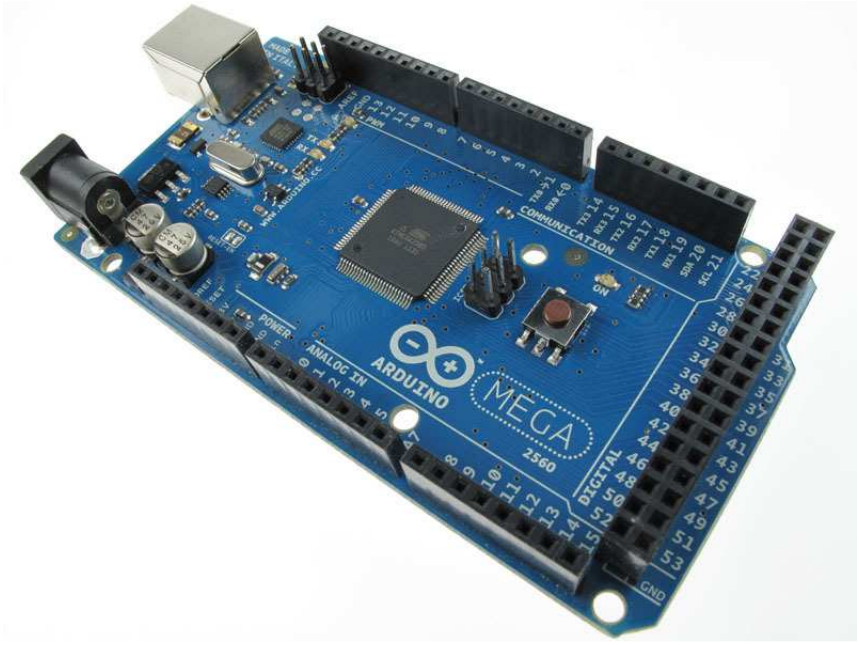


Ilustración 14: Arduino Mega2560

2.2.8. ***Bluetooth HC-06***

El módulo Bluetooth HC-06 es un dispositivo de tamaño reducido, aproximadamente del tamaño de un USB, económico, cuesta unos 5€ y sencillo de configurar y usar.
El módulo HC06 tiene bastantes diferencias con el módulo HC05 que es menester aclarar ya que suele haber bastantes confusiones. El modelo HC-06 dispone de 4 pines, en lugar de los 6 que incluye el modelo HC-05, pero hay además otras importantes diferencias de funcionalidad.
Básicamente el modelo HC-06 sólo puede actuar como esclavo y además dispone de un juego reducido de instrucciones que entiende, mientras que el modelo HC-05 puede actuar como master o como esclavo y acepta un número mayor de órdenes de configuración.
El módulo se configura mediante comandos AT, de ahí su simple configuración.

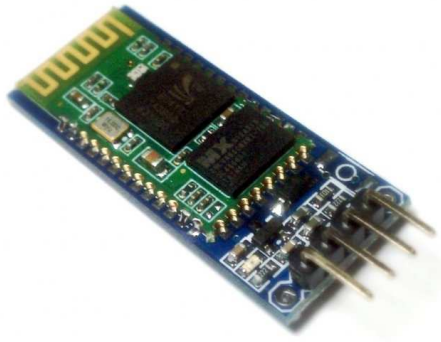


Ilustración 15

3. REPRESENTACIÓN DE LA ORIENTACIÓN ESPACIAL

La localización y orientación de un cuerpo es algo que tiene mucho interés en muchos campos. En este capítulo se introducirán conceptos básicos sobre la orientación.
La orientación de un cuerpo puede ser descrita por un vector de dirección y un ángulo, en esta sencilla idea se basa la fórmula de Rodrigues. Otra forma de representar la orientación es mediante una matriz de rotación, siendo este el caso en el que se va a centrar este capítulo ya que es como el prototipo calculará la rotación.
En la *ilustración 16* se puede ver como son las matrices de rotación en cada uno de los ejes y en la *ilustración 17* se ve una matriz de rotación general teniendo en cuenta el ángulo girado en todos los ejes y considerando un orden determinado en el giro de los ejes.

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Ilustración 16: Matrices de rotación en cada eje

$$\begin{bmatrix} C\phi C\theta & -S\theta C\alpha + C\theta S\phi S\alpha & S\theta S\alpha + C\theta S\phi C\alpha \\ S\theta C\phi & C\theta C\alpha + S\theta S\phi S\alpha & -C\theta S\alpha + S\theta S\phi C\alpha \\ -S\phi & C\phi S\alpha & C\alpha C\phi \end{bmatrix}$$

Ilustración 17: Matriz de rotación general

Para interpretar una matriz de rotación es de vital importancia saber con qué criterio concreto se debe interpretar su información. Hay dos criterios diferenciados el criterio de los ángulos fijos y el de los ángulos de Euler.

3.1.Ángulos fijos

La rotación se lleva a cabo considerando que cada giro en cada uno de los tres ejes se hace respeto al eje X al eje Y y al eje Z inicial.

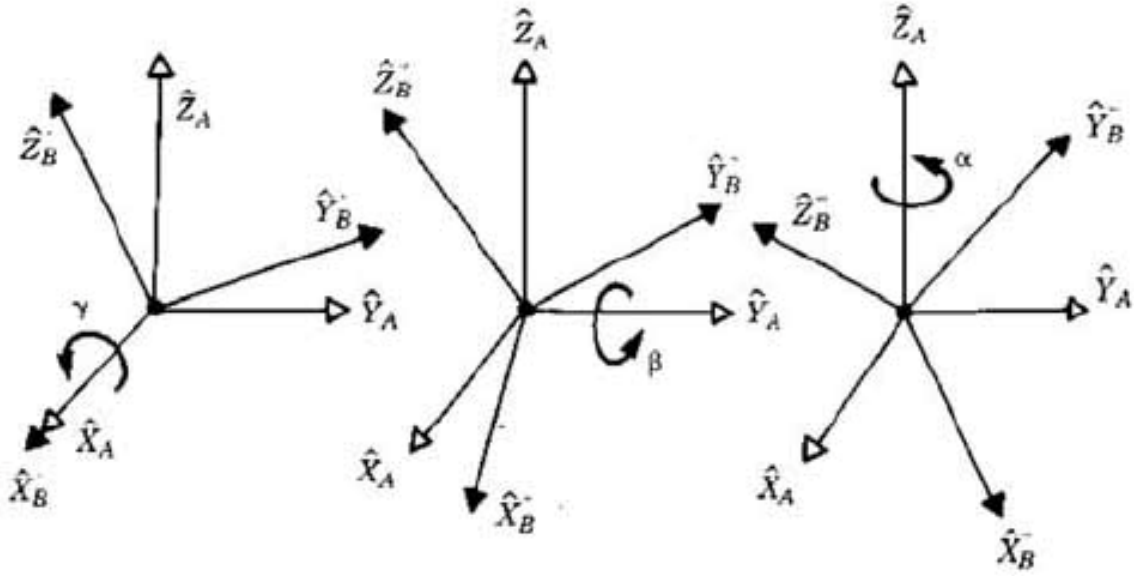


Ilustración 18: Esquema ángulos fijos

Cabe destacar que el orden de los giros es importante ya que no es equivalente rotar sobre el eje X luego sobre el Y y luego sobre el Z, por ejemplo, a hacerlo en cualquier otro orden

3.2.Ángulos de Euler

La rotación se lleva a cabo de forma tal que tras girar respecto a un eje el siguiente giro que se vaya a hacer se hace respecto a uno de los nuevos ejes obtenidos con el anterior giro.
Como con los ángulos fijos con los ángulos de Euler se tiene que tener en cuenta también el orden en que se llevan a cabo los giros, no se obtiene la misma orientación rotando respecto a X, luego respecto a Y y luego respecto a Z que en con cualquier otro orden.

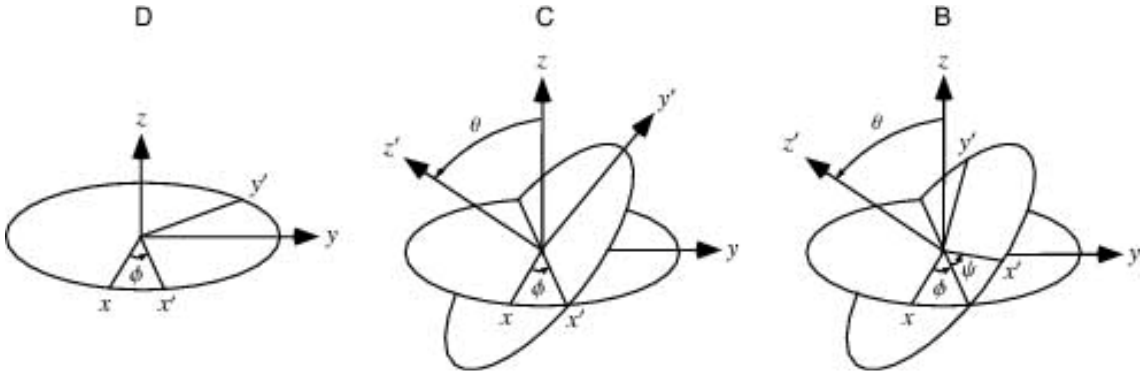


Ilustración 19: Esquema ángulos de Euler

Es menester mencionar que los ángulos de Euler están relacionados con los ángulos fijos ya que hay una equivalencia entre ellos. La matriz de rotación expresada con los ángulos de Euler con cierto orden de los giros es la misma matriz que con el orden inverso de giros en ángulos fijos.

3.3.Obtención de los ángulos a través de la matriz de rotación

Los ángulos rotados en cada eje se deducen de la matriz de rotación. Si se tiene la siguiente matriz de rotación

$$R_{X'Y'Z'}(\alpha,\beta,\gamma) = \begin{bmatrix} cos\beta \cdot cos\gamma & -cos\beta \cdot sen\gamma & sen\beta \\ sen\alpha \cdot sen\beta \cdot cos\gamma + cos\alpha \cdot sen\gamma & -sen\alpha \cdot sen\beta \cdot sen\gamma + cos\alpha \cdot cos\gamma & -sen\alpha \cdot cos\beta \\ -cos\alpha \cdot sen\beta \cdot cos\gamma + sen\alpha \cdot sen\gamma & cos\alpha \cdot sen\beta \cdot sen\gamma + sen\alpha \cdot cos\gamma & cos\alpha \cdot cos\beta \end{bmatrix}$$

Los ángulos α, β, γ se deducen de la siguiente forma.
En este caso se comienza por el ángulo β

$$\beta = atan2\left(\frac{pos(1,3)}{\sqrt{(pos(1,1)^2 + pos(1,2)^2)}}\right) = atan2\left(\frac{sen\beta}{\sqrt{((cos\beta \cdot cos\gamma)^2 + (-cos\beta \cdot sen\gamma)^2)}}\right)$$

$$\alpha = atan2\left(\frac{pos(2,3)/-\cos\beta}{pos(3,3)/\cos\beta}\right) = atan2\left(\frac{-sen\alpha \cdot cos\beta/-cos\beta}{cos\alpha \cdot cos\beta/cos\beta}\right)$$

$$\gamma = atan2\left(\frac{pos(1,2)/-\cos\beta}{pos(1,1)/\cos\beta}\right) = atan2\left(\frac{-cos\beta \cdot sen\gamma/-cos\beta}{cos\beta \cdot cos\gamma/cos\beta}\right)$$

Para calcular el primer ángulo se utiliza directamente su seno por lo tanto interesa que el ángulo inicial a calcular sea un ángulo cercano a 90° para que su seno sea cercano a uno. Si el ángulo es cercano a cero el seno será muy cercano a cero también, lo cual será perjudicial para calcular correctamente y con precisión el ángulo.

Por otro lado no se puede elegir que ángulo a calcular en primer lugar siempre, sino que esto es algo que viene condicionado por el convenio de Euler que se haya elegido siendo así de especial interés si se sabe el movimiento que se va a efectuar elegir el convenio de orden de giros más adecuado.

Siguiendo con la matriz de rotación del ejemplo anterior se puede observar que habría otra manera de calcular el ángulo β. La otra forma sería la siguiente.

$$\beta = atan2\left(\frac{pos(1,3)}{\sqrt{(pos(2,3)^2 + pos(3,3)^2)}}\right) = atan2\left(\frac{sen\beta}{\sqrt{((-sen\alpha \cdot cos\beta)^2 + (cos\alpha \cdot cos\beta)^2)}}\right)$$

Las matrices de rotación que se obtienen distan de ser perfectas por lo que es de interés elegir que valores de la matriz de rotación se eligen para calcular los ángulos siempre que se disponga de más de una opción. Si se puede optar siempre se cogerá el valor más alejado de cero ya que como ya se ha explicado debido a los cosenos cercanos a 90° y los senos cercanos a 0° se suele perder mucha precisión en el cálculo.

4. CALIBRACIÓN DE SENSORES INERCIALES Y MAGNETÓMETROS

Cualquier sensor tiene siempre un error el cual es debido a que es imposible fabricar sensores perfectos. Para poder disminuir los errores y tomar medidas más precisas se calibran los sensores.
Dado que tanto el sensor Xsens como el MPU disponen de acelerómetros, de giróscopos y de magnetómetros se deberán llevar a cabo 3 calibraciones una para cada grupo.
La idea básica de lo que hace la calibración es corregir las medidas tomadas por el sensor para aproximarlas más a la realidad.
Para calibrar tanto los acelerómetros como los magnetómetros se coloca el sensor en posiciones conocidas donde se sabe que valores teóricos debería dar el sensor y se compara con los que realmente da el sensor.
Para los giróscopos el concepto es el mismo pero como que estos lo que miden son una velocidad angular lo que se hace no es colocarlos en posiciones conocidas sino rotar el sensor a velocidades angulares conocidas para comparar el dato que da el sensor con la velocidad angular a la que este está sometido, que es conocida.
Para hacer estas calibraciones se usará el Pan-Tilt, ya descrito en el capítulo anterior, concretamente en el subcapítulo de hardware. Como en la *ilustración 20* se puede observar al Pan-Tilt se le ha dotado de una tabla de madera cuyo único fin es el de poder alejar el sensor de los motores para que estos interfieran menos en los magnetómetros del sensor.

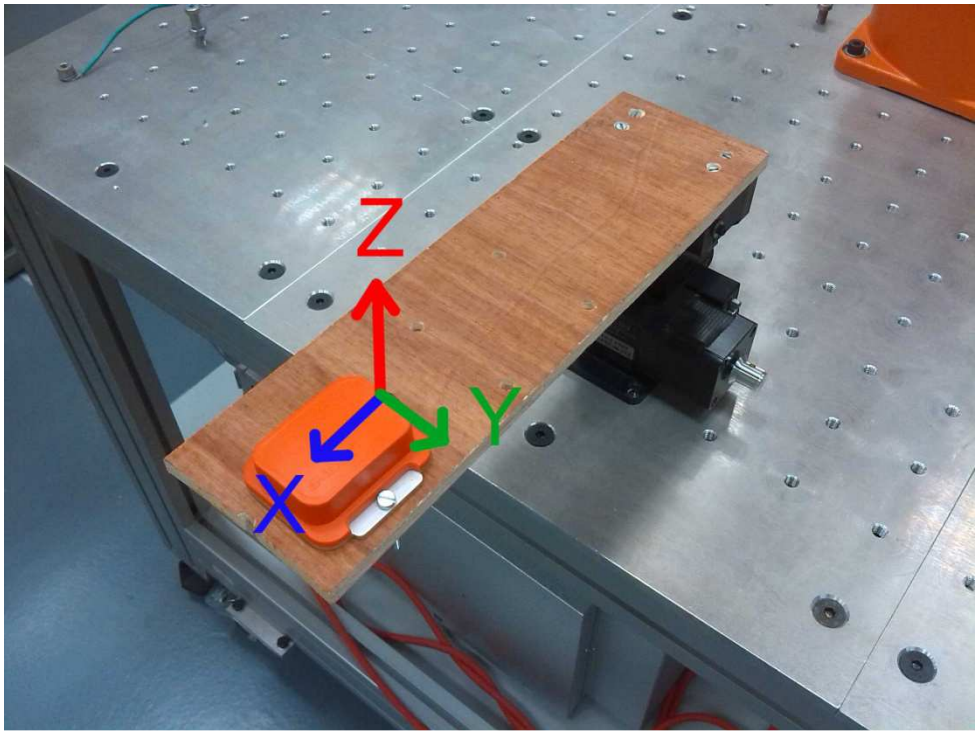


Ilustración 20: Representación ejes del sensor de Xsens

En la *ilustración 20* también se pueden ver los ejes del sensor. Como que el Pan-Tilt tiene solamente dos motores sólo dispone de la posibilidad de rotar el sensor de forma pura respecto a dos ejes, el Z y el Y, sin embargo se logra rotar el sensor respecto al eje X con un movimiento combinado sobre el eje Y y Z.

4.1. Preparación del Pan Tilt

La unidad Pan-Tilt de la que se dispuso no disponía del transformador para conectar directamente a la red, ya que este es accesorio. Por lo tanto hubo que mirar en el manual de usuario como se debían conectar los pines del conector para la alimentación.
En este apartado se va a describir la conexión correcta con el fin de dejar constancia para su posible futuro uso.

De los 5 pines del conector sólo se van a utilizar 3, los cuales son los siguientes que se ven en la *ilustración 21*.

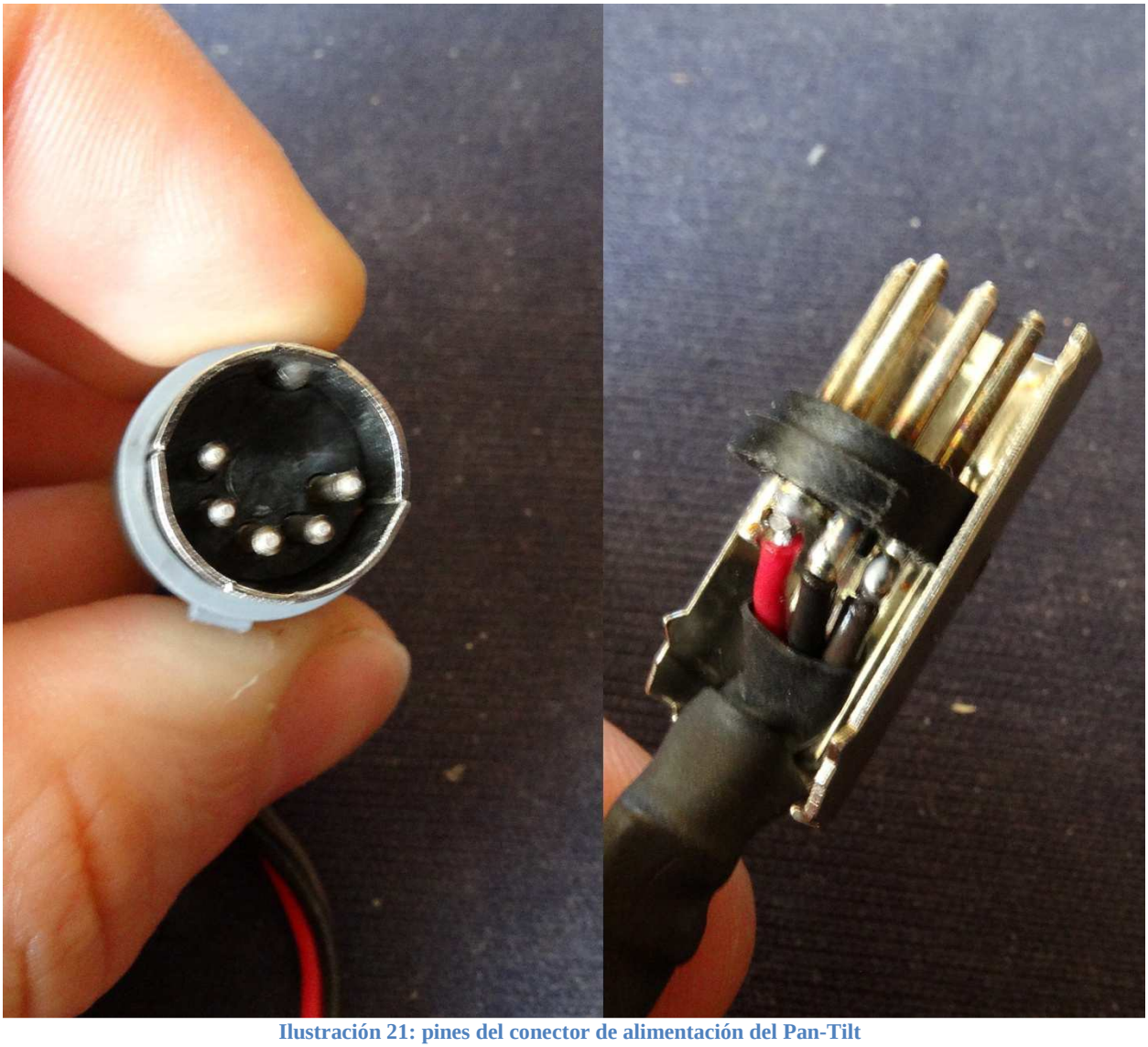


Ilustración 21: pines del conector de alimentación del Pan-Tilt

El cable rojo va a +5V, el cable marrón va a +30V (si es que se desea llegar a poder rotar el Pan-Tilt a la mayor velocidad posible) y el cable negro a masa.
En la *ilustración 22* se puede ver cómo están conectados a una fuente de alimentación de la forma descrita anteriormente, uniendo con un cable gris ambas masas.

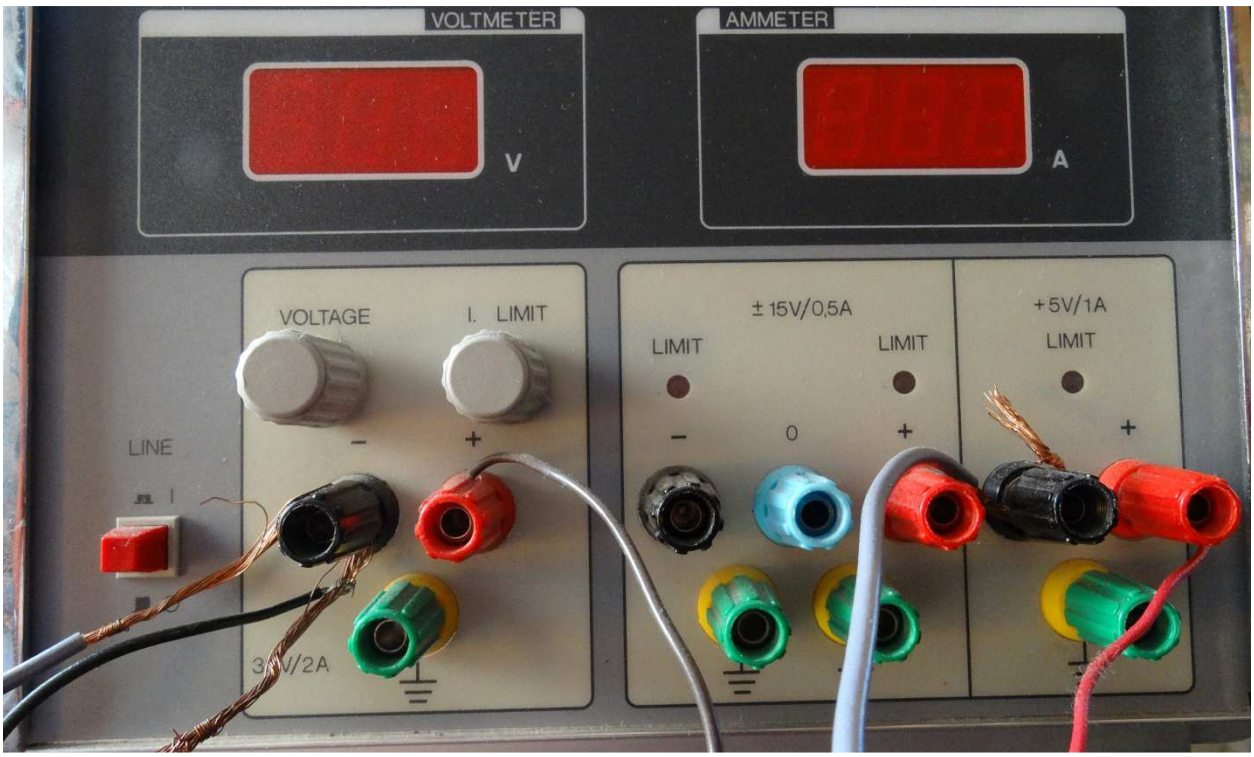


Ilustración 22: Conexiones para alimentar el Pan-Tilt

4.2. Procedimiento matemático para la calibración

Un sensor da una medida "*y*" la cual se compone de la siguiente forma a partir de un factor de escala y un offset $y=sx+o$, siendo "*s*" el factor de escala y "*o*" el offset.
Además se debe tener en cuenta que el sensor siempre viene montado en una placa por lo que la alineación no perfecta de los ejes del sensor y los de la placa va a introducir un error. Teniendo en cuenta la desviación de los ejes se introduce un nuevo parámetro.
En la *ilustración 23* se pueden observar los dos pares de ejes uno perteneciente al sensor y el otro perteneciente a la placa o al soporte donde va montado el sensor.

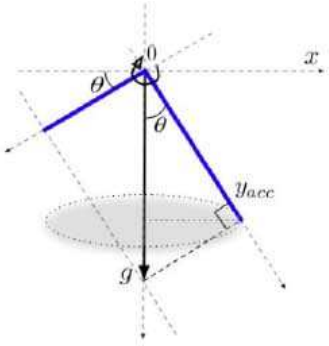


Ilustración 23: Pares de ejes

Por lo tanto se obtiene la ecuación $y=s(x\cos\Theta)+o=s(xu)+o$.
Ahora resta colocar el sensor en posiciones conocidas y tomar los datos que da, con este procedimiento ya se está en disposición de despejar los valores de la calibración. En el siguiente subcapítulo se puede ver paso a paso el procedimiento sobre el ejemplo de la calibración de los acelerómetros.

4.3.Calibración de los acelerómetros

La ecuación *y= s(xu)+o* corresponde al modelo tridimensional que es el sensor real. Como que se trata de un sistema de ecuaciones con más ecuaciones que incógnitas se emplea el método de mínimos cuadrados para obtener la solución con el error más pequeño.
La medida que entrega cada sensor se puede expresar con la siguiente ecuación:

$$y = s\cdot [a_x \quad a_y \quad a_z] \cdot \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} + o = [a_x \quad a_y \quad a_z \quad 1] \cdot \begin{bmatrix} su_z \\ su_y \\ su_z \\ 1 \end{bmatrix}$$

En cada posición se conocen las lecturas de los 3 acelerómetros las cuales se pueden representar de la siguiente forma

$$[a_x \quad a_y \quad a_z \quad 1]$$

Ahora simplemente se dispone de este sistema

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} a_{x1} & a_{y1} & a_{z1} & 1 \\ a_{x2} & a_{y2} & a_{z2} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ a_{xm} & a_{ym} & a_{zm} & 1 \end{bmatrix} \begin{bmatrix} su_x \\ su_x \\ su_x \\ 0 \end{bmatrix}$$

$$y = A\cdot \Theta$$

El cual se resuelve con la siguiente ecuación, obteniendo así la matriz de calibración.

$$\Theta=(A^T \cdot A)^{-1} \cdot A^T \cdot Y$$

Mediante el Pan-Tilt se coloca el sensor en 12 posiciones conocidas, en la *ilustración 24* se pueden ver las 12 posiciones.

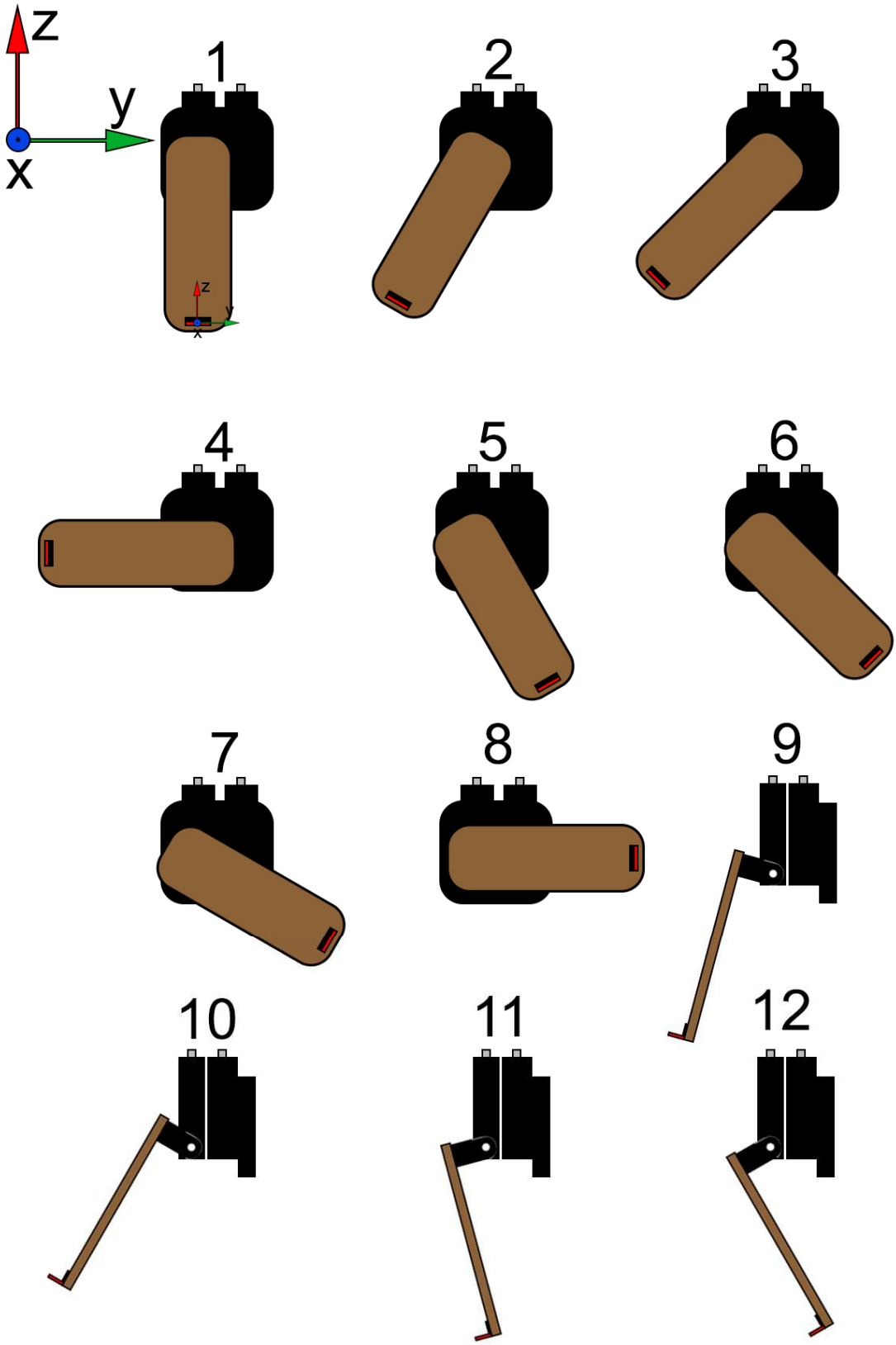


Ilustración 24: Posiciones para calibrar los acelerómetros

En la posición 1el sensor tiene su eje Z alineado con la perpendicular

En la posición 2 se rota -30 grados respecto al eje X

En la posición 3 se rota -45 grados respecto al eje X

En la posición 4 se rota -90 grados respecto al eje X

En la posición 5 se rota 30 grados respecto al eje X

En la posición 6 se rota 45 grados respecto al eje X

En la posición 7 se rota 60 grados respecto al eje X

En la posición 8 se rota 90 grados respecto al eje X

En la posición 9 se rota -15 grados respecto al eje Y

En la posición 10 se rota -30 grados respecto al eje Y

En la posición 11 se rota 15 grados respecto al eje Y

En la posición 12 se rota 30 grados respecto al eje Y

Se obtiene la siguiente gráfica, en la cual se ven representadas claramente las 12 posiciones.

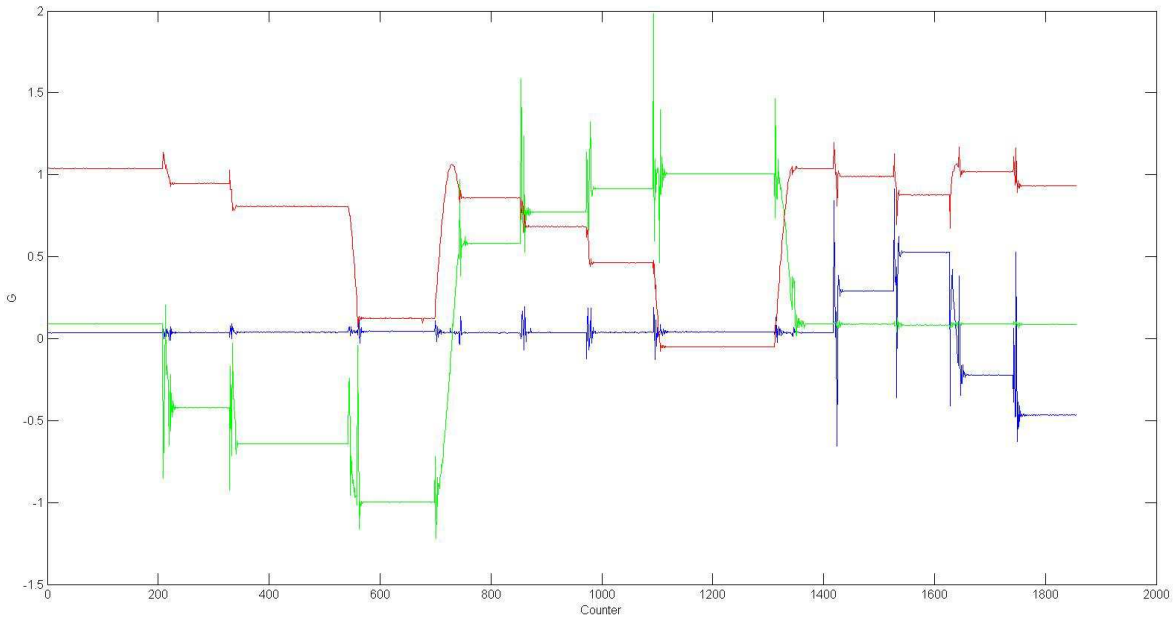


Ilustración 25: Datos obtenidos para las posiciones de calibración de los acelerómetros

Se construye la matriz Y con los valores teóricos de cada una de las 12 posiciones.

$$Y = \begin{bmatrix} 0 & 0 & 1 \\ 0 & -\sin\left(30 \cdot \frac{2\cdot\pi}{360}\right) & \cos\left(30 \cdot \frac{2\cdot\pi}{360}\right) \\ 0 & -\sin\left(45 \cdot \frac{2\cdot\pi}{360}\right) & \cos\left(45 \cdot \frac{2\cdot\pi}{360}\right) \\ 0 & -1 & 0 \\ 0 & \sin\left(30 \cdot \frac{2\cdot\pi}{360}\right) & \cos\left(30 \cdot \frac{2\cdot\pi}{360}\right) \\ 0 & \sin\left(45 \cdot \frac{2\cdot\pi}{360}\right) & \cos\left(45 \cdot \frac{2\cdot\pi}{360}\right) \\ 0 & \sin\left(60 \cdot \frac{2\cdot\pi}{360}\right) & \cos\left(60 \cdot \frac{2\cdot\pi}{360}\right) \\ 0 & 1 & 0 \\ \sin\left(15 \cdot \frac{2\cdot\pi}{360}\right) & 0 & \cos\left(15 \cdot \frac{2\cdot\pi}{360}\right) \\ \sin\left(30 \cdot \frac{2\cdot\pi}{360}\right) & 0 & \cos\left(30 \cdot \frac{2\cdot\pi}{360}\right) \\ -\sin\left(15 \cdot \frac{2\cdot\pi}{360}\right) & 0 & \cos\left(15 \cdot \frac{2\cdot\pi}{360}\right) \\ -\sin\left(30 \cdot \frac{2\cdot\pi}{360}\right) & 0 & \cos\left(30 \cdot \frac{2\cdot\pi}{360}\right) \end{bmatrix}$$

Se obtiene el valor medio para cada acelerómetro en cada posición y se genera la matriz *X* con los valores reales que entrega el sensor.

$$X = \begin{bmatrix} 0.0358 & 0.0914 & 1.0382 & 1 \\ 0.0373 & -0.4207 & 0.9477 & 1 \\ 0.0391 & -0.6423 & 0.8069 & 1 \\ 0.0471 & -0.9985 & 0.1237 & 1 \\ 0.0360 & 0.5820 & 0.8560 & 1 \\ 0.0366 & 0.7737 & 0.6841 & 1 \\ 0.0379 & 0.9151 & 0.4630 & 1 \\ 0.0412 & 1.0058 & -0.0502 & 1 \\ 0.2912 & 0.0888 & 0.9999 & 1 \\ 0.5268 & 0.0834 & 0.8767 & 1 \\ -0.2224 & 0.09095 & 1.0184 & 1 \\ -0.4666 & 0.08738 & 0.9316 & 1 \end{bmatrix}$$

Una vez se tiene esto se resuelve el sistema con MatLab mediante la fórmula ya vista anteriormente

$$A_{acelerometros} = (X^T \cdot X)^{-1} \cdot X^T \cdot Y$$

y se obtiene la matriz de calibración para los acelerómetros

$$A_{acelerometros} = \begin{pmatrix} 1.00772 & -0.00084 & 0.05531 \\ 0.00237 & 0.99048 & 0.08608 \\ 0.00933 & -0.08913 & 0.99131 \\ -0.04385 & -0.00001 & 0.03926 \end{pmatrix}$$

Teniendo de ejemplo esta matriz de rotación para los acelerómetros se puede ver que la diagonal marcada en rojo representa los valores para las ganancias, en azul se representan las desalineaciones entre los tres ejes y en verde se representan los valores de los offset.

$$A_{acelerometros} = \begin{pmatrix} 1.00772 & -0.00084 & 0.05531 \\ 0.00237 & 0.99048 & 0.08608 \\ 0.00933 & -0.08913 & 0.99131 \\ -0.04385 & -0.00001 & 0.03926 \end{pmatrix}$$

Para las matrices de rotación de los siguientes apartados esto es así también.

En la siguiente ilustración se pueden ver los resultados de la calibración, en azul los datos brutos del sensor y en rojo los datos ya calibrados.

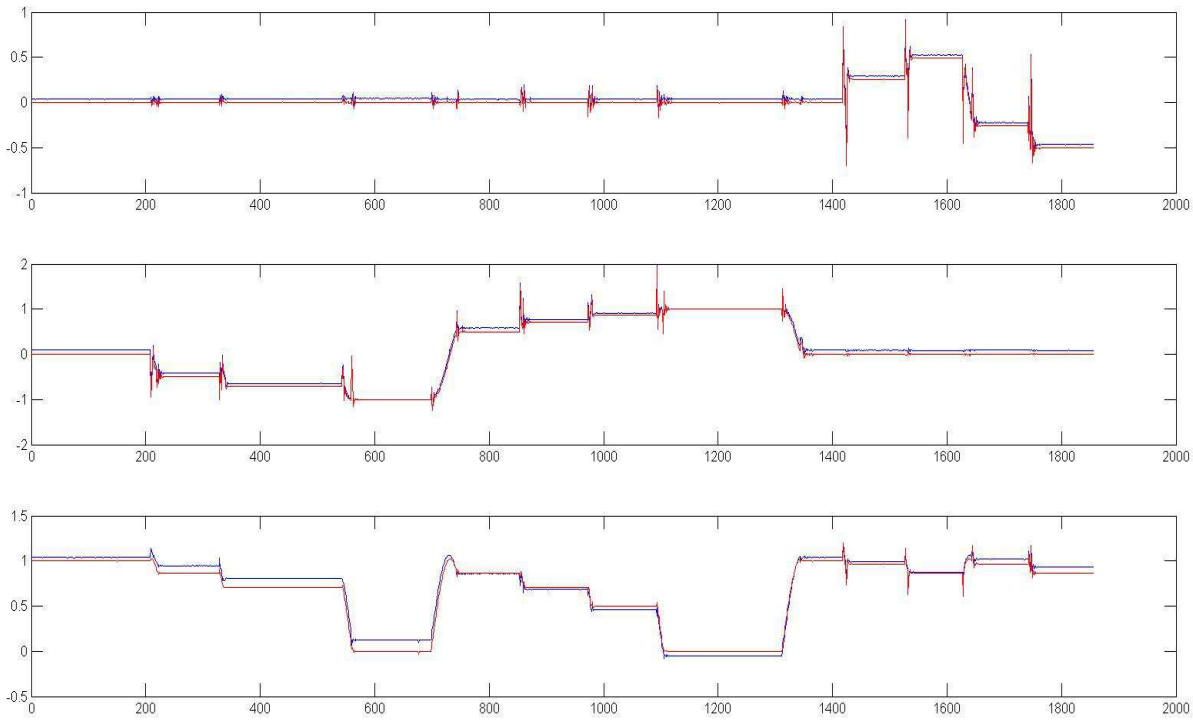


Ilustración 26: Datos en bruto y datos calibrados

4.4. Calibración de los giróscopos

Para calibrar los giróscopos se lleva a cabo el mismo procedimiento matemático que para calibrar los acelerómetros, la diferencia fundamental es que los acelerómetros se calibran de forma estática , se coloca el sensor en diversas posiciones y se leen los datos del sensor mientras que para los giróscopos , dado que estos miden velocidades angulares, se debe mantener el sensor en movimiento a una velocidad constante y conocida.
Con el Pan-Tilt se puede hacer rotar el sensor con una velocidad angular constante. Se programa el Pan-Tilt para que rote respecto a todos sus ejes a distintas velocidades y se toman los datos. En la *ilustración 27* se puede observar la gráfica resultante con sus 15 medidas distintas.

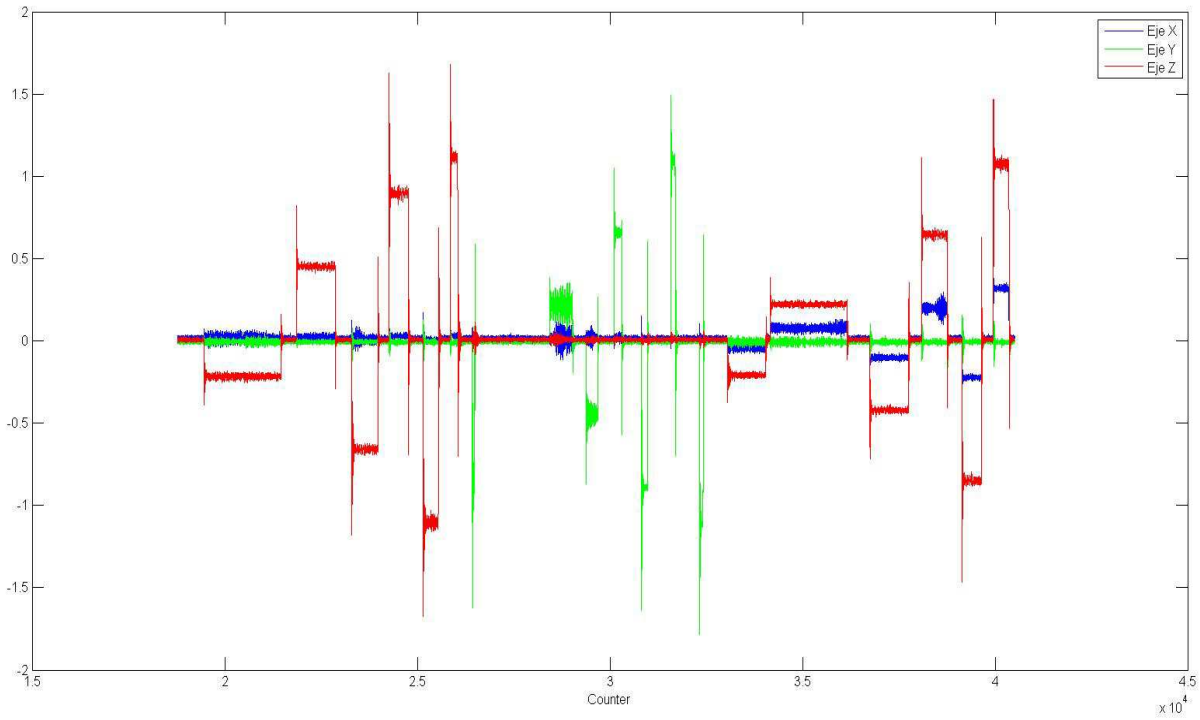


Ilustración 27: Datos obtenidos para las posiciones de calibración de los giróscopos

Al igual que antes se construye la matriz *Y* con los datos teóricos

$$Y = \begin{bmatrix} 0 & 0 & -0.2243 \\ 0 & 0 & 0.4485 \\ 0 & 0 & -0.6728 \\ 0 & 0 & 0.8971 \\ 0 & 0 & -1.1214 \\ 0 & 0.2243 & 0 \\ 0 & -0.4485 & 0 \\ 0 & 0.6728 & 0 \\ 0 & -0.8971 & 0 \\ 0 & 1.1214 & 0 \\ 0.0580 & 0 & 0.2166 \\ -0.1161 & 0 & -0.4333 \\ 0.1741 & 0 & 0.6499 \\ -0.2322 & 0 & -0.8665 \\ 0.2902 & 0 & 1.0832 \end{bmatrix}$$

Ahora se construye la matriz *X* con los valores obtenidos con los giróscopos.

$$X = \begin{bmatrix} 0.0153 & -0.0063 & -0.2163 & 1 \\ 0.0227 & -0.0073 & 0.4515 & 1 \\ 0.0103 & -0.0061 & -0.6572 & 1 \\ 0.0280 & -0.0083 & 0.8962 & 1 \\ 0.0040 & -0.0054 & -1.1025 & 1 \\ 0.0153 & -0.0063 & -0.2163 & 1 \\ 0.0186 & -0.4500 & 0.0049 & 1 \\ 0.0172 & 0.6558 & 0.0108 & 1 \\ 0.0194 & -0.8943 & 0.0024 & 1 \\ 0.0162 & 1.1001 & 0.0139 & 1 \\ 0.0782 & -0.0068 & 0.2220 & 1 \\ -0.1022 & -0.0065 & -0.4212 & 1 \\ 0.1981 & -0.0077 & 0.6468 & 1 \\ -0.2227 & -0.0056 & -0.8498 & 1 \\ 0.3180 & -0.0081 & 1.0757 & 1 \end{bmatrix}$$

Se despeja la matriz A de calibración por último

$$A_{giroscopos} = (X^T \cdot X)^{-1} \cdot X^T \cdot Y$$

$$A_{giroscopos} = \begin{pmatrix} 1{,}00845 & 0{,}03582 & 0{,}04741 \\ 0{,}00160 & 1{,}01038 & -0{,}0080 \\ -0{,}01193 & -0{,}01253 & 0{,}99604 \\ -0{,}01791 & 0{,}02102 & 0{,}00649 \end{pmatrix}$$

En la siguiente ilustración se pueden ver los resultados de la calibración, en azul los datos brutos del sensor y en rojo los datos ya calibrados.

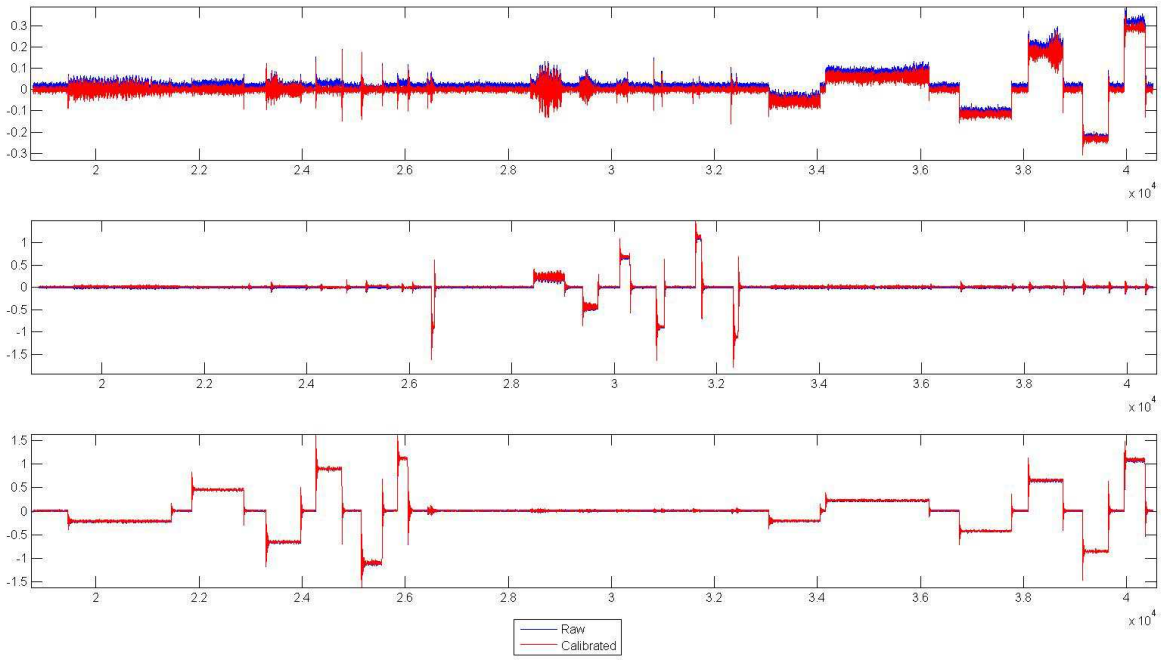


Ilustración 28: Datos en bruto y datos calibrados

Como se puede observar los datos brutos que dan los giróscopos son muy correctos ya que prácticamente los datos calibrados coinciden con los datos en bruto de los giróscopos.

4.5. Calibración de los magnetómetros

La calibración de los magnetómetros se realiza igual que la de los acelerómetros a pesar de haber una pequeña diferencia en cuanto a concepto.
Los valores que dan los magnetómetros son muy inestables, variando sus valores de forma aparentemente irracional. Esto se deduce que sucede al ser influenciados los magnetómetros por diversos aparatos de su entorno, a materiales férreos, etc.
Para disminuir este problema se normalizan los vectores correspondientes a los datos de los magnetómetros. Con esto se pierde el valor exacto del campo magnético, que no nos interesa para nada, y a cambio se reduce el problema comentado anteriormente.
Por otro lado se fija como campo magnético teórico el vector [0 0 1] dado que no se mantiene constante el campo magnético en la realidad ni se sabe con certeza hacía donde va.
El sensor se coloca en 9 posiciones diferentes, las cuales se puede ver en la ilustración X.

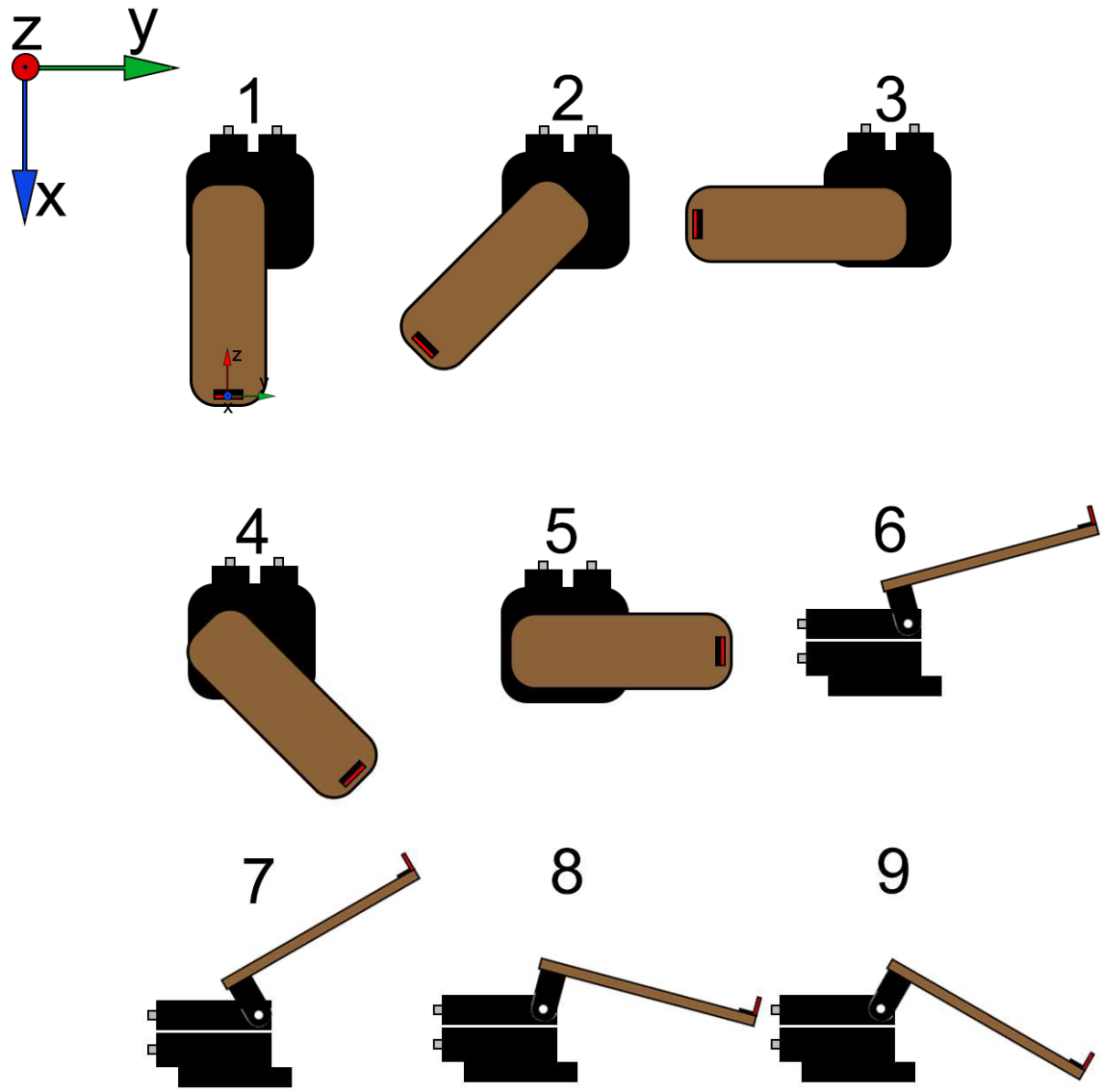


Ilustración 29: Posiciones para calibrar los magnetómetros

En la posición 1el sensor tiene su eje X alineado con la perpendicular

En la posición 2 se rota -45 grados respecto al eje X

En la posición 3 se rota -90 grados respecto al eje X

En la posición 4 se rota 45 grados respecto al eje X

En la posición 5 se rota 90 grados respecto al eje X

En la posición 6 se rota -15 grados respecto al eje Y

En la posición 7 se rota -30 grados respecto al eje Y

En la posición 8 se rota 15 grados respecto al eje Y

En la posición 9 se rota 30 grados respecto al eje Y

En la ilustración se pueden observar los datos proporcionados en bruto por los magnetómetros del sensor.

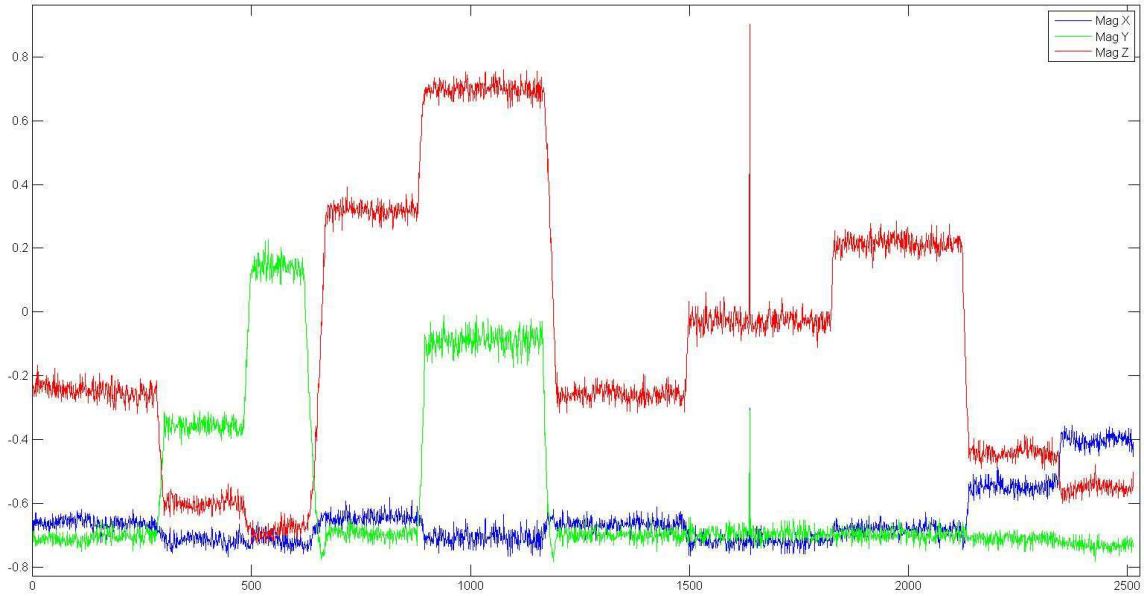


Ilustración 30: Datos obtenidos para las posiciones de calibración de los magnetómetros

Una vez tenido en cuenta lo anterior se procede como en los otros dos casos anteriores. Se construye la matriz *Y* con los datos teóricos.

$$Y = \begin{bmatrix} 0 & 0 & -1 \\ 0 & -\sin\left(45 \cdot \frac{2 \cdot \pi}{360}\right) & -\cos\left(45 \cdot \frac{2 \cdot \pi}{360}\right) \\ 0 & -1 & 0 \\ 0 & \sin\left(45 \cdot \frac{2 \cdot \pi}{360}\right) & -\cos\left(45 \cdot \frac{2 \cdot \pi}{360}\right) \\ 0 & 1 & 0 \\ \sin\left(15 \cdot \frac{2 \cdot \pi}{360}\right) & 0 & -\cos\left(15 \cdot \frac{2 \cdot \pi}{360}\right) \\ \sin\left(30 \cdot \frac{2 \cdot \pi}{360}\right) & 0 & -\cos\left(30 \cdot \frac{2 \cdot \pi}{360}\right) \\ -\sin\left(15 \cdot \frac{2 \cdot \pi}{360}\right) & 0 & -\cos\left(15 \cdot \frac{2 \cdot \pi}{360}\right) \\ -\sin\left(30 \cdot \frac{2 \cdot \pi}{360}\right) & 0 & -\cos\left(30 \cdot \frac{2 \cdot \pi}{360}\right) \end{bmatrix}$$

Ahora se construye la matriz *X* con los valores obtenidos con los magnetómetros.

$$X = \begin{bmatrix} -0{,}6610 & -0{,}7080 & -0{,}2460 & 1 \\ -0{,}7150 & -0{,}3564 & -0{,}6000 & 1 \\ -0{,}7126 & 0{,}1445 & -0{,}6852 & 1 \\ -0{,}6430 & -0{,}6955 & 0{,}3188 & 1 \\ -0{,}7069 & -0{,}0881 & 0{,}7002 & 1 \\ -0{,}7170 & -0{,}6954 & -0{,}0282 & 1 \\ -0{,}6815 & -0{,}6985 & 0{,}2155 & 1 \\ -0{,}5465 & -0{,}7104 & -0{,}4421 & 1 \\ -0{,}4017 & -0{,}7306 & -0{,}5507 & 1 \end{bmatrix}$$

Se despeja la matriz A de calibración por último

$$A_{magnetometros} = (X^T \cdot X)^{-1} \cdot X^T \cdot Y$$

$$A_{magnetometros} = \begin{pmatrix} -2{,}47015 & 2{,}41399 & 0{,}90901 \\ -0{,}34759 & -0{,}07048 & 1{,}30218 \\ 0{,}07704 & 1{,}2783 & 0{,}28014 \\ -1{,}75194 & 1{,}70345 & 0{,}60671 \end{pmatrix}$$

En la siguiente ilustración se pueden ver los resultados de la calibración, en azul los datos brutos del sensor y en rojo los datos ya calibrados.

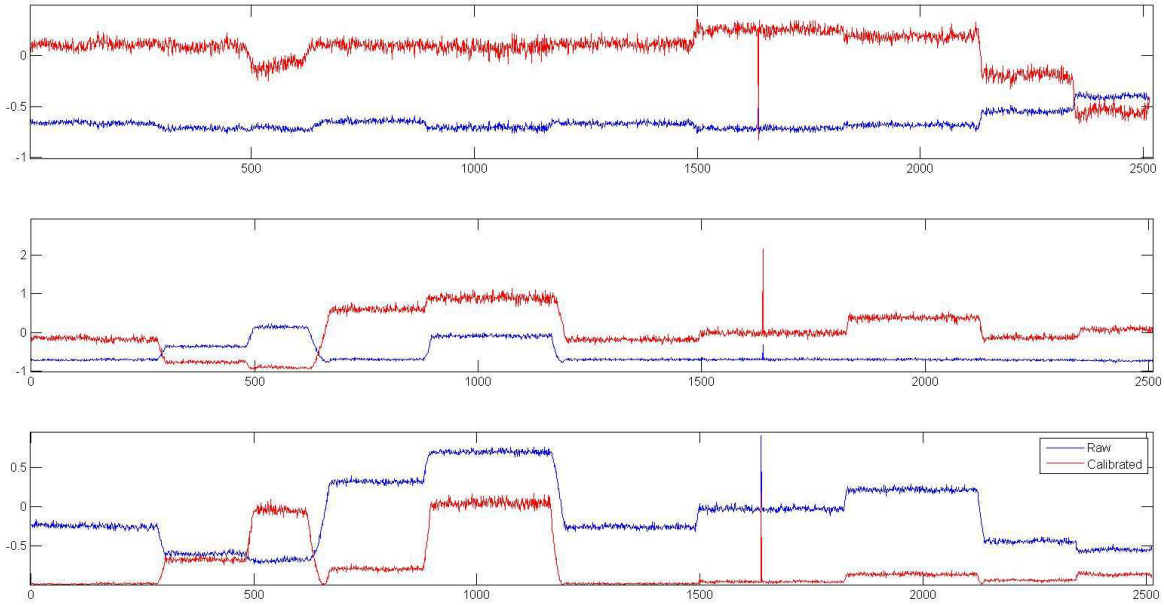


Ilustración 31: Datos en bruto y datos calibrados

5. ESTIMACION DE LA ORIENTACIÓN MEDIANTE TRIAD

5.1.Algoritmo TRIAD

El método de TRIAD es una de las soluciones más antiguas para determinar la orientación de una nave espacial. El algoritmo de TRIAD fue introducido en 1964 por el ingeniero Harold Black y se basa en la siguiente idea.
Se tienen dos sistemas de tres vectores ortogonales, uno de los dos sistemas se considera el sistema fijo o de referencia y el otro es el que varía su orientación. De esta forma se puede deducir los ángulos rotados de un sistema al otro teniendo en cuenta la orientación del segundo sistema respecto al de referencia.
Su desarrollo se debió a la necesidad de poder determinar la orientación de las naves y los satélites en el espacio.
Para poder aplicar TRIAD se necesitan los datos del acelerómetro de tres ejes y los datos del magnetómetro de tres ejes, con estos datos se llegan a componer 3 vectores los cuales son ortogonales y se comparan las orientaciones de la triada de vectores de referencia con la de los vectores que varían a lo largo del tiempo.
Aquí se pueden ver las ecuaciones matemáticas para obtener ambas triadas de vectores.
Primero para el marco de referencia

$$t_{1ref} = a_{ref}$$

$$t_{2ref} = \frac{a_{ref} \cdot m_{ref}}{|a_{ref} \times m_{ref}|}$$

$$t_{3ref} = t_{1ref} \times t_{2ref}$$

Después para el objeto en movimiento

$$t_{1obj} = a_{obj}$$

$$t_{2obj} = \frac{a_{obj} \cdot m_{obj}}{|a_{obj} \times m_{obj}|}$$

$$t_{3obj} = t_{1obj} \times t_{2obj}$$

Una vez se tienen este par de triadas se puede obtener la matriz de rotación con la siguiente ecuación:

$$A = \left[t_{1ref}\ t_{2ref}\ t_{3ref}\right] \left[\ t_{1obj}\ t_{2obj}\ t_{3obj}\right]^T$$

Como se puede observar mirando las fórmulas los datos que se utilizan para el primer vector es conveniente que sean los más fiables ya que tienen mayor peso, por eso se han elegido los datos de los acelerómetros y no los de los magnetómetros.

5.2. Precisión con el algoritmo TRIAD

Con el fin de ver la precisión del algoritmo TRIAD se toman datos del sensor estando parado sin movimiento alguno.
Los resultados son los que se pueden observar en la *ilustración 32*. En la ilustración se observan tres gráficas con idéntico resultado, cada una de las tres gráficas es la representación de los ángulos según diversos convenios de Euler por lo tanto se observa que la variación tan grande de unos 6 grados para el "roll" no depende del convenio sino de otros factores.

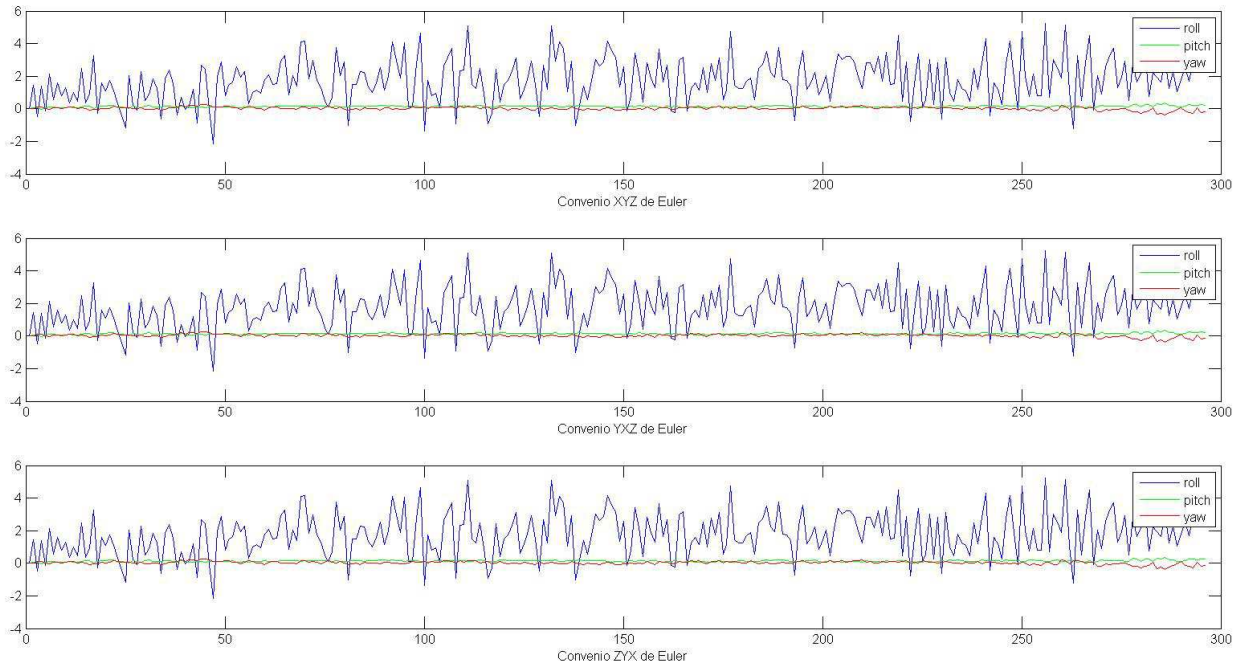


Ilustración 32

Para ver el motivo de porque el "roll" tiene una variación orientativamente de ± 3 grados se recorrerá el proceso de cálculo del ángulo al revés a partir de un ejemplo. El ejemplo concreto será tomado de la iteración 99 de la prueba realizada y que ha sido representada en la ilustración X con 3 gráficas, se escoge este instante ya que es un instante donde el ángulo correspondiente al "roll" tiene un valor bastante lejano a 0 el cual es su valor teórico.
Para el primer convenio XYZ se calcula el ángulo correspondiente al "roll" de la siguiente forma:

$$roll: \alpha \ = \ atan2\left(\frac{-pos(2,3)}{pos(3,3)}\right)$$

Para el segundo convenio YXZ se calcula el ángulo correspondiente al "roll" de la siguiente forma:

$$roll: \beta \ = \ atan2\left(\frac{pos(2,3)}{\sqrt{(pos(2,1)^2 + pos(2,2)^2)}}\right)$$

Para el tercer convenio ZYX se calcula el ángulo correspondiente al "roll" de la siguiente forma:

$$roll: \gamma \ = \ atan2\left(\frac{pos(3,2)}{pos(3,3)}\right)$$

Como se puede observar en los tres casos se usa en el numerador o la posición (2,3) o la posición (3,2) de la matriz de rotación, siendo la matriz de rotación la siguiente.

$$\begin{bmatrix} 1{,}000 & -0.0011 & 0{,}0020 \\ 0{,}0012 & 0{,}9967 & -0{,}0813 \\ -0{,}0019 & 0{,}0813 & 0{,}9967 \end{bmatrix}$$

La matriz teórica de rotación al no haberse movido ni rotado el sensor tendría que ser la siguiente

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Al comparar ambas matrices se observa que las posiciones (2,3) y (3,2) tienen un error mayor que el resto de posiciones.
Ahora se analiza el cálculo de una de esas dos posiciones por ejemplo la posición (2,3) para ver de dónde proviene dicho error. Paralelamente se analiza lo mismo en una posición en la que el valor del ángulo es cercano al teórico para ver donde están las diferencias y detectar la causa del error. Los datos correctos son los que corresponden a la iteración número 96.
Sacando la expresión directamente del código de Matlab se tiene que la matriz de rotación se calcula de la siguiente forma

```
    A(i,:,:)=[w1(i,:);w2(i,:);w3(i,:)]*[v1(i,:);v2(i,:);v3(i,:)]';
```

Teniendo previamente el cálculo de los vectores w1, w2, w3, v1, v2 y v3 de la siguiente forma

```
%v es el observador/nave
    w1(i,:)=Acc(i,:);
    w2(i,:)= cross(w1(i,:),Mag(i,:))/norm(cross(w1(i,:),Mag(i,:)));
    w3(i,:)=cross(w1(i,:),w2(i,:))/norm(cross(w1(i,:),w2(i,:)));
%w es la referencia
    v1(i,:)=Acc(1,:);
    v2(i,:)=cross(v1(1,:),Mag(1,:))/norm(cross(v1(1,:),Mag(1,:)));
    v3(i,:)=cross(v1(1,:),v2(1,:))/norm(cross(v1(1,:),v2(1,:)));
```

Por lo tanto el cálculo para hallar la posición (2,3) es el siguiente:

$$W_{21} \cdot V_{31} + W_{22} \cdot V_{32} + W_{23} \cdot V_{33}$$

Siendo para la iteración 99 los datos los siguientes:
$W_{21} = -0{,}0319;$
$V_{31} = -0{,}9917;$
$W_{22} = 0{,}1388;$
$V_{32} = -0{,}0768;$
$W_{23} = -0{,}9898;$
$V_{33} = 0{,}1034;$

y para la iteración 96 los datos los siguientes:
$W_{21} = -0{,}1107;$
$V_{31} = -0{,}9917;$
$W_{22} = 0{,}1320;$
$V_{32} = -0{,}0768;$
$W_{23} = -0{,}9851;$
$V_{33} = 0{,}1034;$

Se observa que claramente el dato que más varía de una iteración a la otra es el W_{21}, siendo este la causa del error.

El valor de W_{21} es el primer valor del vector W_2 para la iteración 99 por lo que se calcula el vector W_2 para poder apreciar de donde viene el error.

$$W_2 = \frac{[0{,}988 \cdot (-18) - 0{,}1406 \cdot (-98)]\,\vec{\imath} - [-0{,}0639 \cdot (-18) - 0.1046 \cdot 132]\vec{\jmath} + [-0{,}0639 \cdot (-98) - 0{,}9880 \cdot 132]\vec{k}}{\sqrt{[0{,}988 \cdot (-18) - 0{,}1406 \cdot (-98)]^2 + [-0{,}0639 \cdot (-18) - 0.1046 \cdot 132]^2 + [-0{,}0639 \cdot (-98) - 0{,}9880 \cdot 132]^2}}$$

$$W_2 = -0{,}0319\,\vec{\imath} + 0{,}1388\vec{\jmath} - 0{,}9898\vec{k}$$

el dato de interés es el valor de la componente $\vec{\imath}$ es decir el -0,0319 que como se ha podido observar se ha obtenido mediante los siguientes valores.

$$Accelerometro_Y = 0{,}988$$
$$Magnetrometro_Z = -18$$
$$Accelerometro_Z = 0{,}1406$$
$$Magnetrometro_Y = -98$$

Del mismo modo que se ha hecho con la iteración 99 se hace con la 96 y se ve que los valores son los siguientes

$$Accelerometro_Y = 0.988$$
$$Magnetrometro_Z = -27$$
$$Accelerometro_Z = 0.1396$$
$$Magnetrometro_Y = -95$$

Observando los datos se concluye que el error es debido a que los valores correspondientes al $Magnetometro_Z$ varían demasiado y estos influyen directamente en valor de W_{21} y por tanto a la posición (2,3).

Otra forma de comprobar lo concluido anteriormente es mediante la representación del vector aceleración y del vector del campo magnético. Al estar el sensor quieto ambos vectores deberían mantenerse constantes.
En la *ilustración 33* se puede observar como el vector de la aceleración se mantiene constante como la teoría nos indica que debería ser.

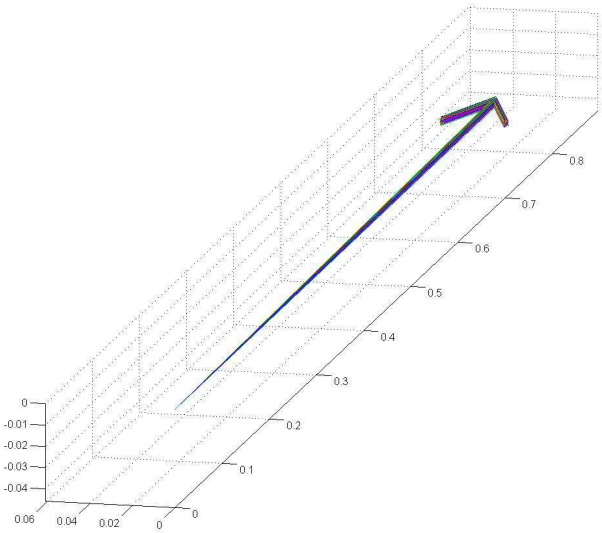


Ilustración 33: Vector aceleración

Por otro lado se puede observar que el vector que representa el campo magnético no es estable y varia a lo largo del tiempo a pesar de estar quieto el sensor, por lo que obviamente se deduce que dicha inestabilidad del campo magnético es la causa inicial para que el algoritmo TRIAD dé resultado inexactos. En las *ilustraciones 34* y *35* se puede observar el vector representante del campo magnético.

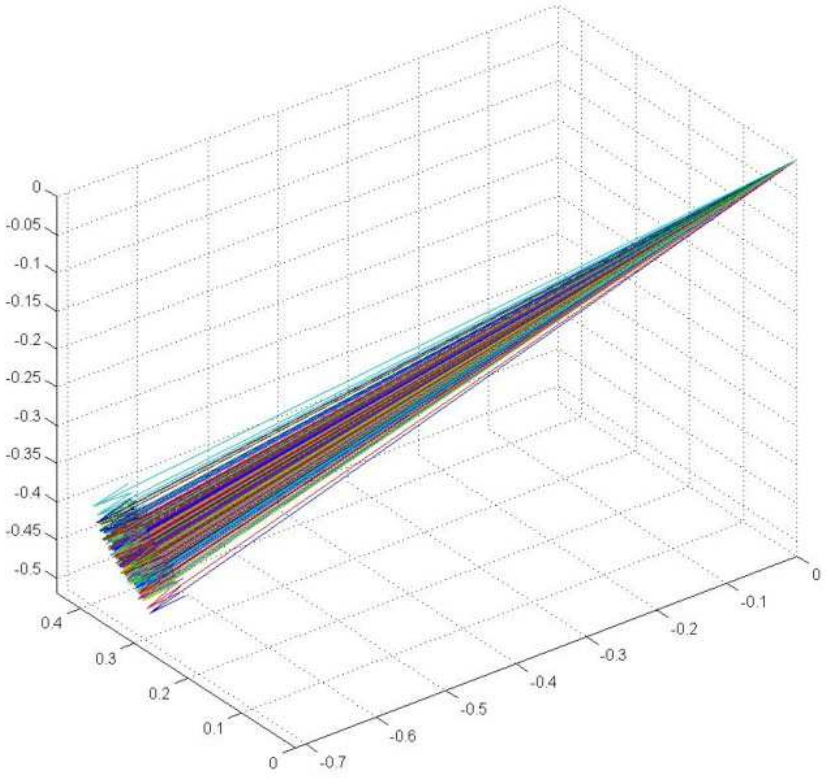


Ilustración 34:Vector del campo magnético

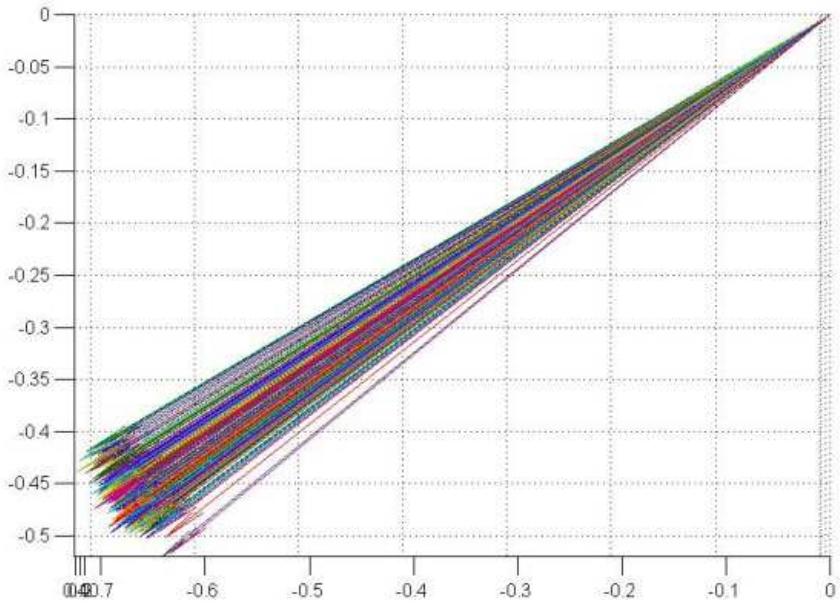


Ilustración 35: Vector del campo magnético

5.3. *Comprobación de los vectores correspondientes a la aceleración y al campo magnético*

En ocasiones se han obtenido resultados inesperados, no congruentes y no se sabía de dónde provenía el error.
Se intuía que estos errores eran debidos a los cambios del campo magnético y por eso se decidió llevar a cabo una comprobación en cuanto a los datos obtenidos con el sensor. Esta comprobación consiste en ver si los datos de los acelerómetros y los de los magnetómetros son adecuados y tienen sentido. Se harán 3 pruebas, un giro sobre el eje X del sensor otro giro sobre el eje Y del sensor y otros sobre el eje Z del sensor.

5.3.1. *Giro respecto al eje X del sensor*

Para efectuar este giro se ha situado el sensor de tal forma que su eje X sea perpendicular a la superficie de la tierra y se ha rotado respecto a este eje un ángulo aproximado de 90°.

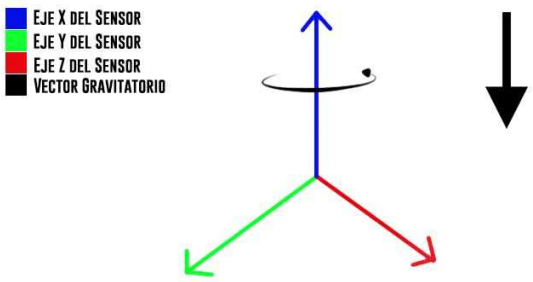


Ilustración 36: Representación de los ejes respecto a la gravedad

Por lo tanto el vector aceleración debería mantenerse constante y el magnético que no se sabe su dirección rotará un ángulo desconocido aunque lo que si se sabe es que las proyecciones del

vector magnético inicial y final sobre el plano perpendicular al vector aceleración deberían formar un ángulo cercano a 90°.
Otro dato de importancia es que el ángulo entre el vector aceleración y el vector del campo magnético debería mantenerse constante, lo cual no será así con exactitud debido a las variaciones del campo magnético pero se debería cumplir con cierta tolerancia.
En la *ilustración 37* se puede ver visualmente que el vector aceleración se mantiene prácticamente constante.

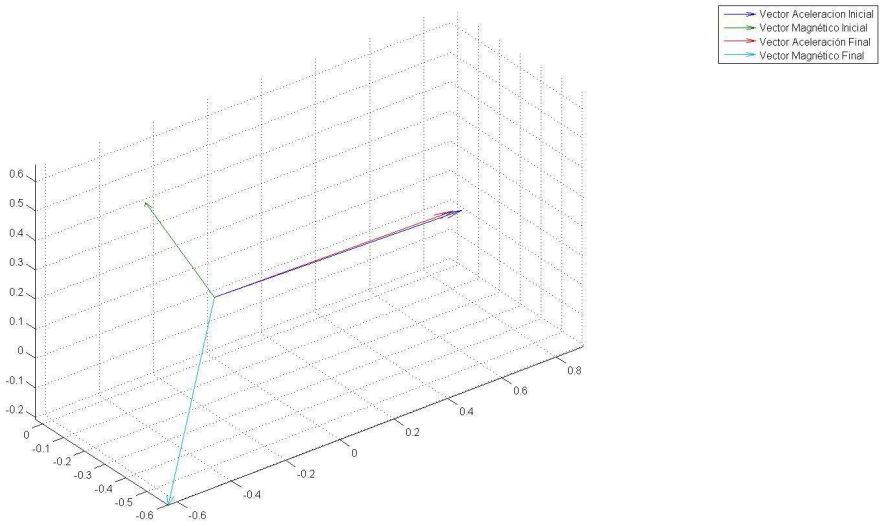


Ilustración 37: Vectores de la aceleración y del campo magnético

Se comprueba además numéricamente el ángulo rotado entre el vector aceleración inicial y el final y este es de aproximadamente 3°.

$$\cos\alpha = \frac{\overrightarrow{Acel}_{inicial} \cdot \overrightarrow{Acel}_{final}}{|Acel_{inicial}| \cdot |Acel_{final}|}$$

$$\alpha = 2{,}7573°$$

Los dos vectores magnéticos describen un ángulo de 60° lo cual se sabe al calcularlo numéricamente, como se ha hecho para el ángulo entre los vectores aceleración, aunque como era de esperar sus proyecciones sobre el plano perpendicular al vector aceleración forman prácticamente 90°, como en la *ilustración 38* se puede ver.

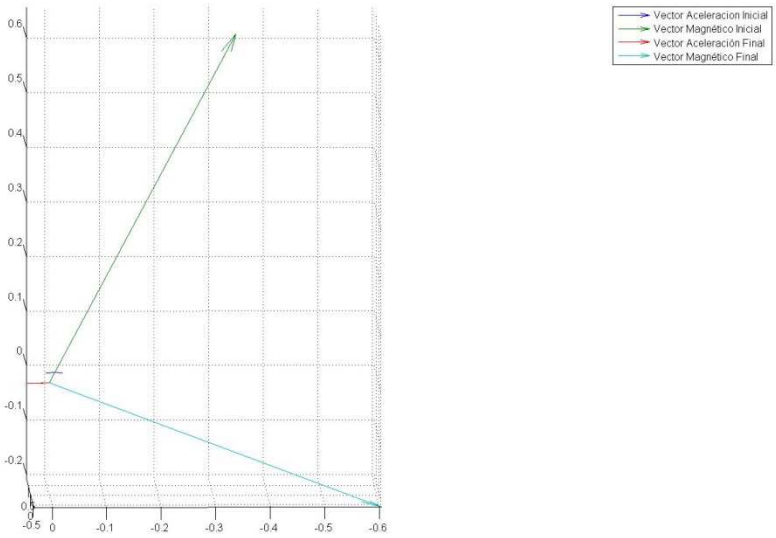


Ilustración 38: Vectores de la aceleración y del campo magnético, visión lateral

En cuanto a al ángulo descrito entre el vector aceleración y el magnético se ve que se mantiene relativamente constante dentro de una franja de 125°-135°. Si el campo magnético fuese tan estable como el gravitatorio y no tuviese fluctuaciones este ángulo sería totalmente constante.

5.3.2. Giro respecto al eje Y

Para efectuar este giro se ha situado el sensor de tal forma que su eje Z sea perpendicular a la superficie de la tierra y se ha rotado respecto al eje Y un ángulo aproximado de 90°.

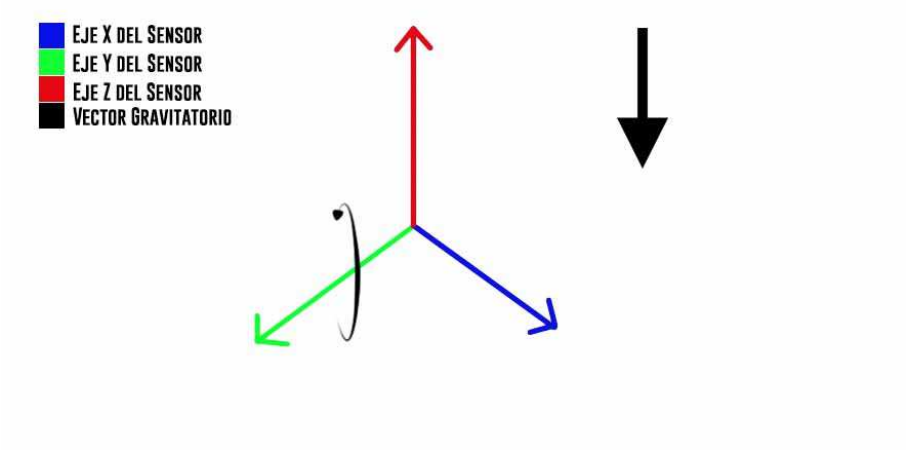


Ilustración 39: Representación de los ejes respecto a la gravedad

Por lo tanto el vector aceleración debería variar generando un ángulo de 90° entre el vector de aceleración inicial y el final. El magnético que no se sabe su dirección rotará un ángulo desconocido.
Otra dato de importancia es que el ángulo entre el vector aceleración y el vector del campo magnético debería mantenerse constante, lo cual no será así con exactitud debido a las variaciones del campo magnético pero se debería cumplir con cierta tolerancia.

En la *ilustración 40* se puede ver visualmente que el vector aceleración inicial y el vector aceleración final describen aproximadamente 90°.

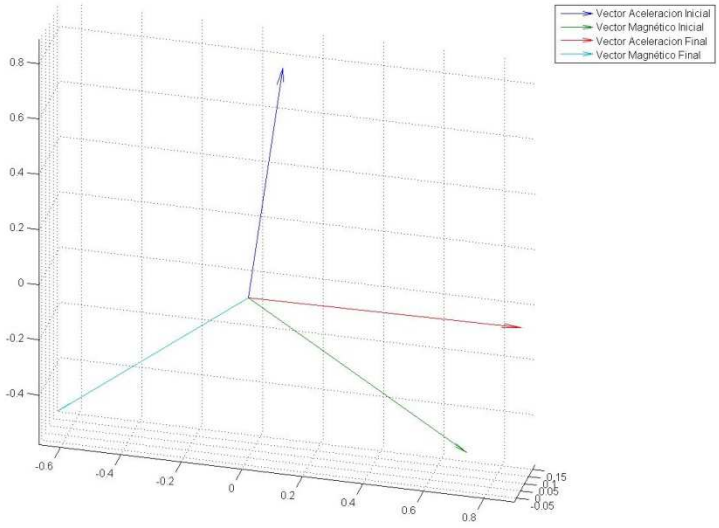


Ilustración 40: Vectores de la aceleración y del campo magnético

Se comprueba además numéricamente el ángulo rotado entre el vector aceleración inicial y el final y este es de 81,1756°.

$$\cos\alpha = \frac{\overrightarrow{Acel}_{inicial} \cdot \overrightarrow{Acel}_{final}}{|Acel_{inicial}| \cdot |Acel_{final}|}$$

$$\alpha = 81{,}1756°$$

Los dos vectores magnéticos describen un ángulo de 97° lo cual se sabe al calcularlo numéricamente. En cuanto a el ángulo descrito entre el vector aceleración y el magnético se ve que se mantiene relativamente constante dentro de una franja de 121°-140°.

5.3.3. Giro respecto al eje Z

Para efectuar este giro se ha situado el sensor de tal forma que su eje Y sea perpendicular a la superficie de la tierra y se ha rotado respecto al eje Z un ángulo aproximado de 90°.

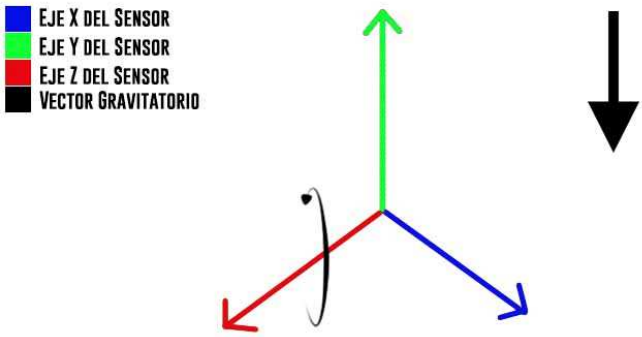


Ilustración 41: Representación de los ejes respecto a la gravedad

Por lo tanto el vector aceleración debería variar generando un ángulo de 90° entre el vector de aceleración inicial y el final. El magnético que no se sabe su dirección rotará un ángulo

desconocido. Otro dato de importancia es que el ángulo entre el vector aceleración y el vector del campo magnético debería mantenerse constante, lo cual no será así con exactitud debido a las variaciones del campo magnético pero se debería cumplir con cierta tolerancia.

En la *ilustración 42* se puede ver visualmente que el vector aceleración inicial y el vector aceleración final describen aproximadamente 90°.

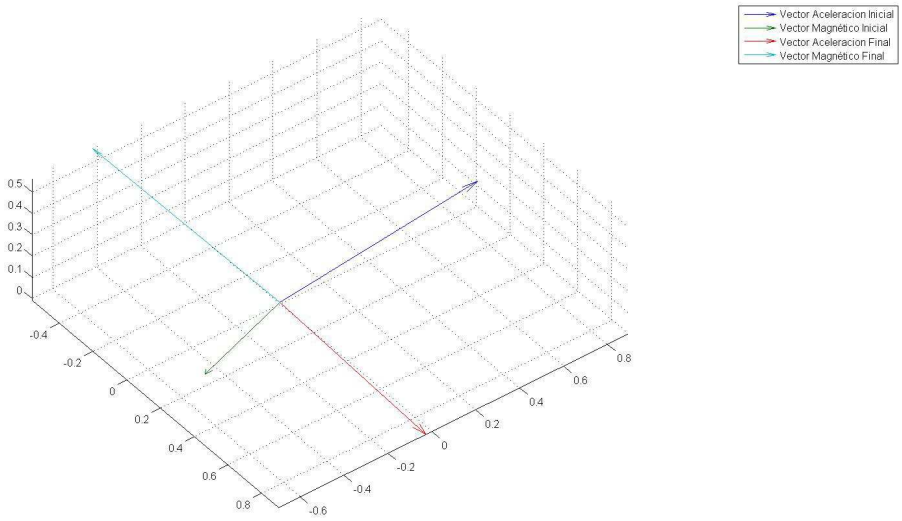


Ilustración 42: Vectores de la aceleración y del campo magnético

Se comprueba además numéricamente el ángulo rotado entre el vector aceleración inicial y el final y este es de 91,1725°.

$$\cos\alpha = \frac{\overrightarrow{Acel}_{inicial} \cdot \overrightarrow{Acel}_{final}}{|Acel_{inicial}| \cdot |Acel_{final}|}$$

$$\alpha = 91{,}1725°$$

Los dos vectores magnéticos describen un ángulo de 74° lo cual se sabe al calcularlo numéricamente. En cuanto a al ángulo descrito entre el vector aceleración y el magnético se ve que se mantiene relativamente constante dentro de una franja de 128°-135°.

Con estas tres comprobaciones se ha verificado que los datos de entrada para el algoritmo TRIAD son razonables a pesar de notar una variación indebida en el campo magnético. Que el campo magnético no sea constante podría suponer un problema pero parece ser que sus deformaciones no son muy grandes y se puede trabajar con él.

5.4. Evaluación de los resultados del TRIAD para giros simples

Con el objetivo de evaluar la precisión y el funcionamiento del algoritmo TRIAD se han realizado los giros ya descritos anteriormente, es decir un giro de 90° sobre el eje X del sensor, otro sobre el Y y otro sobre el Z obteniendo los siguientes resultados.

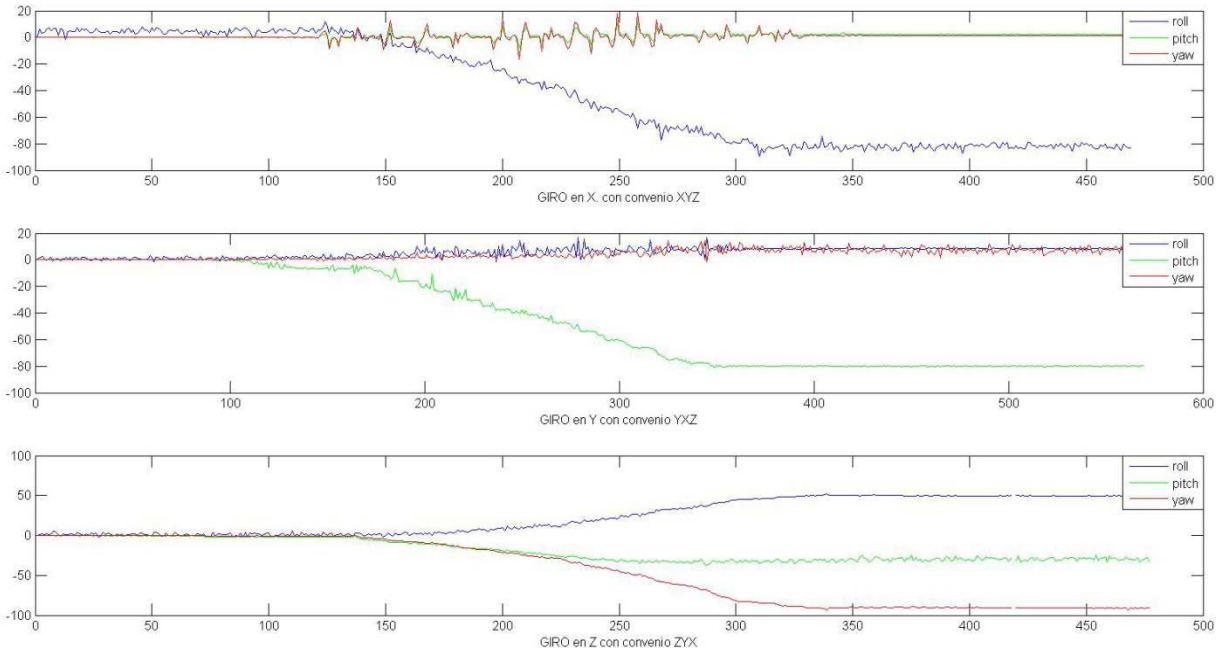


Ilustración 43: Representaciones de un giro de 90 grados en cada uno de los ejes

A primera vista parece ser que el giro sobre el eje X del sensor, el primero de la gráfica de la *ilustración 43*, se ha representado con bastante precisión, lo mismo se podría decir del giro respecto al eje Y del sensor, segundo en la gráfica.
Sin embargo el giro respecto al eje Z del sensor no parece estar bien representado.
En realidad los tres giros se han representado perfectamente, lo que sucede es que hay que tener en cuenta lo que estos giros para el algoritmo TRIAD representan y no confundir los ejes del TRIAD con los ejes del sensor.

El algoritmo TRIAD como ya se ha explicado al inicio del capítulo lo que hace es generar un triedro a partir del vector de la aceleración y del vector del campo magnético. Los ángulos correspondientes al "roll" al "pitch" y al "yaw" que se ven representados en la *ilustración 43*son los ángulos respecto a los ejes X, Y y Z del triedro del TRIAD y no del sensor.
En el primer ejemplo, rotando 90° sobre el eje X del sensor la gráfica muestra un giro correcto de 90° para el "roll" y de 0° para el "pitch" y para el "yaw" , esto es así porque para generar el vector X del triedro se coge el vector aceleración que coincide con el eje X del sensor en este caso porque así se quiso colocando el sensor con su eje X perpendicular al suelo.
En la siguiente ilustración se pueden observar ambos triedros, tanto el inicial como el final, y como realmente quedan rotados 90° respecto a su eje X.

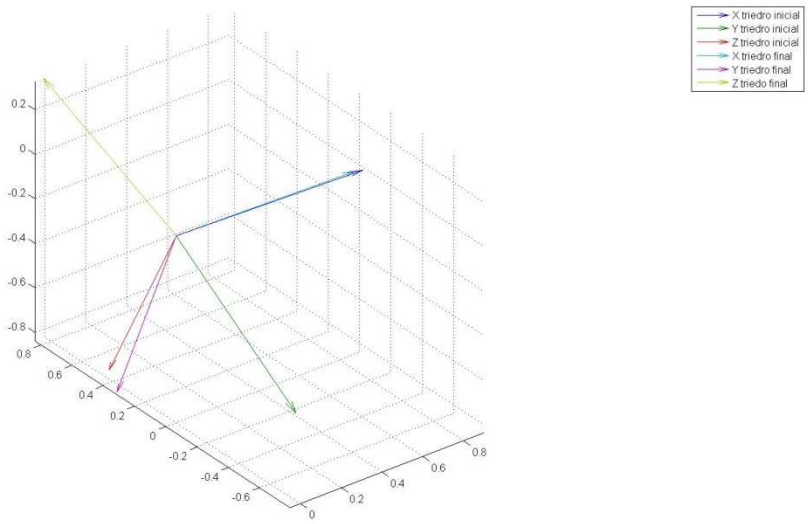


Ilustración 44: Representación del triedro inicial y del triedro final de TRIAD

Para generar el segundo vector del triedro, es decir el vector Y del triedro, se utiliza la proyección del vector magnético en el plano perpendicular al vector X del triedro, es decir del vector aceleración. En la gráfica se puede observar que se ha rotado 90° respecto al eje Y del triedro, esto ha sido así porque casualmente ha coincidido más o menos el eje Y del sensor con el eje Y del triedro.
En la siguiente ilustración se puede observar gráficamente que se cumple lo descrito anteriormente.

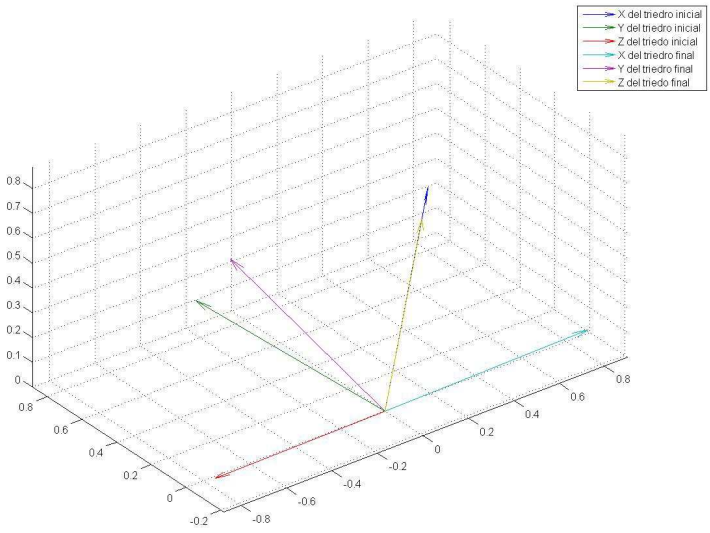


Ilustración 45: Representación del triedro inicial y del triedro final de TRIAD

Para el último giro el realizado respecto al eje Z del sensor se obtiene la gráfica que se puede observar en la ilustración 43, concretamente la de más abajo. Este resultado de entrada incoherente deja de serlo y se comprende al pensar que el eje Z del triedro no coincide con el eje Z del sensor y por lo tanto resulta un giro combinado de los tres ejes del triedro a pesar de ser un giro simple para el sensor ya que se giró respecto a su eje Z.
El eje Z del triedro se obtiene de forma muy simple una vez se tienen ya los otros dos, simplemente tiene que se perpendicular al plano formado por los otros dos vectores del triedro.

Se puede ver en la *ilustración 46* como se rota el triedro respecto a un eje desconocido por lo que la gráfica anterior correspondiente a la *ilustración 43* ya cobra total sentido.

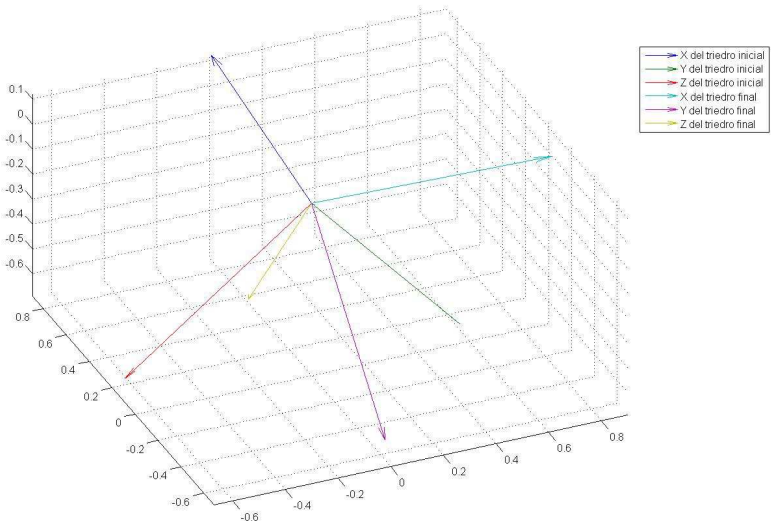


Ilustración 46: Representación del triedro inicial y del triedro final de TRIAD

6. DISEÑO DEL PROTOTIPO

El prototipo debe ser capaz de capturar el movimiento realizado durante el entrenamiento para procesarlo y determinar la velocidad y aceleración del movimiento.

6.1. Caja negra

Para determinar todas las características necesarias se parte de la idea general descrita anteriormente y se genera el gráfico conocido como "la caja negra". En este gráfico no se da información detallada alguna, ni se determinar como se va a llevar a cabo el proceso pero si se indica con que datos se parte, con que recursos se cuenta y que es lo que se debe obtener.

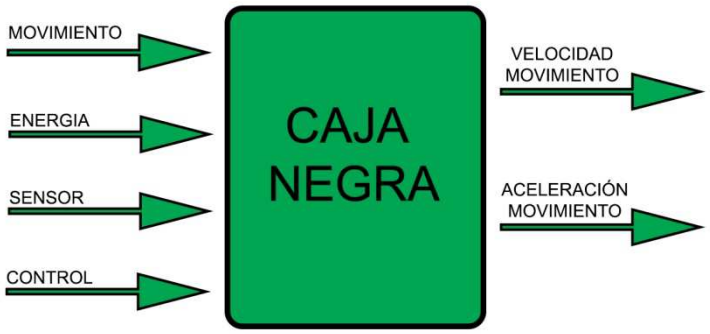


Ilustración 47: Caja negra

6.2. Caja blanca

Una vez realizado el gráfico anterior se procede a realizar el siguiente que se denomina "caja blanca" y consiste en abrir la caja negra y ver el funcionamiento interno de forma esquemática. En este gráfico se puede observar en que momento los recursos de lado izquierdo de la caja negra se implementan en el prototipo y como se combinan entre ellos con el resultado final de obtener los resultados del lado derecho.

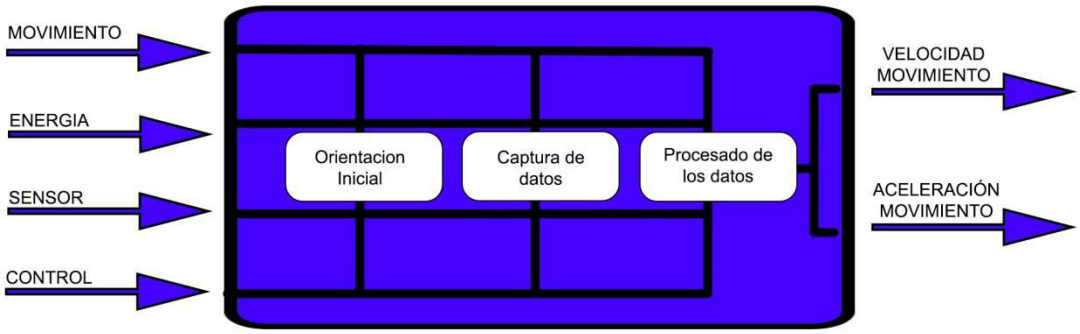


Ilustración 48: Caja blanca

6.3.Exigencias y deseos

Con el fin de determinar de forma más precisa diversas características del prototipo se escriben en una tabla las características que el prototipo debería poseer o sería aconsejable que poseyese determinando de forma definitiva cuales son exigencias y cuales son deseos.

FECHA	Deseo/Exigencia	Descripción
11/09/1990	E	El peso ha de ser lo más reducido posible
11/09/1990	E	La robustez ha de ser máxima
11/09/1990	E	Debe poderse acoplar a las barras olímpicas
11/09/1990	D	Uso de un sólo sensor
11/09/1990	D	Conexión inalámbrica
11/09/1990	D	Procesado de datos a tiempo real
11/09/1990	E	Rápido de poner y quitar de la barra
11/09/1990	E	Bajo consumo de energía

6.4.Diseño electrónico

Para capturar y enviar los datos de los sensores se ha utilizado una placa de Arduino, concretamente el Arduino Mega2560. Se ha utilizado esta placa porque dispone de un controlador ATMega2560 que tiene una velocidad de 16Mhz la cual es más que suficiente para la tarea que debe realizar, además permite la comunicación con un PC a través de su conexión USB lo cual es una gran ventaja para poder empezar a trabajar con ella sin tener todos los componentes montados, es decir antes de que se le conectase el módulo bluetooth HC06.
Otra gran ventaja es la enorme comunidad online que hay para Arduino, con miles de ejemplos y librerías de gratuito acceso, sin olvidar por supuesto la simpleza de su programación.
Aunque como ya se ha dicho inicialmente antes de llegar a tener todos los componentes se pudo alimentar y conectar el Arduino al PC a través de su conexión USB. Esto luego se modificó para conectarse a cualquier dispositivo que dispusiese de bluetooth a través del módulo HC06 y se alimentó con una batería de 9V.
En la *ilustración 49* se puede observar el esquema con las conexiones y todos los elementos ya montados.

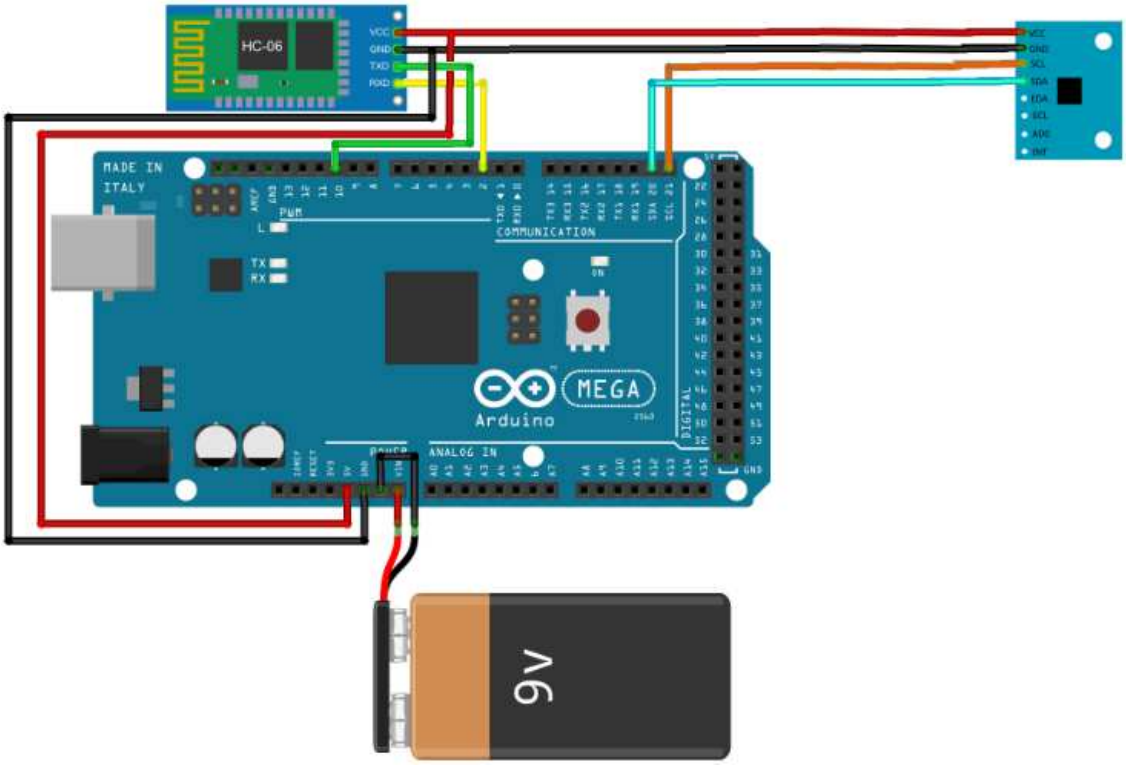


Ilustración 49: Esquema electrónico

Un punto de especial interés en las conexiones es que se podría conectar el módulo bluetooth HC06 a los pines 0 (RX) y 1(TX) de Arduino simplemente cruzándolos, pero no se hizo así ya que estos pines se utilizan en la comunicación serie de Arduino con el PC a través del cable USB y por lo tanto si se emplean para el Bluetooth se pierde la conexión con el PC.
La solución es muy sencilla, se importa una librería para habilitar la comunicación serie con otros pines dejando así libres los pines 0 y 1 para poder usarlos para conectarnos con el PC mediante cable si así se desea.
Cabe destacar que no todos los pines del Arduino se pueden habilitar para tal uso, en concreto para el Mega2560 un par de los posibles son los pines 2 y 10 que son los que se han elegido en este caso, como en la *ilustración 49* se puede observar.
En cuanto a la alimentación, se ha comentado que el Arduino se alimentaba finalmente mediante una pila de 9V. Para poder conectar la pila de 9V al Arduino se emplea un adaptador de 9V a Jack. Este cable se puede ver en la ilustración 50.

Ilustración 50: Adaptador para pila de 9V

6.5.Diseño mecánico

Para obtener un resultado en la práctica óptimo hay que pensar en el tipo de usuario que va a utilizar el producto. El prototipo será muy útil para personas que tengan un nivel de entrenamiento bastante alto incluyendo en este nivel a personas que entrenan a nivel amateur sin competir pero buscan optimizar sus resultados, por lo tanto se considera que estas personas entrenan con material olímpico el cual tiene unas medidas únicas y estandarizadas. Teniendo en cuenta esto el soporte para alojar tanto la placa, como la pila como el sensor y el módulo bluetooth tiene que ser capaz de acoplarse a una barra olímpica de competición.
La idea básica desde un inicio fue la de acoplar un cilindro a la barra, estando este cilindro compuesto por dos partes, una en la cual se alojan todos los dispositivos y la otra es simplemente medio cilindro que sirve para fijar el conjunto. En la siguiente ilustración se puede ver un esbozo inicial a mano alzada.

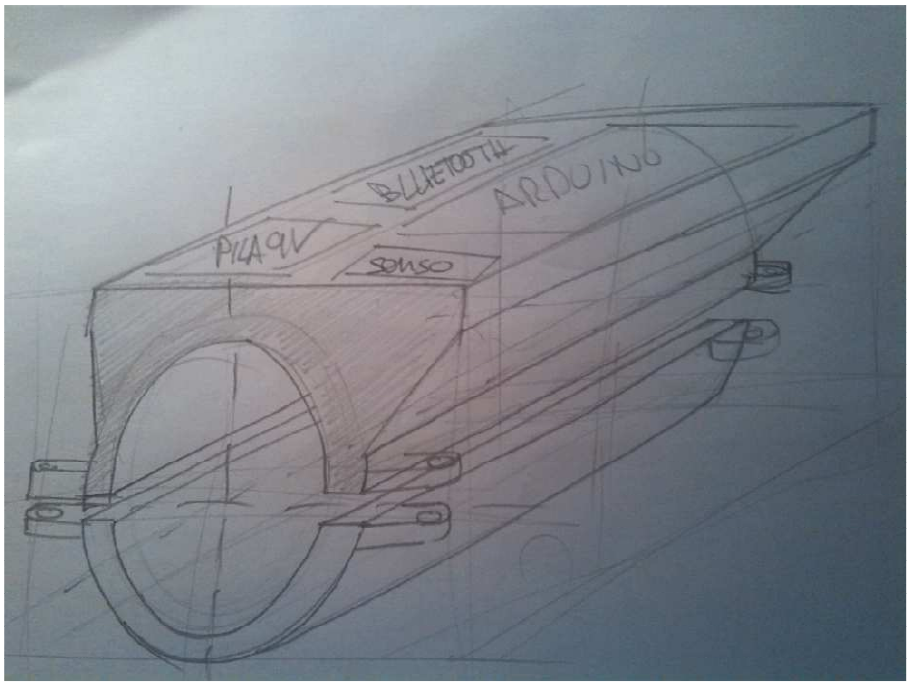


Ilustración 51: Esbozo inicial del prototipo

Para desarrollar el diseño mecánico se hizo una construcción en CAD con el programa SolidWorks, generando las dos piezas en 3D, haciendo un ensamblaje con el programa, montando todos los elementos y optimizando así con facilidad el soporte de cara a su fácil montaje y economizando su construcción.
En la *ilustración 52* se puede observar la pieza inferior que es la más sencilla y básicamente es medio cilindro con un vaciado para ahorrar material y los 4 agujeros para pasar los 4 tornillos a través de los cuales esta pieza se unirá a la otra pieza, a la pieza superior.

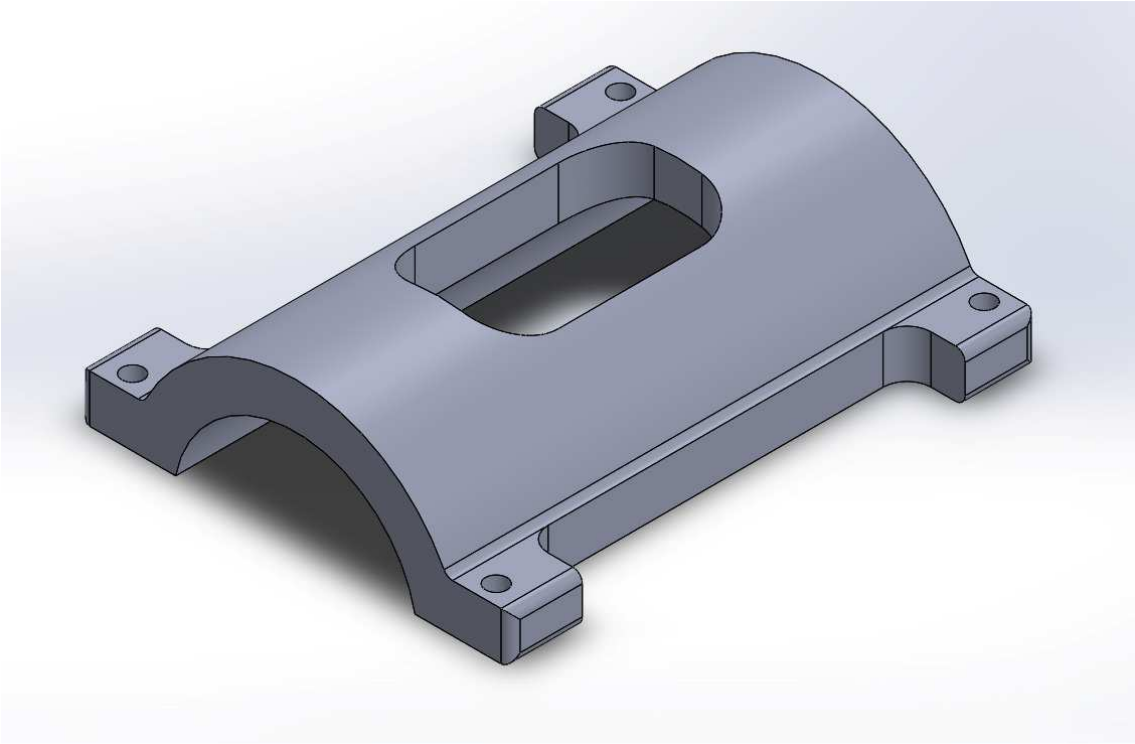


Ilustración 52: Pieza inferior

En la *ilustración 53* se puede observar la pieza superior que es la más compleja y donde van instalados el resto de componentes.

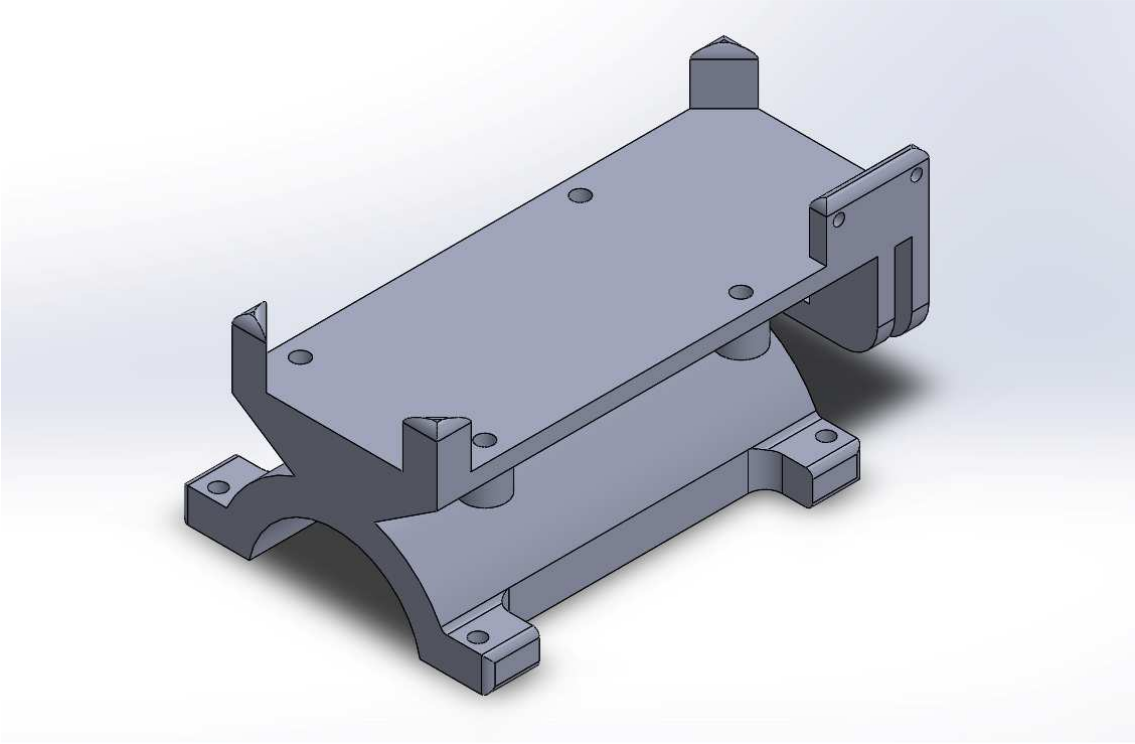


Ilustración 53: Pieza superior

Como se puede observar en la *ilustración 55* en su parte más amplia va atornillada la placa de Arduino, representada en azul oscuro. En el extremo va situado el módulo bluetooth, que está representado con un color azul claro y la batería de 9V que está representada en color rojo. Finalmente por último el sensor está situado en el lateral de la pieza y en la ilustración se puede observar en un color granate oscuro.

Cabe destacar que se ha diseñado la pieza con sus esquina sobresaliendo para aumentar la vida de la placa de Arduino ya que aunque se cayese el prototipo al suelo la placa jamás se golpearía contra el suelo. Esto se puede ver claramente en la *ilustración 54* donde se ve la vista lateral del prototipo con la placa Arduino ya montada.

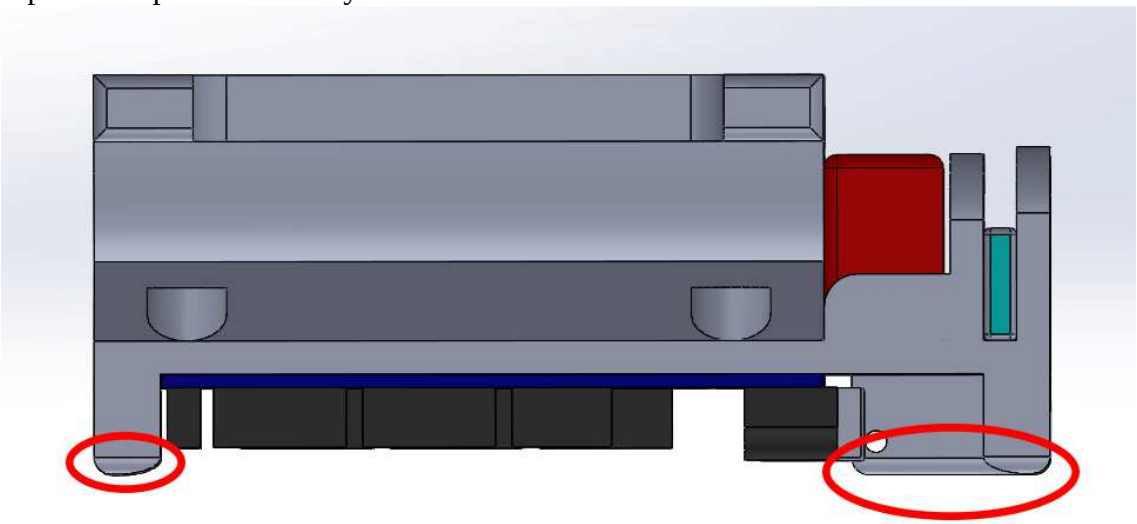


Ilustración 54: Detalle de protección de la placa Arduino

En la *ilustración 55* se puede observar el montaje completo de ambas partes con todos los componentes.

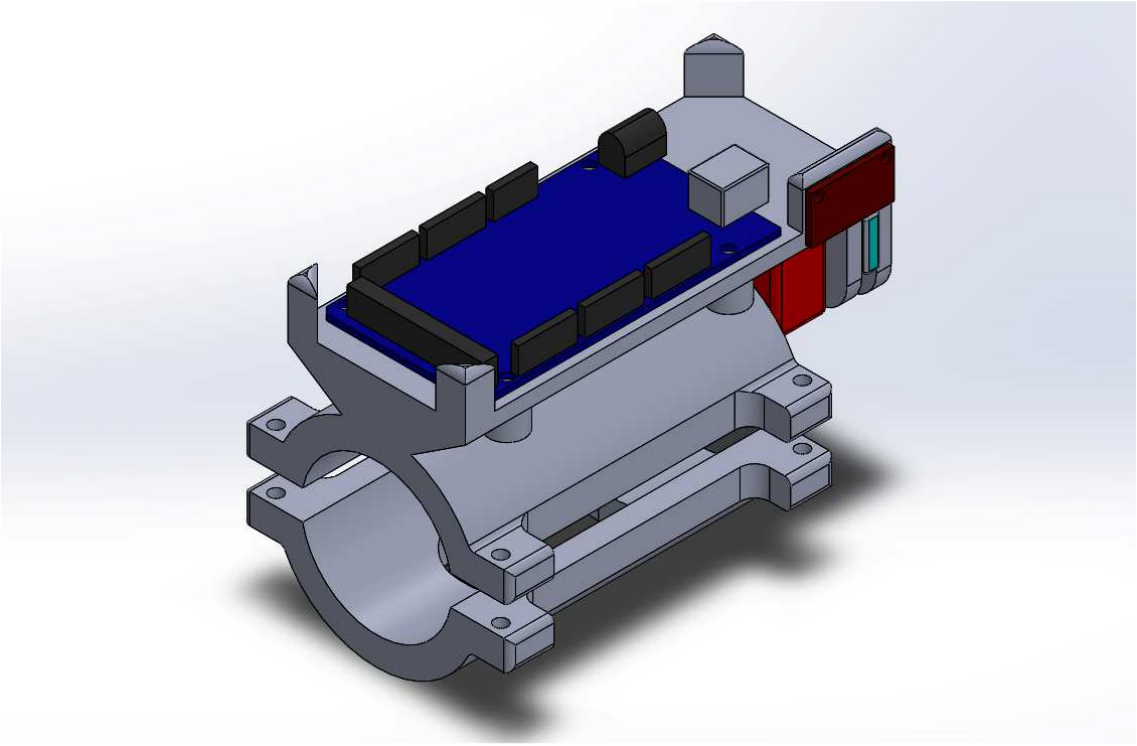


Ilustración 55: Prototipo final

Al hacer el diseño 3D con el programa SolidWorks se puede directamente guardar las piezas diseñadas en formato .STL y por tanto imprimirlas con una impresora 3D mediante fabricación aditiva.

6.6. *Código*

El código tiene una estructura muy simple, tras la definición de los registros y la inicialización de la comunicación serie se leen los datos del sensor enviando la información por el puerto serie.
Se incluye la librería <*Wire.h*> para llevar a cabo la lectura y escritura de los datos del sensor MPU9150 y la librería <*Softwareserial.h*> para poder llevar a cabo una comunicación serial a través de otros pines diversos del Arduino al 0 y al 1.
El código se puede ver en el anexo correspondiente.

6.7. *Procesado de los datos*

El procesado se los datos se ha decidido realizar offline para poder obtener datos a una frecuencia mucho mayor durante el movimiento.
Por tanto el calibrado y la ejecución del algoritmo TRIAD para obtener la matriz de rotación y con ello los ángulos se lleva a cabo a posteriori con Matlab. Teniendo esto en cuenta el flujo de trabajo del prototipo desde la obtención de los datos hasta el cálculo de la velocidad de movimiento es el que se muestra en la *ilustración 56*.

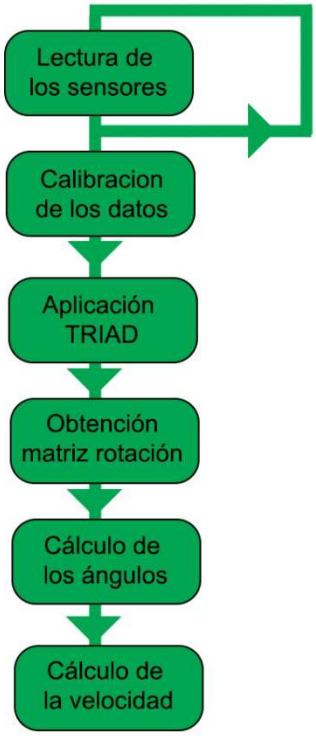


Ilustración 56:Diagrama de flujo

7. RESULTADOS EXPERIMENTALES EN EL GIMNASIO

Para evaluar el funcionamiento del prototipo se va a utilizar este para registrar el movimiento durante la ejecución del ejercicio de press de banca.
El press de banca también conocido como press de pecho es un ejercicio con pesas que trabaja principalmente la zona superior del cuerpo.
Este ejercicio físico es considerado uno de los tres básicos siendo estos básicos el peso muerto, las sentadillas y el press de banca.
Los culturistas lo ejecutan para entrenar principalmente los pectorales, aunque también él tríceps y la cabeza anterior del deltoides. El atleta se tumba sobre su espalda en un banco, levantando y bajando la barra directamente por encima del pecho.
En la ilustración 57 se puede ver con claridad el movimiento.

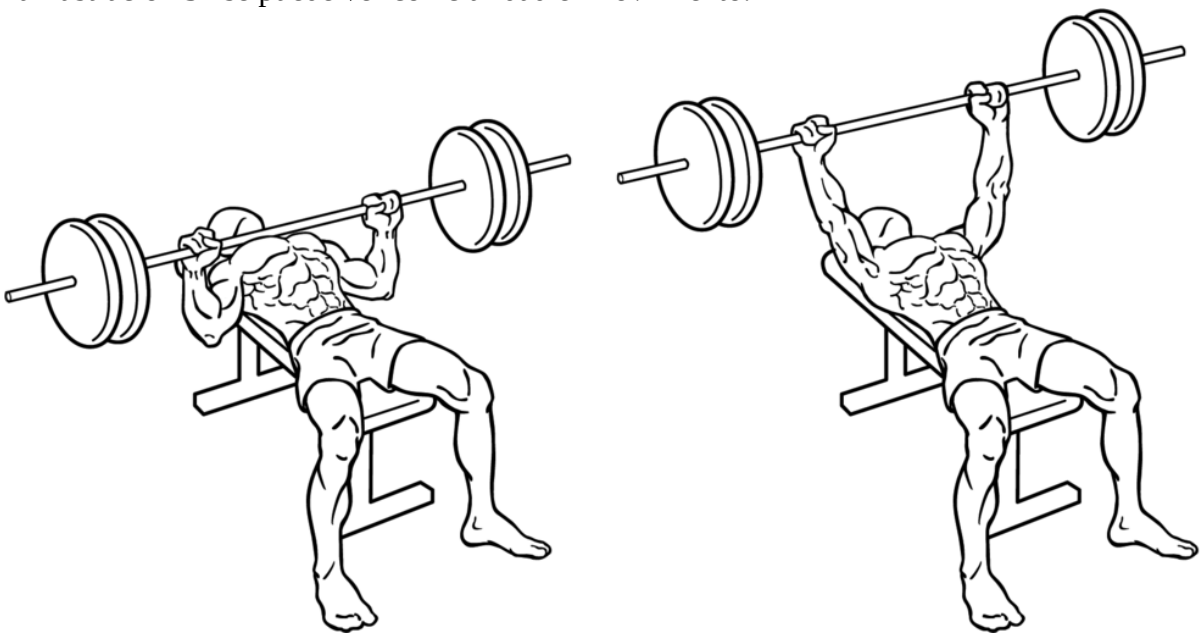


Ilustración 57: Esquema del ejercicio Press de banca

7.1. Obtención de los datos relativos al movimiento de la barra

Para registrar el movimiento de la barra se fija el sensor a la barra mediante varias tiras de cinta adhesiva de tal forma que queda el sensor totalmente solidario a la barra, además se coloca de tal forma que el eje Y del sensor coincide en la medida de lo posible con la barra.
El sensor se coloca de esta forma porque teóricamente la barra siempre debería mantenerse paralela al suelo tanto en su movimiento descendente como en el ascendente y por tanto el acelerómetro en el eje de la barra debería registrar una aceleración nula, en la práctica esto se cumple aunque no a la perfección como era de esperar.

En la ilustración 58 se puede observar los datos obtenidos por el sensor ya calibrados y pasados a Gs. Con esta gráfica se puede observar que efectivamente el valor de la aceleración en el eje Y del sensor es prácticamente cero y además se mantiene constante durante todo el movimiento.
La gravedad y la aceleración debida al movimiento ascendente de la barra se ven reflejadas en el eje X y en el Z del sensor como en la gráfica se puede ver.

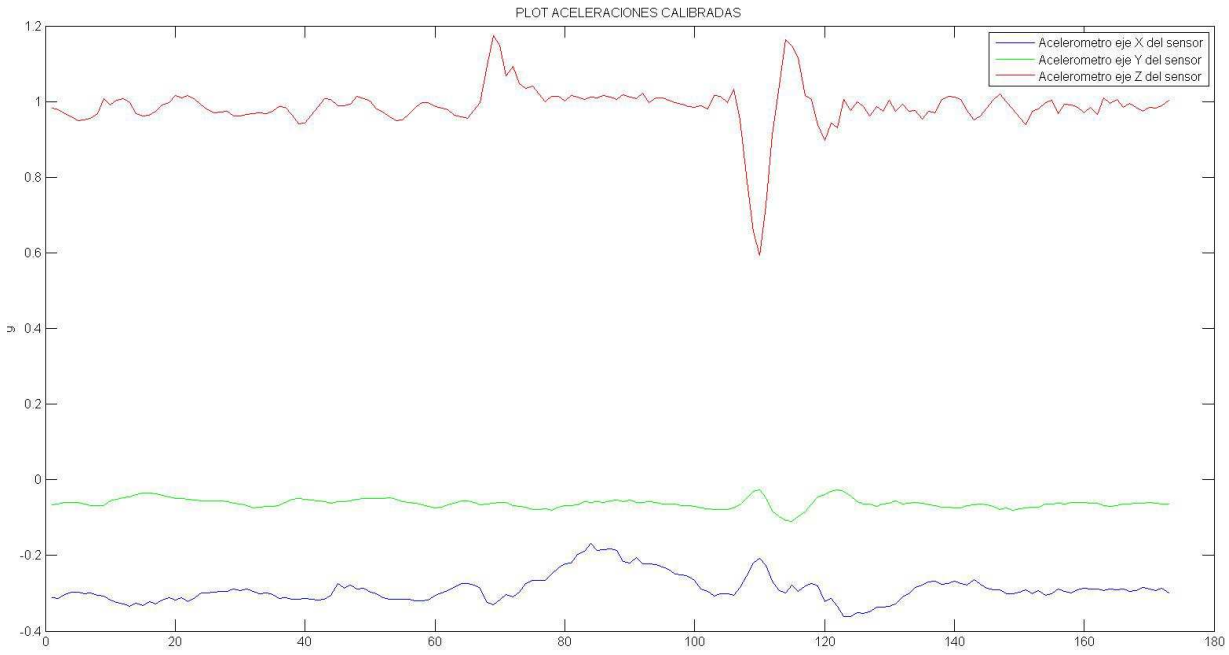


Ilustración 58: Datos de los acelerómetros en G´s

Para obtener la aceleración debida a la gravedad y a la aceleración del movimiento ascendente se hace la media cuadrática de los valores registrados para el eje Z y para el eje X del sensor, obteniendo la gráfica que se puede ver en la ilustración 59.

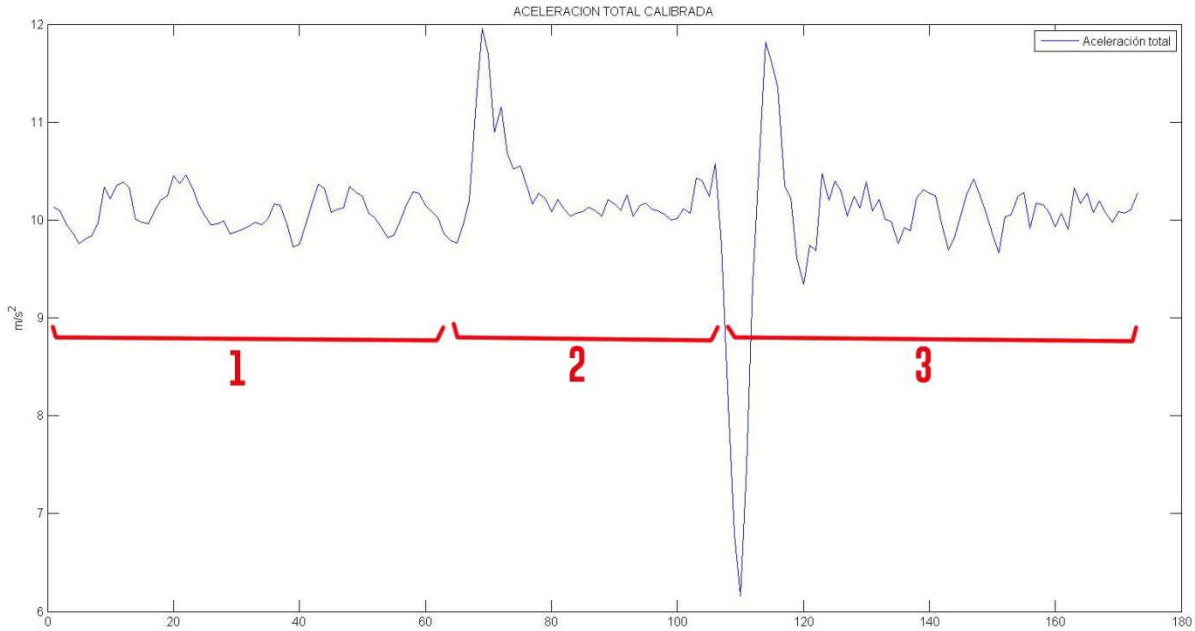


Ilustración 59: Representación de la aceleración en m/s^2

El tramo correspondiente al número uno corresponde al tiempo en el que la barra esta abajo, mantenida por las manos y apoyada en el pecho, como se puede observar se mantiene una aceleración relativamente constante y próxima a la de la gravedad.

El siguiente tramo, el correspondiente al número dos, es el importante, el cual corresponde a la subida de la barra. Se observa como era de esperar una aceleración grande al inicio del movimiento, seguida de una aceleración pequeña a medio recorrido y volviendo a acelerar más ligeramente al final del recorrido. De este tramo se obtiene la información necesaria para poder calcular la velocidad de ejecución del levantamiento y con ello el peso máximo que se podría levantar a una sola repetición.
El tercer tramo simplemente muestra como la barra llega a tener una aceleración en el otro sentido ya que es cuando la barra se detiene bruscamente arriba y se mantiene arriba lo más estable posible antes de posarla en su soporte.

7.2.Cálculo de la rotación de la barra

Con el fin de poder saber la orientación final de la barra respecto a la inicial se ejecuta el algoritmo de TRIAD y se observa que la barra prácticamente no ha rotado a lo largo de ningún eje, ni siquiera ha rotado prácticamente sobre si misma como era de esperar.
En la ilustración 60 se puede observar visualmente que la rotación de la barra es mínima.

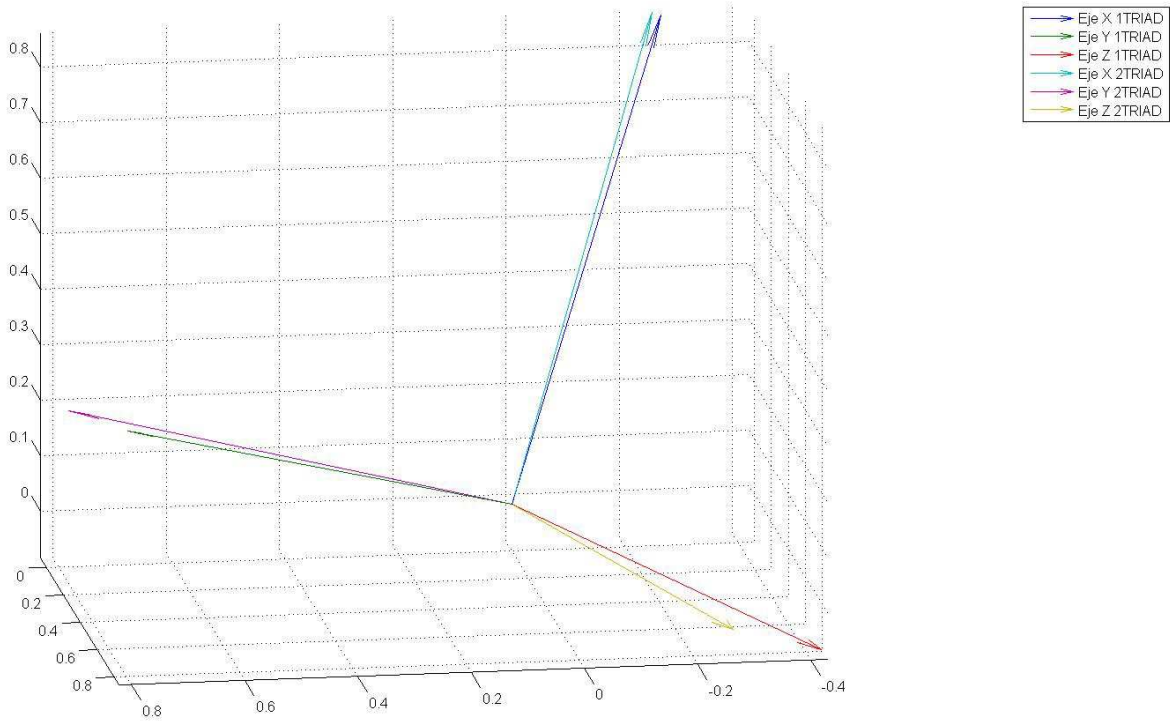


Ilustración 60: Representación del triedro inicial y final de TRIAD

Para poder además confirmar que la rotación es mínima se calculan los ángulos rotados en cada eje de los tres del TRIAD, recordando siempre que estos no tienen porqué coincidir con los del sensor.
En la ilustración 61 se pueden observar los ángulos en cada uno de los ejes. Podría parecer que si que se rota bastante dado que en la etapa del movimiento ascendente los ángulos aumentan pero esto no se debe tener en cuenta ya que el algoritmo TRIAD sirve para calcular la posición evaluando la aceleración y el campo magnético con el objeto de estudio en reposo.

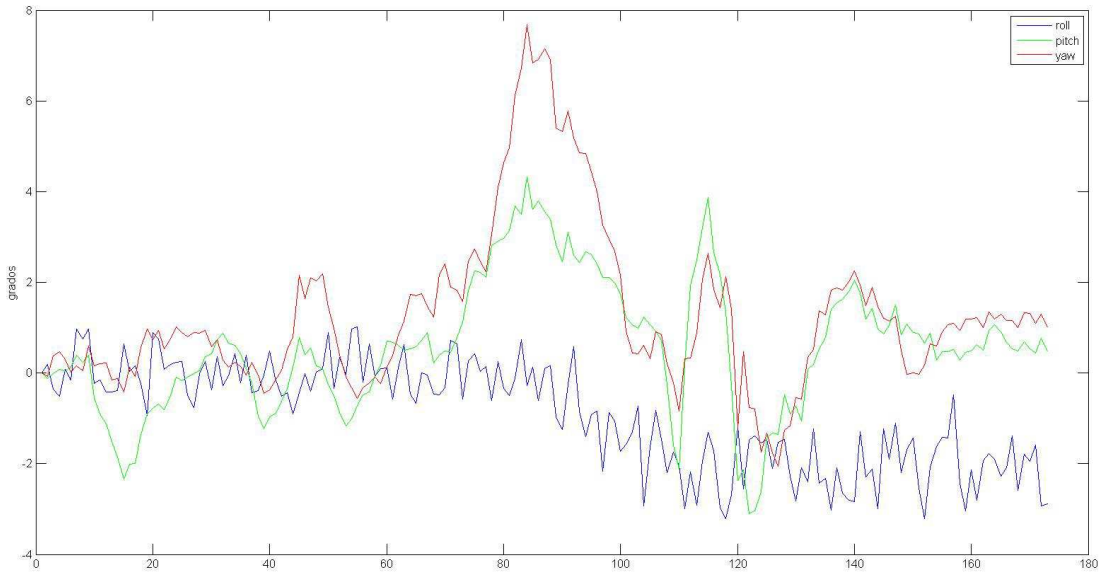


Ilustración 61: Representación de los ángulos girados en cada uno de los ejes

La ilustración 62 es un zoom de la parte final de la ilustración 61, de la parte cuando la barra está arriba y se mantiene lo más estable posible en esa posición. Hay que recordar que aunque se mantenga lo más estable posible dejarla quieta por completo es imposible ya que la barra está siendo sujetada por las manos del atleta y por tanto algún pequeño movimiento tiene. Al no estar la barra absolutamente parada los acelerómetros registran aceleraciones externas debidas al pequeño movimiento y no sólo la de la gravedad por lo que los cálculos de los ángulos están algo sobredimensionados, pero a pesar de ello se ve que rondan de media los 2° o 3° a lo sumo por lo que se concluye que se puede despreciar para el cálculo de la velocidad la rotación de la barra ya que es insignificante.

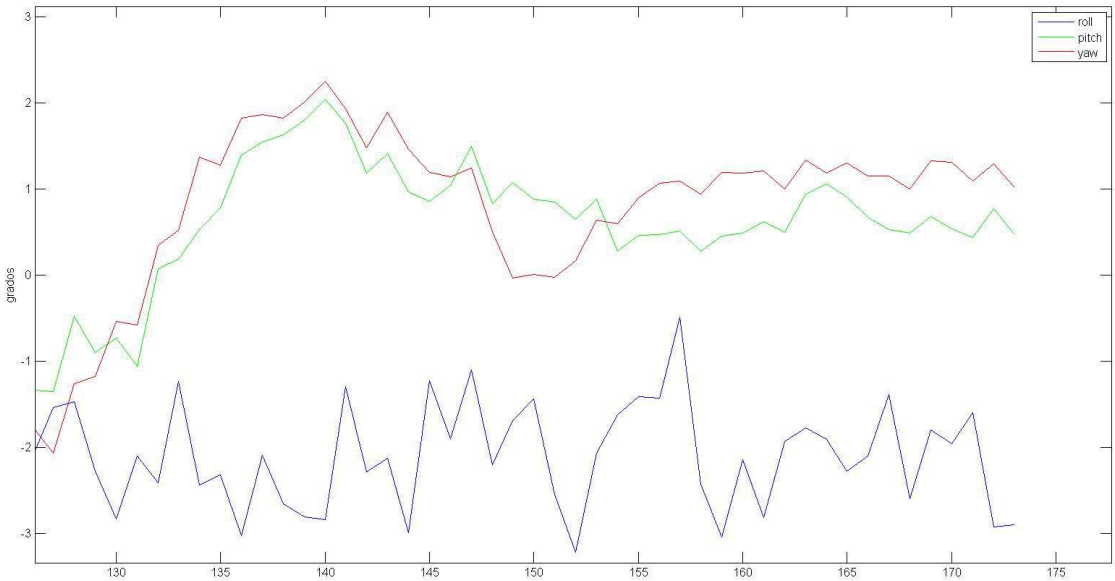


Ilustración 62: Representación de los ángulos girados en cada uno de los ejes, tramo final

Está conclusión es algo que no era de esperar, la teoría dice que la barra ha de subir siempre paralela al suelo y en dirección perpendicular al suelo pero aún así se esperaba una rotación de la barra a lo largo de su eje longitudinal significativa y en cambio ha sido mínima.

7.3.Cálculo de la velocidad

Para obtener la velocidad de la barra en su movimiento ascendente se trabaja con los datos de las aceleraciones a partir del momento justo en el que se inicia el movimiento ascendente hasta que se finaliza. En la ilustración 63 se puede ver el tramo marcado en rojo estando la aceleración ya en m/s^2 y habiéndole restado ya la debida a la gravedad aislando así la debida simplemente al movimiento de la barra.

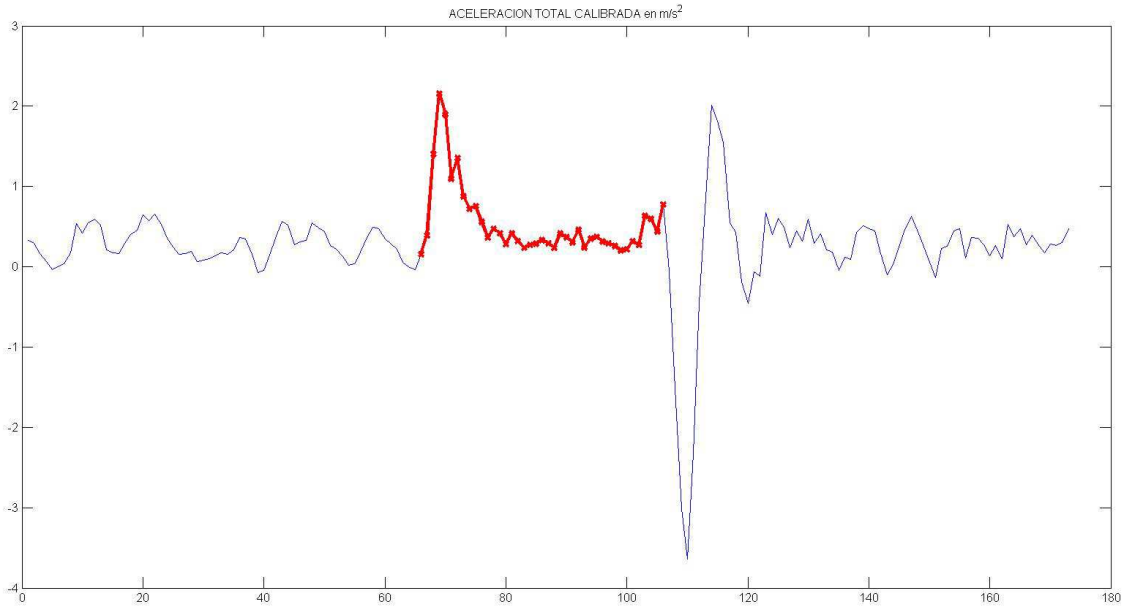


Ilustración 63: Representación de la aceleración debida al movimiento, tramo de subida de la barra marcado en rojo

A partir de las aceleraciones en cada momento se puede calcular la velocidad en cada punto del movimiento y haciendo la media se obtiene la velocidad media del levantamiento en cuestión. El sensor está trabajando registrando 25 muestras por segundo por lo que ya se tienen todos los datos necesarios para calcular la velocidad.

La velocidad media de este levantamiento en concreto es de 0,2769 m/s.

7.4.Explicación y cálculo del peso máximo a una sola repetición

Antes de pasar a calcular el 1RM se va a explicar en qué consiste exactamente este concepto. 1RM se refiere a una repetición con el peso máximo que es posible que cierto individuo mueva. De tal forma existe también un 5RM, 8 RM y 12 RM siendo estos los pesos máximos que un individuo puede mover a 5, 8 y 12 repeticiones respectivamente.
Es de gran interés saber el 1RM porque a partir de él se pueden calcular los pesos óptimos a utilizar para ganar fuerza, o hipertrofiar los músculos, según se desee.

En el levantamiento en concreto correspondiente a la gráfica de la ilustración 63 se utilizó un peso total de 76,5kg y se ejecutó el levantamiento con una velocidad media de 0,2769 m/s por lo que se calcula el 1RM con la siguiente fórmula.

$$\%1\text{RM} = 11{,}4196v^2 - 81{,}904v + 114{,}03$$

$$1RM = \frac{PESO \cdot 100}{\%1RM}$$

Estas fórmulas son las que importantes entrenadores y especialistas en la materia utilizan, sirva de ejemplo que Sánchez-Medina, Pérez y González-Badillo las emplean siendo un referente en este campo.

Simplemente sustituyendo en las fórmulas las variables por sus valores numéricos para el caso concreto que se está tratando se obtiene que la 1RM es de 82,9508kg.

Este valor es más que razonable teniendo en cuenta que el atleta, yo mismo en este caso, ha llegado a tener un 1RM teórico en el pasado de 95kg y el día de ejecución de las pruebas con el sensor estaba sometido a un proceso de más de 3 semanas de pérdida de peso y consumo de kcal inferior al básico para mantener mi peso.

7.5. Confirmación con una segunda prueba experimental

A pesar de haber obtenido unos resultados que parecían ser correctos en el primer ensayo, se consideró necesario hacer una segunda tanda de ensayos para confirmar la estabilidad del prototipo en cuanto a resultados.
En esta tanda de ensayos definitiva se va a ejecutar un levantamiento máximo real para comprobar que efectivamente lo calculado corresponde con la realidad.

Se repite el ensayo anterior, es decir se vuelve a ejecutar el mismo levantamiento con el mismo peso y se obtiene la siguiente gráfica.

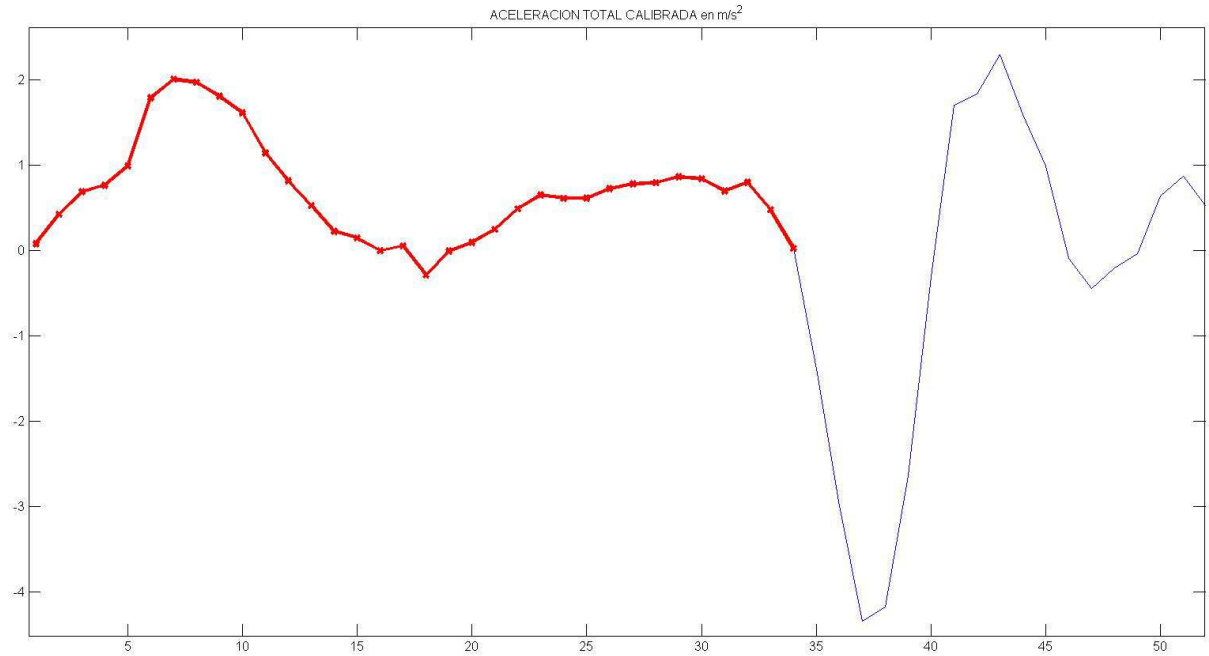


Ilustración 64: Representación de la aceleración debida al movimiento, tramo de subida de la barra marcado en rojo

Se calcula la velocidad media de ejecución del movimiento y se obtiene una velocidad teórica de 0.2756 m/s, posteriormente se calcula con ella el peso máximo a levantar y se obtiene un valor teórico de 82.8603 kg.

Como se puede observar ambos ensayos que fueron realizados en días distintos da un resultado prácticamente igual, pero para comprobar aún más que es un resultado realista en la misma sesión de entrenamiento se realizo un levantamiento con 82,5kg lográndolo hacer pero al límite de las posibilidades sintiendo que no había prácticamente posibilidad de poner más peso a la barra, por lo que queda comprobado que con el prototipo se puede efectivamente llegar a calcular el 1RM con una precisión más que aceptable día a día.

8. CONCLUSIONES

La principal conclusión es que el prototipo ha cumplido con su objetivo de forma más que adecuada aunque habría que matizar que en vista de los resultados se ha visto que no hacía falta emplear un sensor con giróscopos y con magnetómetros que sólo con los acelerómetros es suficiente.

Por otra parte ha sido una sorpresa la estabilidad con la que la barra sube en su recorrido ascendente manteniendo la posición relativa sin rotar prácticamente sobre sí misma.

En cuanto al apartado electrónico se ha visto que el Arduino empleado ha cumplido de sobra con las necesidades que se tenían y que incluso se le podía haber dado un uso más intensivo modificando el código y eliminando la lectura de los magnetómetros con tal de poder obtener más datos de los acelerómetros y con ello más precisión aunque tampoco es algo que fuese a mejorar en mucho los resultados ya que estos ya son más precisos de lo fino que físicamente se puede hilar con la precisión de pesos en los ejercicios.

Discusión y desarrollos futuros

Por otro lado cabe concluir que en este proyecto se empezó con una idea general y un objetivo concreto el cual se ha logrado pero se tiene margen de mejora en muchos aspectos. Una mejora primordial en un posible desarrollo más avanzado del prototipo sería poder automatizar el proceso realizado con Matlab y que se calculase la velocidad del movimiento y el 1RM directamente en el dispositivo móvil al cual se envían los datos brutos de los sensores es decir generalmente en el Smartphone obteniendo así de forma instantánea los resultados sin tener que tratar los datos obtenidos del sensor con Matlab y emplear un ordenador en ello.

Otro posible paso hacia delante el cual supondría un cambio bastante drástico, de hecho tan drástico que prácticamente sería otro prototipo sería además de hacer que el Smartphone calcule los resultados con los datos brutos de los sensores hacer que este tomase directamente él mismo los datos ya que todos los smartphones actuales disponen de acelerómetros. Se debería simplemente desarrollar un soporte físico para fijar el Smartphone a la barra.

9. BIBLIOGRAFÍA

[1] ST Application Note AN3182. Tilt measurement using a low-g-3-axis accelerometer
[2] F. Landis Markley. John L. Crassidis. Fundamentals of Spacecraft Attitude Determination and Control. Springer
[3] John J. Craig. Robótica
[4] E. Rosario-Gabriel, H. Rodríguez-Cortés, M. Velasco-Villa. Estimación de la orientación de un cuerpo rígido: Estudio experimental comparativo. Departamento de Mecatrónica. CINVESTAV
[5] David Marchante. Powerexplosive Entrenamiento eficiente, explota tus límites. Editorial Luhu.

http://playground.arduino.cc/Main/MPU-9150

http://www.starlino.com/imu_guide.html

http://justlivinlife.skn1.com/2015/11/05/relacion-intensidad-velocidad-de-ejecucion/

10. PRESUPUESTO

En este presupuesto se recoge el coste completo de la realización del proyecto. Este presupuesto consta de cuatro apartados, costes generales, costes de mano de obra , costes de amortización de equipos y software, costes de materiales y componentes comerciales además de el coste total del proyecto.

10.1. Costes generales

En este apartado se incluye el material de oficina y los recursos que han sido necesarios para la realización de esta documentación.

MÁSTER DE INGENIERÍA MECATRÓNICA	DISEÑO Y VALIDACIÓN DE UN SISTEMA PARA LA ESTIMACIÓN DEL MOVIMIENTO	GASTOS GENERALES		
Nº ORDEN	CONCEPTO	Nº UNIDADES	PRECIO UNITARIO MATERIAL	TOTAL (€)
1.0	Material de escritorio	2	20,00	40,00
1.1	Material informático	1	75,00	75,00
Coste total de Gastos Generales en euros				115€

10.2. Costes de mano de obra

En este apartado se incluye el gasto en recursos humanos y conocimientos técnicos para la realización del proyecto.

MÁSTER DE INGENIERÍA MECATRÓNICA	DISEÑO Y VALIDACIÓN DE UN SISTEMA PARA LA ESTIMACIÓN DEL MOVIMIENTO	COSTES DE MANO DE OBRA	
Nº ORDEN	CONCEPTO	PRECIO UNITARIO (€/h)	Nº HORAS	TOTAL (€)
2.1	Adquisición de conocimientos previos	35,00	150	5250
2.2	Diseño del prototipo	35,00	180	6300
2.3	Pruebas y ensayos	35,00	320	11.200
2.4	Elaboración documentación formal	35,00	120	4200
Coste total de mano de obra en euros				26950 €

10.3. Costes de mano de obra

En este apartado se incluye el gasto de los equipos, de los programas informáticos con sus licencias de software que han sido necesarias.
Para poder rebajar el presupuesto en lo posible siempre que ha habido una alternativa de software open-source se ha utilizado.
Por otro lado no se ha adquirido ningún software nuevo ni ninguna licencia nueva para poder desarrollar el proyecto por lo que se valorará el tiempo en el que se ha usado cada software y licencia en proporción a un tiempo de amortización estimado que se fija en 4 años.
Obteniendo de la siguiente manera el coeficiente de amortización

$$Coef.de\ amortización = Tiempo\ de\ proyecto / Tiempo\ estimado\ de\ amortización$$

MÁSTER DE INGENIERÍA MECATRÓNICA	DISEÑO Y VALIDACIÓN DE UN SISTEMA PARA LA ESTIMACIÓN DEL MOVIMIENTO	COSTES DE AMORTIZACIÓN DE EQUIPOS Y SOFTWARE

Nº ORDEN	PRODUCTO	DESCRIPCION	DESARROLLADOR	VERSION/ MODELO	PRECIO (€)	COEF. AMOR	TOTAL (€)
3.0	Software	Mt Manager	Xsens	2010	-	-	0,00
3.1	Software	Bluetooth SPP	-	-	Libre	-	0,00
3.2	Software	Matlab	MathWorks	2015a	6.000	0.083	500,00
3.3	Software	Arduino	Arduino	1.6.4	Libre	-	0,00
3.4	Software	SolidWorks	SolidWorks Corp.	2014	6.500	0,042	270,83
3.5	Software	Office Professional	Microsoft	2013	550	0,042	22,91
3.6	Software	Hyperterminal	Mixrosoft	6.1	-	-	0,00
3.7	Software	Acrobat	Adobe	X Pro	659	0,01	6,59
3.8	Hardware	PC Portátil	Acer	Windows 8	800	0,125	100
3.9	Hardware	MEMS	Xsens	Mti 100 series	2.529,90	0,063	159,40
3.10	Hardware	Pan-Tilt PTU	FLIR	D46	2900	0,083	241,67
Coste total de Amortización de Equipos y Software en euros							1301,4 €

10.4. Costes de materiales

En este apartado se incluye el gasto en materiales para la fabricación del prototipo

MÁSTER DE INGENIERÍA MECATRÓNICA		DISEÑO Y VALIDACIÓN DE UN SISTEMA PARA LA ESTIMACIÓN DEL MOVIMIENTO		COSTES DE MATERIAL		
Nº ORDEN	CONCEPTO	DESCRIPCION	Nº UNIDADES	FABRICANTE	PRECIO UNITARIO MATERIAL (€)	TOTAL (€)
4.1	Placa Arduino	Mega 2560	1	Arduino	20,00	20,00
4.2	Sensor MPU-9150	MEMS	1	Sparkfun	14,00	14,00
4.3	Modulo Bluetooth	HC-06	1	Arduino	7,50	7,50
4.5	Jumpers	Cables de enganche	4	Sonytel	0,95	3.80
4.6	Tiras de pines	Macho	1	Sonytel	1,45	1,45
4.7	Tiras de pines	Hembra	1	Sonytel	1,45	1,45
4.11	Adaptador pila	Pilas 9V	1	Sonytel	2,50	2,50
4.12	Pila	9V	2	Duracell	13,90	13,90
4.13	Tornillos	Ø=1.5	2	Ferretería	0,20	0,40
Coste Total del Material en euros						64,90€

10.5. Coste Total

El coste total de ejecución material se obtiene de la suma de los anteriores costes desglosados. Para obtener el coste total del proyecto se debe añadir a este coste el beneficio industrial que se desea obtener del proyecto y el impuesto sobre el valor añadido (IVA).

MÁSTER DE INGENIERÍA MECATRÓNICA	DISEÑO Y VALIDACIÓN DE UN SISTEMA PARA LA ESTIMACIÓN DEL MOVIMIENTO	COSTE GLOBAL

CONCEPTO	TOTAL (€)
Materiales y componentes comerciales	64,90€
Mano de Obra	26.950,00 €
Equipos y Software	1301,40 €
Gastos Generales	115,00 €
Coste Total de Ejecución Material	28.431,3 €

Beneficio Industrial (10%)	2843,13€
Coste Total exento de IVA	31.274,43 €

I.V.A (21%)	6.567,63 €
Coste Total del Proyecto	37.842,06 €

El precio total del proyecto es:

TREINTA Y SIETE MIL OCHOCIENTOS CUARENTA Y DOS CON SEIS.

Fecha:
11/02/2016

Autorizado por:
Juan Carlos Álvarez Álvarez
Tutor académico del proyecto

ANEXOS

Anexo de código

Anexo de planificación

Anexo Datasheet

Anexo de código

Código principal cargado en Arduino

```
#include <Wire.h>
#include <SoftwareSerial.h>
SoftwareSerial MiPuertoSerial(10,2);//RX,TX

// Register names according to the datasheet.
// According to the InvenSense document
// "MPU-9150 Register Map and Descriptions Revision 4.0",

#define MPU9150_SELF_TEST_X       0x0D   // R/W
#define MPU9150_SELF_TEST_Y       0x0E   // R/W
#define MPU9150_SELF_TEST_Z       0x0F   // R/W
#define MPU9150_SELF_TEST_A       0x10   // R/W
#define MPU9150_SMPLRT_DIV        0x19   // R/W
#define MPU9150_CONFIG            0x1A   // R/W
#define MPU9150_GYRO_CONFIG       0x1B   // R/W
#define MPU9150_ACCEL_CONFIG      0x1C   // R/W
#define MPU9150_FF_THR            0x1D   // R/W
#define MPU9150_FF_DUR            0x1E   // R/W
#define MPU9150_MOT_THR           0x1F   // R/W
#define MPU9150_MOT_DUR           0x20   // R/W
#define MPU9150_ZRMOT_THR         0x21   // R/W
#define MPU9150_ZRMOT_DUR         0x22   // R/W
#define MPU9150_FIFO_EN           0x23   // R/W
#define MPU9150_I2C_MST_CTRL      0x24   // R/W
#define MPU9150_I2C_SLV0_ADDR     0x25   // R/W
#define MPU9150_I2C_SLV0_REG      0x26   // R/W
#define MPU9150_I2C_SLV0_CTRL     0x27   // R/W
#define MPU9150_I2C_SLV1_ADDR     0x28   // R/W
#define MPU9150_I2C_SLV1_REG      0x29   // R/W
#define MPU9150_I2C_SLV1_CTRL     0x2A   // R/W
#define MPU9150_I2C_SLV2_ADDR     0x2B   // R/W
#define MPU9150_I2C_SLV2_REG      0x2C   // R/W
#define MPU9150_I2C_SLV2_CTRL     0x2D   // R/W
#define MPU9150_I2C_SLV3_ADDR     0x2E   // R/W
#define MPU9150_I2C_SLV3_REG      0x2F   // R/W
#define MPU9150_I2C_SLV3_CTRL     0x30   // R/W
#define MPU9150_I2C_SLV4_ADDR     0x31   // R/W
#define MPU9150_I2C_SLV4_REG      0x32   // R/W
#define MPU9150_I2C_SLV4_DO       0x33   // R/W
#define MPU9150_I2C_SLV4_CTRL     0x34   // R/W
#define MPU9150_I2C_SLV4_DI       0x35   // R
#define MPU9150_I2C_MST_STATUS    0x36   // R
#define MPU9150_INT_PIN_CFG       0x37   // R/W
#define MPU9150_INT_ENABLE        0x38   // R/W
#define MPU9150_INT_STATUS        0x3A   // R
#define MPU9150_ACCEL_XOUT_H      0x3B   // R
#define MPU9150_ACCEL_XOUT_L      0x3C   // R
```

```
#define MPU9150_ACCEL_YOUT_H      0x3D   // R
#define MPU9150_ACCEL_YOUT_L      0x3E   // R
#define MPU9150_ACCEL_ZOUT_H      0x3F   // R
#define MPU9150_ACCEL_ZOUT_L      0x40   // R
#define MPU9150_TEMP_OUT_H        0x41   // R
#define MPU9150_TEMP_OUT_L        0x42   // R
#define MPU9150_GYRO_XOUT_H       0x43   // R
#define MPU9150_GYRO_XOUT_L       0x44   // R
#define MPU9150_GYRO_YOUT_H       0x45   // R
#define MPU9150_GYRO_YOUT_L       0x46   // R
#define MPU9150_GYRO_ZOUT_H       0x47   // R
#define MPU9150_GYRO_ZOUT_L       0x48   // R
#define MPU9150_EXT_SENS_DATA_00   0x49   // R
#define MPU9150_EXT_SENS_DATA_01   0x4A   // R
#define MPU9150_EXT_SENS_DATA_02   0x4B   // R
#define MPU9150_EXT_SENS_DATA_03   0x4C   // R
#define MPU9150_EXT_SENS_DATA_04   0x4D   // R
#define MPU9150_EXT_SENS_DATA_05   0x4E   // R
#define MPU9150_EXT_SENS_DATA_06   0x4F   // R
#define MPU9150_EXT_SENS_DATA_07   0x50   // R
#define MPU9150_EXT_SENS_DATA_08   0x51   // R
#define MPU9150_EXT_SENS_DATA_09   0x52   // R
#define MPU9150_EXT_SENS_DATA_10   0x53   // R
#define MPU9150_EXT_SENS_DATA_11   0x54   // R
#define MPU9150_EXT_SENS_DATA_12   0x55   // R
#define MPU9150_EXT_SENS_DATA_13   0x56   // R
#define MPU9150_EXT_SENS_DATA_14   0x57   // R
#define MPU9150_EXT_SENS_DATA_15   0x58   // R
#define MPU9150_EXT_SENS_DATA_16   0x59   // R
#define MPU9150_EXT_SENS_DATA_17   0x5A   // R
#define MPU9150_EXT_SENS_DATA_18   0x5B   // R
#define MPU9150_EXT_SENS_DATA_19   0x5C   // R
#define MPU9150_EXT_SENS_DATA_20   0x5D   // R
#define MPU9150_EXT_SENS_DATA_21   0x5E   // R
#define MPU9150_EXT_SENS_DATA_22   0x5F   // R
#define MPU9150_EXT_SENS_DATA_23   0x60   // R
#define MPU9150_MOT_DETECT_STATUS  0x61   // R
#define MPU9150_I2C_SLV0_DO        0x63   // R/W
#define MPU9150_I2C_SLV1_DO        0x64   // R/W
#define MPU9150_I2C_SLV2_DO        0x65   // R/W
#define MPU9150_I2C_SLV3_DO        0x66   // R/W
#define MPU9150_I2C_MST_DELAY_CTRL 0x67   // R/W
#define MPU9150_SIGNAL_PATH_RESET  0x68   // R/W
#define MPU9150_MOT_DETECT_CTRL    0x69   // R/W
#define MPU9150_USER_CTRL          0x6A   // R/W
#define MPU9150_PWR_MGMT_1         0x6B   // R/W
#define MPU9150_PWR_MGMT_2         0x6C   // R/W
#define MPU9150_FIFO_COUNTH        0x72   // R/W
#define MPU9150_FIFO_COUNTL        0x73   // R/W
#define MPU9150_FIFO_R_W           0x74   // R/W
#define MPU9150_WHO_AM_I           0x75   // R
#define MPU9150_TEMP_OUT_H         0X41
```

```
#define MPU9150_TEMP_OUT_L        0X42


//MPU9150 Compass
#define MPU9150_CMPS_XOUT_L       0x4A   // R
#define MPU9150_CMPS_XOUT_H       0x4B   // R
#define MPU9150_CMPS_YOUT_L       0x4C   // R
#define MPU9150_CMPS_YOUT_H       0x4D   // R
#define MPU9150_CMPS_ZOUT_L       0x4E   // R
#define MPU9150_CMPS_ZOUT_H       0x4F   // R


// I2C address 0x69 could be 0x69 depends on your wiring.
int MPU9150_I2C_ADDRESS=0x68;


//Variables where our values can be stored



void setup()
{
  //Inicializo el MiPuertoSerial
  MiPuertoSerial.begin(9600);

  // Initialize the 'Wire' class for the I2C-bus.
  Wire.begin();

  // Clear the 'sleep' bit to start the sensor.

  MPU9150_writeSensor(MPU9150_PWR_MGMT_1, 0);
  MPU9150_setupCompass();
Serial.print( "AccX  AccY AccZ   GyrX GyrY GyrZ   MagX MagY MagZ");
Serial.println();
}


void loop()
{
  MPU9150_I2C_ADDRESS =0x68;
  int magX=(double)
MPU9150_readSensor(MPU9150_CMPS_XOUT_L,MPU9150_CMPS_XOUT_H);
  int magY=(double)
MPU9150_readSensor(MPU9150_CMPS_YOUT_L,MPU9150_CMPS_YOUT_H);
  int magZ=(double)
MPU9150_readSensor(MPU9150_CMPS_ZOUT_L,MPU9150_CMPS_ZOUT_H);
  int gyrX=(double)
MPU9150_readSensor(MPU9150_GYRO_XOUT_L,MPU9150_GYRO_XOUT_H);
  int gyrY=(double)
MPU9150_readSensor(MPU9150_GYRO_YOUT_L,MPU9150_GYRO_YOUT_H);
  int gyrZ=(double)
MPU9150_readSensor(MPU9150_GYRO_ZOUT_L,MPU9150_GYRO_ZOUT_H);
```

```
  int acelX=(double)
MPU9150_readSensor(MPU9150_ACCEL_XOUT_L,MPU9150_ACCEL_XOUT_H);
  int acelY=(double)
MPU9150_readSensor(MPU9150_ACCEL_YOUT_L,MPU9150_ACCEL_YOUT_H);
  int acelZ=(double)
MPU9150_readSensor(MPU9150_ACCEL_ZOUT_L,MPU9150_ACCEL_ZOUT_H);
  int temp=(double)
MPU9150_readSensor(MPU9150_TEMP_OUT_L,MPU9150_TEMP_OUT_H);

  MiPuertoSerial.print(acelX);
  MiPuertoSerial.print(",");
  MiPuertoSerial.print(acelY);
  MiPuertoSerial.print(",");
  MiPuertoSerial.print(acelZ);
  MiPuertoSerial.print(",");

  MiPuertoSerial.print(gyrX);
  MiPuertoSerial.print(",");
  MiPuertoSerial.print(gyrY);
  MiPuertoSerial.print(",");
  MiPuertoSerial.print(gyrZ);
  MiPuertoSerial.print(",");

  MiPuertoSerial.print(magX);
  MiPuertoSerial.print(",");
  MiPuertoSerial.print(magY);
  MiPuertoSerial.print(",");
  MiPuertoSerial.print(magZ);
  MiPuertoSerial.println();
  //delay(1000);

}

//http://pansenti.wordpress.com/2013/03/26/pansentis-invensense-mpu-9150-software-for-
arduino-is-now-on-github/
//Thank you to pansenti for setup code.
//I will documented this one later.

void MPU9150_setupCompass(){

    MPU9150_I2C_ADDRESS = 0x0C;      //change Adress to Compass

  MPU9150_writeSensor(0x0A, 0x00); //CNTL PowerDownMode
  MPU9150_writeSensor(0x0A, 0x08); //SelfTest
  MPU9150_writeSensor(0x0A, 0x00); //PowerDownMode


 MPU9150_I2C_ADDRESS = 0x68;      //change Adress to MPU

  MPU9150_writeSensor(0x24, 0x40); //Wait for Data at Slave0
  MPU9150_writeSensor(0x25, 0x8C); //Set i2c address at slave0 at 0x0C
  MPU9150_writeSensor(0x26, 0x02); //Set where reading at slave 0 starts
```

```
  MPU9150_writeSensor(0x27, 0x88); //set offset at start reading and enable
  MPU9150_writeSensor(0x28, 0x0C); //set i2c address at slv1 at 0x0C
  MPU9150_writeSensor(0x29, 0x0A); //Set where reading at slave 1 starts
  MPU9150_writeSensor(0x2A, 0x81); //Enable at set length to 1
  MPU9150_writeSensor(0x64, 0x01); //overvride register
  MPU9150_writeSensor(0x67, 0x03); //set delay rate
  MPU9150_writeSensor(0x01, 0x80);
  MPU9150_writeSensor(0x1A, 0x86);

  MPU9150_writeSensor(MPU9150_ACCEL_CONFIG, 0x08);//ESCALA DEL
ACELEROMETRO 0X18 +-16 / 0X00 +-2 / 0X08 +-4 / 0X10 +-8


  MPU9150_writeSensor(0x34, 0x04); //set i2c slv4 delay
  MPU9150_writeSensor(0x64, 0x00); //override register
  MPU9150_writeSensor(0x6A, 0x00); //clear usr setting
  MPU9150_writeSensor(0x64, 0x01); //override register
  MPU9150_writeSensor(0x6A, 0x20); //enable master i2c mode
  MPU9150_writeSensor(0x34, 0x13); //disable slv4
  MPU9150_writeSensor(0x1A, 0x86);




}

//////////////////////////////////////////////////////////
///////// I2C functions to get easier all values ////////////
//////////////////////////////////////////////////////////

int MPU9150_readSensor(int addrL, int addrH){
  Wire.beginTransmission(MPU9150_I2C_ADDRESS);
  Wire.write(addrL);
  Wire.endTransmission(false);

  Wire.requestFrom(MPU9150_I2C_ADDRESS, 1, true);
  byte L = Wire.read();

  Wire.beginTransmission(MPU9150_I2C_ADDRESS);
  Wire.write(addrH);
  Wire.endTransmission(false);

  Wire.requestFrom(MPU9150_I2C_ADDRESS, 1, true);
  byte H = Wire.read();

  return (int16_t)((H<<8)+L);
}

int MPU9150_readSensor(int addr){
```

```
  Wire.beginTransmission(MPU9150_I2C_ADDRESS);
  Wire.write(addr);
  Wire.endTransmission(false);

  Wire.requestFrom(MPU9150_I2C_ADDRESS, 1, true);
  return Wire.read();
}

int MPU9150_writeSensor(int addr,int data){
  Wire.beginTransmission(MPU9150_I2C_ADDRESS);
  Wire.write(addr);
  Wire.write(data);
  Wire.endTransmission(true);

  return 1;
}
```

Código para configurar el módulo bluetooth HC06 con Arduino

```
#include <SoftwareSerial.h>
SoftwareSerial BT1(10,2);//RX,TX

void setup()
  {
    Serial.begin(9600);
    Serial.println("Enter AT commands:");
    BT1.begin(9600);
  }

void loop()
  {
    if (BT1.available())
       Serial.write(BT1.read());

    if (Serial.available())
      {  String S = GetLine();
        BT1.print(S);
        Serial.println("---> " + S);
      }
}

String GetLine()
  {   String S = "" ;
    if (Serial.available())
      {    char c = Serial.read(); ;
           while ( c != '\n')          //Hasta que el caracter sea intro
             {     S = S + c ;
                   delay(25) ;
                   c = Serial.read();
             }
           return( S + '\n') ;
      }
  }
```

Anexo Scripts de Matlab

Script para representar el giro según 3 convenios de Euler

```
    A_acelerometros=[   1.00777272995569 -0.000843392495858512
0.0553193603953005;%Matriz de calibracion de los acelerómetros
                        0.00236924278902134 0.990479763669570
0.0860789733204620;
                        0.00933624482358070 -0.0891368885841005
0.991307863455836;
                        -0.0438512536282694 -1.45096163410852e-05 -
0.0392679436268272];

    A_magnetometros=[  -2.47015963356884 2.41399164652808
0.909013316870544;%Matriz de calibración de los magnetómetros
                        -0.347598771961752 -0.0704870064120331
1.30218147029183;
                        0.0770435139915094 1.27831034697555
0.280149242251529;
                        -1.75194135686915 1.70345951668685
0.606717934629462];

    Counter=[1:length(VarName1)]; %Generamos la variable Counter para hacer
los plots
    Acc_X=VarName1;
    Acc_Y=VarName2;
    Acc_Z=VarName3;
    Mag_X=VarName8; %cambiamos el eje para que conicida con el de los acc y
lo giros
    Mag_Y=VarName7; %cambiamos el eje para que conicida con el de los acc y
lo giros
    Mag_Z=-VarName9; %cambiamos el eje para que conicida con el de los acc y
lo giros

    %Ahora hacemos un cambio de ejes, pasamos de los ejes del sensor a los
EJES
    %de la tabla, donde Z es el vertical (Yaw) X es el (Roll) e Y es el
(Pitch)



for i=1:length(Mag_X)
    MX(i,1)=Mag_X(i)/sqrt(Mag_X(i)^2+Mag_Y(i)^2+Mag_Z(i)^2);%hacemos el
vector unitario
    MY(i,1)=Mag_Y(i)/sqrt(Mag_X(i)^2+Mag_Y(i)^2+Mag_Z(i)^2);%hacemos el
vector unitario
    MZ(i,1)=Mag_Z(i)/sqrt(Mag_X(i)^2+Mag_Y(i)^2+Mag_Z(i)^2);%hacemos el
vector unitario
end

for i=1:length(Mag_X)
    AX(i,1)=Acc_X(i)/sqrt(Acc_X(i)^2+Acc_Y(i)^2+Acc_Z(i)^2);%hacemos el
vector unitario
    AY(i,1)=Acc_Y(i)/sqrt(Acc_X(i)^2+Acc_Y(i)^2+Acc_Z(i)^2);%hacemos el
vector unitario
    AZ(i,1)=Acc_Z(i)/sqrt(Acc_X(i)^2+Acc_Y(i)^2+Acc_Z(i)^2);%hacemos el
vector unitario
end

Acc=[AX AY AZ ones(length(AX),1)]*A_acelerometros;
```

```
Mag=[MX MY MZ ones(length(MX),1)]*A_magnetometros;

%Ahora hacemos un cambio de ejes, pasamos de los ejes del sensor a los EJES
%de la tabla, donde Z es el vertical (Yaw) X es el (Roll) e Y es el (Pitch)
%Acc=[AccSC(:,3) AccSC(:,2) AccSC(:,1)];
%Mag=[MagSC(:,3) MagSC(:,2) MagSC(:,1)];




for i=1:length(AX)
    %v es el observador/nave
    w1(i,:)=Acc(i,:);
    w2(i,:)= cross(w1(i,:),Mag(i,:))/norm(cross(w1(i,:),Mag(i,:)));
    w3(i,:)=cross(w1(i,:),w2(i,:))/norm(cross(w1(i,:),w2(i,:)));
    %w es la referencia
    v1(i,:)=Acc(1,:);
    v2(i,:)=cross(v1(1,:),Mag(1,:))/norm(cross(v1(1,:),Mag(1,:)));
    v3(i,:)=cross(v1(1,:),v2(1,:))/norm(cross(v1(1,:),v2(1,:)));
    % Matriz de transformacion= matriz observador * matriz referencia
traspuesta
    %A(i,:,:)=v1(i,:)'*w1(i,:)+v2(i,:)'*w2(i,:)+v3(i,:)'*w3(i,:);
   %A(i,:,:)=w1(i,:)'*v1(i,:)+w2(i,:)'*v2(i,:)+w3(i,:)'*v3(i,:); %ESTA ES
   %LA QUE USABA ALHUERTO Y ESTABA MAL.
  %A_prueba(i,:,:)=[w1(i,:);w2(i,:);w3(i,:)]'*[v1(i,:);v2(i,:);v3(i,:)];
  A(i,:,:)=[w1(i,:);w2(i,:);w3(i,:)]*[v1(i,:);v2(i,:);v3(i,:)]';%ESTA ES LA
CORRECTA
  %XYZ(i,:,:)=w1(i,:)'*v1(i,:)+w2(i,:)'*v2(i,:)+w3(i,:)'*v3(i,:);
end
prueba=[w1(1,:);w2(1,:);w3(1,:)]*[v1(1,:);v2(1,:);v3(1,:)]'
prueba200=[w1(200,:);w2(200,:);w3(200,:)]*[v1(200,:);v2(200,:);v3(200,:)]'
%prueba350=[w1(350,:);w2(350,:);w3(350,:)]*[v1(350,:);v2(350,:);v3(350,:)]'


for i=1:length(AX)%XYZ

    pitch(i)=atan2(A(i,1,3),sqrt(A(i,1,1)^2+A(i,1,2)^2))*(180/pi);
    yaw(i)=atan2(A(i,1,2)/-cosd(pitch(i)),A(i,1,1)/cosd(pitch(i)))*(180/pi);
    roll(i)=atan2(A(i,2,3)/-cosd(pitch(i)),A(i,3,3)/cosd(pitch(i)))*(180/pi);

end
subplot(3,1,1)
plot(Counter,roll,'b',Counter,pitch,'g',Counter,yaw,'r');



for i=1:length(AX)%YXZ

    roll(i)=atan2(-A(i,2,3),sqrt(A(i,2,1)^2+A(i,2,2)^2))*(180/pi);
    yaw(i)=atan2(A(i,2,1)/cosd(roll(i)),A(i,2,2)/cosd(roll(i)))*(180/pi);
    pitch(i)=atan2(A(i,1,3)/cosd(roll(i)),A(i,3,3)/cosd(roll(i)))*(180/pi);

end
subplot(3,1,2)
plot(Counter,roll,'b',Counter,pitch,'g',Counter,yaw,'r');


for i=1:length(AX)%ZYX

    pitch(i)=atan2(-A(i,3,1),sqrt(A(i,1,1)^2+A(i,2,1)^2))*(180/pi);
```

```
    yaw(i)=atan2(A(i,2,1)/cosd(pitch(i)),A(i,1,1)/cosd(pitch(i)))*(180/pi);
    roll(i)=atan2(A(i,3,2)/cosd(pitch(i)),A(i,3,3)/cosd(pitch(i)))*(180/pi);

end
subplot(3,1,3)
plot(Counter,roll,'b',Counter,pitch,'g',Counter,yaw,'r');
```

Script para Plot de las triadas considerando un giro en el eje X del sensor

```
quiver3(0,0,0,w1(1,1),w1(1,2),w1(1,3));
hold
quiver3(0,0,0,w2(1,1),w2(1,2),w2(1,3));
quiver3(0,0,0,w3(1,1),w3(1,2),w3(1,3));
quiver3(0,0,0,w1(469,1),w1(469,2),w1(469,3));
quiver3(0,0,0,w2(469,1),w2(469,2),w2(469,3));
quiver3(0,0,0,w3(469,1),w3(469,2),w3(469,3));

axis equal
%Se hace plot del vector Aceleracion
      %quiver3(0,0,0,Acc(1,1),Acc(1,2),Acc(1,3));
      %quiver3(0,0,0,Acc(469,1),Acc(469,2),Acc(469,3));
%Posicion intermedia
      %quiver3(0,0,0,w2(200,1),w2(200,2),w2(200,3));
      %quiver3(0,0,0,w1(200,1),w1(200,2),w1(200,3));
      %quiver3(0,0,0,w3(200,1),w3(200,2),w3(200,3));
```

Script para Plot de las triadas considerando un giro en el eje Y del sensor

```
quiver3(0,0,0,w1(1,1),w1(1,2),w1(1,3));
hold
quiver3(0,0,0,w2(1,1),w2(1,2),w2(1,3));
quiver3(0,0,0,w3(1,1),w3(1,2),w3(1,3));
quiver3(0,0,0,w1(569,1),w1(569,2),w1(569,3));
quiver3(0,0,0,w2(569,1),w2(569,2),w2(569,3));
quiver3(0,0,0,w3(569,1),w3(569,2),w3(569,3));

axis equal
%Se hace plot del vector Aceleracion
      %quiver3(0,0,0,Acc(1,1),Acc(1,2),Acc(1,3));
      %quiver3(0,0,0,Acc(596,1),Acc(569,2),Acc(569,3));
%Posicion intermedia
      %quiver3(0,0,0,w2(200,1),w2(200,2),w2(200,3));
      %quiver3(0,0,0,w1(200,1),w1(200,2),w1(200,3));
      %quiver3(0,0,0,w3(200,1),w3(200,2),w3(200,3));
```

Script para Plot de las triadas considerando un giro en el eje Z del sensor

```
quiver3(0,0,0,w1(1,1),w1(1,2),w1(1,3));
hold
quiver3(0,0,0,w2(1,1),w2(1,2),w2(1,3));
quiver3(0,0,0,w3(1,1),w3(1,2),w3(1,3));
quiver3(0,0,0,w1(476,1),w1(476,2),w1(476,3));
quiver3(0,0,0,w2(476,1),w2(476,2),w2(476,3));
quiver3(0,0,0,w3(476,1),w3(476,2),w3(476,3));

axis equal
%Se hace plot del vector Aceleracion
    %quiver3(0,0,0,Acc(1,1),Acc(1,2),Acc(1,3));
    %quiver3(0,0,0,Acc(476,1),Acc(476,2),Acc(476,3));
 %Posicion intermedia
    %quiver3(0,0,0,w2(200,1),w2(200,2),w2(200,3));
    %quiver3(0,0,0,w1(200,1),w1(200,2),w1(200,3));
    %quiver3(0,0,0,w3(200,1),w3(200,2),w3(200,3));
```

Script para el cálculo de los ángulos entre vectores

```
%angulo formado por el vector aceleración inicial y el final
angulo_entre_vectores_aceleracion=
acosd((Acc(1,1)*Acc(469,1)+Acc(1,2)*Acc(469,2)+Acc(1,3)*Acc(469,3))/(sqrt(Acc
(1,1)^2+Acc(1,2)^2+Acc(1,3)^2)*sqrt(Acc(469,1)^2+Acc(469,2)^2+Acc(469,3)^2)))

%angulo formado por el vector magnético inicial y el final
angulo_entre_vectores_magnetico=
acosd((Mag(1,1)*Mag(469,1)+Mag(1,2)*Mag(469,2)+Mag(1,3)*Mag(469,3))/(sqrt(Mag
(1,1)^2+Mag(1,2)^2+Mag(1,3)^2)*sqrt(Mag(469,1)^2+Mag(469,2)^2+Mag(469,3)^2)))

%angulo formado por el vector aceleración inicial y el magnetico inicial
angulo_entre_vectores_aceleracion_inicial_magnetico_inicial=
acosd((Acc(1,1)*Mag(1,1)+Acc(1,2)*Mag(1,2)+Acc(1,3)*Mag(1,3))/(sqrt(Acc(1,1)^
2+Acc(1,2)^2+Acc(1,3)^2)*sqrt(Mag(1,1)^2+Mag(1,2)^2+Mag(1,3)^2)))

%angulo formado por el vector aceleración final y el magnetico final
angulo_entre_vectores_aceleracion_final_magnetico_final=
acosd((Acc(469,1)*Mag(469,1)+Acc(469,2)*Mag(469,2)+Acc(469,3)*Mag(469,3))/(sq
rt(Acc(469,1)^2+Acc(469,2)^2+Acc(469,3)^2)*sqrt(Mag(469,1)^2+Mag(469,2)^2+Mag
(469,3)^2)))
```

Script para visualizar los datos de los levantamientos en banca

```
A_acelerometros=[   1.00777272995569 -0.000843392495858512
0.0553193603953005;%Matriz de calibracion de los acelerómetros
                        0.00236924278902134 0.990479763669570
0.0860789733204620;
                        0.00933624482358070 -0.0891368885841005
0.991307863455836;
                        -0.0438512536282694 -1.45096163410852e-05 -
0.0392679436268272];

    A_magnetometros=[  -2.47015963356884 2.41399164652808
0.909013316870544;%Matriz de calibración de los magnetómetros
                        -0.347598771961752 -0.0704870064120331
1.30218147029183;
                        0.0770435139915094 1.27831034697555
0.280149242251529;
                        -1.75194135686915 1.70345951668685
0.606717934629462];

    Counter=[1:length(VarName1)]; %Generamos la variable Counter para hacer
los plots
    Acc_X=VarName1;
    Acc_Y=VarName2;
    Acc_Z=VarName3;
    Mag_X=VarName5; %cambiamos el eje para que conicida con el de los acc y
lo giros
    Mag_Y=VarName4; %cambiamos el eje para que conicida con el de los acc y
lo giros
    Mag_Z=-VarName6; %cambiamos el eje para que conicida con el de los acc y
lo giros

    %Ahora hacemos un cambio de ejes, pasamos de los ejes del sensor a los
EJES
    %de la tabla, donde Z es el vertical (Yaw) X es el (Roll) e Y es el
(Pitch)



for i=1:length(VarName1)
    MX(i,1)=Mag_X(i)/sqrt(Mag_X(i)^2+Mag_Y(i)^2+Mag_Z(i)^2);%hacemos el
vector unitario
    MY(i,1)=Mag_Y(i)/sqrt(Mag_X(i)^2+Mag_Y(i)^2+Mag_Z(i)^2);%hacemos el
vector unitario
    MZ(i,1)=Mag_Z(i)/sqrt(Mag_X(i)^2+Mag_Y(i)^2+Mag_Z(i)^2);%hacemos el
vector unitario
end

for i=1:length(VarName1)
    AX(i,1)=Acc_X(i)/sqrt(Acc_X(i)^2+Acc_Y(i)^2+Acc_Z(i)^2);%hacemos el
vector unitario
    AY(i,1)=Acc_Y(i)/sqrt(Acc_X(i)^2+Acc_Y(i)^2+Acc_Z(i)^2);%hacemos el
vector unitario
    AZ(i,1)=Acc_Z(i)/sqrt(Acc_X(i)^2+Acc_Y(i)^2+Acc_Z(i)^2);%hacemos el
vector unitario
end

Acc=[AX AY AZ ones(length(AX),1)]*A_acelerometros;
Mag=[MX MY MZ ones(length(MX),1)]*A_magnetometros;
```

```
%Ahora hacemos un cambio de ejes, pasamos de los ejes del sensor a los EJES
%de la tabla, donde Z es el vertical (Yaw) X es el (Roll) e Y es el (Pitch)
%Acc=[AccSC(:,3) AccSC(:,2) AccSC(:,1)];
%Mag=[MagSC(:,3) MagSC(:,2) MagSC(:,1)];




for i=1:length(VarName1)
    %v es el observador/nave
    w1(i,:)=Acc(i,:);
    w2(i,:)= cross(w1(i,:),Mag(i,:))/norm(cross(w1(i,:),Mag(i,:)));
    w3(i,:)=cross(w1(i,:),w2(i,:))/norm(cross(w1(i,:),w2(i,:)));
    %w es la referencia
    v1(i,:)=Acc(1,:);
    v2(i,:)=cross(v1(1,:),Mag(1,:))/norm(cross(v1(1,:),Mag(1,:)));
    v3(i,:)=cross(v1(1,:),v2(1,:))/norm(cross(v1(1,:),v2(1,:)));
    % Matriz de transformacion= matriz observador * matriz referencia
traspuesta
    %A(i,:,:)=v1(i,:)'*w1(i,:)+v2(i,:)'*w2(i,:)+v3(i,:)'*w3(i,:);
   %A(i,:,:)=w1(i,:)'*v1(i,:)+w2(i,:)'*v2(i,:)+w3(i,:)'*v3(i,:); %ESTA ES
   %LA QUE USABA ALHUERTO Y ESTABA MAL.
  %A_prueba(i,:,:)=[w1(i,:);w2(i,:);w3(i,:)]'*[v1(i,:);v2(i,:);v3(i,:)];
  A(i,:,:)=[w1(i,:);w2(i,:);w3(i,:)]*[v1(i,:);v2(i,:);v3(i,:)]';%ESTA ES LA
CORRECTA
  %XYZ(i,:,:)=w1(i,:)'*v1(i,:)+w2(i,:)'*v2(i,:)+w3(i,:)'*v3(i,:);
end
prueba=[w1(1,:);w2(1,:);w3(1,:)]*[v1(1,:);v2(1,:);v3(1,:)]'
%prueba100=[w1(100,:);w2(100,:);w3(100,:)]*[v1(100,:);v2(100,:);v3(100,:)]'
%prueba350=[w1(350,:);w2(350,:);w3(350,:)]*[v1(350,:);v2(350,:);v3(350,:)]'


%////////////////////////////////////////////////////////////////PLOT XYZ
for i=1:length(VarName1)%XYZ

    pitch(i)=atan2(A(i,1,3),sqrt(A(i,1,1)^2+A(i,1,2)^2))*(180/pi);
    yaw(i)=atan2(A(i,1,2)/-cosd(pitch(i)),A(i,1,1)/cosd(pitch(i)))*(180/pi);
    roll(i)=atan2(A(i,2,3)/-cosd(pitch(i)),A(i,3,3)/cosd(pitch(i)))*(180/pi);

end
subplot(3,1,1)
plot(Counter,roll,'b',Counter,pitch,'g',Counter,yaw,'r');


%////////////////////////////////////////////////////////////////PLOT YXZ
for i=1:length(VarName1)%YXZ

    roll(i)=atan2(-A(i,2,3),sqrt(A(i,2,1)^2+A(i,2,2)^2))*(180/pi);
    yaw(i)=atan2(A(i,2,1)/cosd(roll(i)),A(i,2,2)/cosd(roll(i)))*(180/pi);
    pitch(i)=atan2(A(i,1,3)/cosd(roll(i)),A(i,3,3)/cosd(roll(i)))*(180/pi);

end
subplot(3,1,2)
plot(Counter,roll,'b',Counter,pitch,'g',Counter,yaw,'r');



%////////////////////////////////////////////////////////////////PLOT ZYX
for i=1:length(VarName1)%ZYX
```

```
    pitch(i)=atan2(-A(i,3,1),sqrt(A(i,1,1)^2+A(i,2,1)^2))*(180/pi);
    yaw(i)=atan2(A(i,2,1)/cosd(pitch(i)),A(i,1,1)/cosd(pitch(i)))*(180/pi);
    roll(i)=atan2(A(i,3,2)/cosd(pitch(i)),A(i,3,3)/cosd(pitch(i)))*(180/pi);

end
subplot(3,1,3)
plot(Counter,roll,'b',Counter,pitch,'g',Counter,yaw,'r');



%//////////////////////PLOT VECTORES TRIADA INICIAL Y FINAL////////////

figure

quiver3(0,0,0,w1(1,1),w1(1,2),w1(1,3));
hold
quiver3(0,0,0,w2(1,1),w2(1,2),w2(1,3));
quiver3(0,0,0,w3(1,1),w3(1,2),w3(1,3));
%quiver3(0,0,0,w1(173,1),w1(173,2),w1(173,3));
%quiver3(0,0,0,w2(173,1),w2(173,2),w2(173,3));
%quiver3(0,0,0,w3(173,1),w3(173,2),w3(173,3));

axis equal


%//////////////////////PLOT ACELERACIONES/////////////////////////////////
figure
plot(Counter,VarName1,'b',Counter,VarName2,'g',Counter,VarName3,'r');

%////////////////////////////ACELERACION TOTAL//////////////////////////
for i=1:length(VarName1)
    Aceleracion_total(i)=sqrt(Acc(i,1)^2+Acc(i,2)^2+Acc(i,3)^2);%hacemos el
vector unitario

end
figure
plot(Counter,Aceleracion_total,'b');

%////////////////////////////ACELERACION TOTAL
netooooooooooo/////////////////////////
for i=1:length(VarName1)

Aceleracion_total_neto(i)=sqrt((VarName1(i))^2+(VarName2(i))^2+(VarName3(i))^
2);
end
figure
plot(Counter,Aceleracion_total_neto,'b');
```

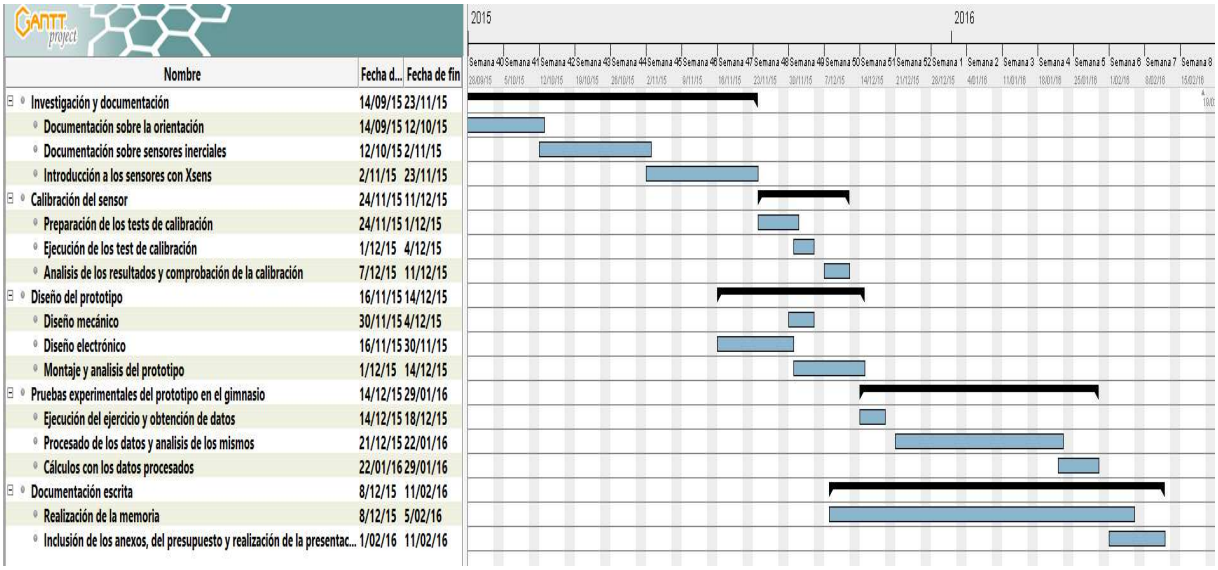
Script para calcular la velocidad del movimiento de la barra

```
velocidad_instantanea=cumsum(press_g(:,2))*1/25/2;
velocidad_media=sum(velocidad_instantanea)/length(press_g(:,2))
```

Script para calcular el 1RM a partir de la velocidad

```
porcentaje1RM=11.4196*velocidad_media^2-81.904*velocidad_media+114.03;
oneRM=76.5*100/porcentaje1RM
```

Anexo de planificación



Nombre	Fecha d...	Fecha de fin
Investigación y documentación	14/09/15	23/11/15
Documentación sobre la orientación	14/09/15	12/10/15
Documentación sobre sensores inerciales	12/10/15	2/11/15
Introducción a los sensores con Xsens	2/11/15	23/11/15
Calibración del sensor	24/11/15	11/12/15
Preparación de los tests de calibración	24/11/15	1/12/15
Ejecución de los test de calibración	1/12/15	4/12/15
Analisis de los resultados y comprobación de la calibración	7/12/15	11/12/15
Diseño del prototipo	16/11/15	14/12/15
Diseño mecánico	30/11/15	4/12/15
Diseño electrónico	16/11/15	30/11/15
Montaje y analisis del prototipo	1/12/15	14/12/15
Pruebas experimentales del prototipo en el gimnasio	14/12/15	29/01/16
Ejecución del ejercicio y obtención de datos	14/12/15	18/12/15
Procesado de los datos y analisis de los mismos	21/12/15	22/01/16
Cálculos con los datos procesados	22/01/16	29/01/16
Documentación escrita	8/12/15	11/02/16
Realización de la memoria	8/12/15	5/02/16
Inclusión de los anexos, del presupuesto y realización de la presentac...	1/02/16	11/02/16

Anexo Datasheet

MPU-9150
Product Specification
Revision 4.0

CONTENTS

1 Revision History

Revision Date	Revision	Description
5/27/2011	1.0	Initial Release of Product Specification
06/14/2011	2.0	Modified for Rev C Silicon (sections 5.2, 6.2, 6.4, 6.6, 8.2, 8.3, 8.4) Edits for clarity (several sections)
10/21/2011	2.1	Updated Supply current vs. operating modes (sections 5.3, 5.4, 6.4) Modified Self-Test Response of Accelerometers (section 6.2) Modified absolute maximum rating for acceleration (section 6.9) Updated latch up current rating (sections 6.9, 12.2) Modified package dimensions and PCB design guidelines (sections 11.2, 11.3) Updated assembly precautions (section 11.4) Updated qualification test plan (section 12.2) Edits for clarity (several sections)
10/24/2011	3.0	Modified for Rev D Silicon (sections 6.2, 8.2, 8.3, 8.4) Edits for Clarity (several sections)
12/23/2011	3.1	Updated package dimensions (section 11.2)
5/14/2012	4.0	Added Gyroscope specifications (section 6.1) Added Accelerometer specifications (section 6.2) Updated Electrical Other Common Specifications (section 6.3) Updated latch-up information (section 6.9) Updated Block Diagram (section 7.5) Update Self-Test description (section 7.13) Updated PCB design guidelines (section 11.3) Updated packing and shipping information (sections 11.8, 11.9, 11.10, 11.11) Updated reliability references (section 12.2)

2 Purpose and Scope

This product specification provides preliminary information regarding the electrical specification and design related information for the MPU-9150™ Motion Processing Unit™ or MPU™.

Electrical characteristics are based upon design analysis and simulation results only. Specifications are subject to change without notice. Final specifications will be updated based upon characterization of production silicon. For references to register map and descriptions of individual registers, please refer to the MPU-9150 Register Map and Register Descriptions document.

3 Product Overview

3.1 MPU-9150 Overview

MotionInterface™ is becoming a "must-have" function being adopted by smartphone and tablet manufacturers due to the enormous value it adds to the end user experience. In smartphones, it finds use in applications such as gesture commands for applications and phone control, enhanced gaming, augmented reality, panoramic photo capture and viewing, and pedestrian and vehicle navigation. With its ability to precisely and accurately track user motions, MotionTracking technology can convert handsets and tablets into powerful 3D intelligent devices that can be used in applications ranging from health and fitness monitoring to location-based services. Key requirements for MotionInterface enabled devices are small package size, low power consumption, high accuracy and repeatability, high shock tolerance, and application specific performance programmability – all at a low consumer price point.

The MPU-9150 is the world's first integrated 9-axis MotionTracking device that combines a 3-axis MEMS gyroscope, a 3-axis MEMS accelerometer, a 3-axis MEMS magnetometer and a Digital Motion Processor™ (DMP™) hardware accelerator engine. The MPU-9150 is an ideal solution for handset and tablet applications, game controllers, motion pointer remote controls, and other consumer devices. The MPU-9150's 9-axis MotionFusion combines acceleration and rotational motion plus heading information into a single data stream for the application. This MotionProcessing™ technology integration provides a smaller footprint and has inherent cost advantages compared to discrete gyroscope, accelerometer, plus magnetometer solutions. The MPU-9150 is also designed to interface with multiple non-inertial digital sensors, such as pressure sensors, on its auxiliary I^2C port to produce a 10-Axis sensor fusion output. The MPU-9150 is a 3rd generation motion processor and is footprint compatible with the MPU-60X0 and MPU-30X0 families.

The MPU-9150 features three 16-bit analog-to-digital converters (ADCs) for digitizing the gyroscope outputs ,three 16-bit ADCs for digitizing the accelerometer outputs and three 13-bit ADCs for digitizing the magnetometer outputs. For precision tracking of both fast and slow motions, the parts feature a user-programmable gyroscope full-scale range of ±250, ±500, ±1000, and ±2000°/sec (dps), a user-programmable accelerometer full-scale range of ±2*g*, ±4*g*, ±8*g*, and ±16*g*, and a magnetometer full-scale range of ±1200µT.

The MPU-9150 is a multi-chip module (MCM) consisting of two dies integrated into a single LGA package. One die houses the 3-Axis gyroscope and the 3-Axis accelerometer. The other die houses the AK8975 3-Axis magnetometer from Asahi Kasei Microdevices Corporation.

An on-chip 1024 Byte FIFO buffer helps lower system power consumption by allowing the system processor to read the sensor data in bursts and then enter a low-power mode as the MPU collects more data. With all the necessary on-chip processing and sensor components required to support many motion-based use cases, the MPU-9150 uniquely supports a variety of advanced motion-based applications entirely on-chip. The MPU-9150 thus enables low-power MotionProcessing in portable applications with reduced processing requirements for the system processor. By providing an integrated MotionFusion output, the DMP in the MPU-9150 offloads the intensive MotionProcessing computation requirements from the system processor, minimizing the need for frequent polling of the motion sensor output.

Communication with all registers of the device is performed using I^2C at 400kHz. Additional features include an embedded temperature sensor and an on-chip oscillator with ±1% variation over the operating temperature range.

By leveraging its patented and volume-proven Nasiri-Fabrication platform, which integrates MEMS wafers with companion CMOS electronics through wafer-level bonding, InvenSense has driven the MPU-9150 package size down to a revolutionary footprint of 4x4x1mm (LGA), while providing the highest performance, lowest noise, and the lowest cost semiconductor packaging required for handheld consumer electronic devices. The part features a robust 10,000*g* shock tolerance, and has programmable low-pass filters for the gyroscopes, accelerometers, magnetometers, and the on-chip temperature sensor.

4 Applications

- *BlurFree*™ technology (for Video/Still Image Stabilization)
- *AirSign*™ technology (for Security/Authentication)
- *TouchAnywhere*™ technology (for "no touch" UI Application Control/Navigation)
- *MotionCommand*™ technology (for Gesture Short-cuts)
- Motion-enabled game and application framework
- InstantGesture™ iG™ gesture recognition
- Location based services, points of interest, and dead reckoning
- Handset and portable gaming
- Motion-based game controllers
- 3D remote controls for Internet connected DTVs and set top boxes, 3D mice
- Wearable sensors for health, fitness and sports
- Toys
- Pedestrian based navigation
- Navigation
- Electronic Compass

5 Features

5.1 Gyroscope Features

The triple-axis MEMS gyroscope in the MPU-9150 includes a wide range of features:

- Digital-output X-, Y-, and Z-Axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of ±250, ±500, ±1000, and ±2000°/sec
- External sync signal connected to the FSYNC pin supports image, video and GPS synchronization
- Integrated 16-bit ADCs enable simultaneous sampling of gyros
- Enhanced bias and sensitivity temperature stability reduces the need for user calibration
- Improved low-frequency noise performance
- Digitally-programmable low-pass filter
- Factory calibrated sensitivity scale factor
- User self-test

5.2 Accelerometer Features

The triple-axis MEMS accelerometer in MPU-9150 includes a wide range of features:

- Digital-output 3-Axis accelerometer with a programmable full scale range of ±2*g*, ±4*g*, ±8*g* and ±16*g*
- Integrated 16-bit ADCs enable simultaneous sampling of accelerometers while requiring no external multiplexer
- Orientation detection and signaling
- Tap detection
- User-programmable interrupts
- High-G interrupt
- User self-test

5.3 Magnetometer Features

The triple-axis MEMS magnetometer in MPU-9150 includes a wide range of features:

- 3-axis silicon monolithic Hall-effect magnetic sensor with magnetic concentrator
- Wide dynamic measurement range and high resolution with lower current consumption.
- Output data resolution is 13 bit (0.3 µT per LSB)
- Full scale measurement range is ±1200 µT
- Self-test function with internal magnetic source to confirm magnetic sensor operation on end products

5.4 Additional Features

The MPU-9150 includes the following additional features:

- 9-Axis MotionFusion via on-chip Digital Motion Processor (DMP)
- Auxiliary master I^2C bus for reading data from external sensors (e.g., pressure sensor)
- Flexible VLOGIC reference voltage supports multiple I^2C interface voltages
- Smallest and thinnest package for portable devices: 4x4x1mm LGA
- Minimal cross-axis sensitivity between the accelerometer, gyroscope and magnetometer axes
- 1024 byte FIFO buffer reduces power consumption by allowing host processor to read the data in bursts and then go into a low-power mode as the MPU collects more data
- Digital-output temperature sensor
- User-programmable digital filters for gyroscope, accelerometer, and temp sensor
- 10,000 *g* shock tolerant

- 400kHz Fast Mode I^2C for communicating with all registers
- MEMS structure hermetically sealed and bonded at wafer level
- RoHS and Green compliant

5.5 MotionProcessing

- Internal Digital Motion Processing™ (DMP™) engine supports 3D MotionProcessing and gesture recognition algorithms
- The MPU-9150 collects gyroscope, accelerometer and magnetometer data while synchronizing data sampling at a user defined rate. The total dataset obtained by the MPU-9150 includes 3-Axis gyroscope data, 3-Axis accelerometer data, 3-Axis magnetometer data, and temperature data.
- The FIFO buffers the complete data set, reducing timing requirements on the system processor by allowing the processor burst read the FIFO data. After burst reading the FIFO data, the system processor can save power by entering a low-power sleep mode while the MPU collects more data.
- Programmable interrupt supports features such as gesture recognition, panning, zooming, scrolling, zero-motion detection, tap detection, and shake detection
- Digitally-programmable low-pass filters.
- Low-power pedometer functionality allows the host processor to sleep while the DMP maintains the step count.

5.6 Clocking

- On-chip timing generator ±1% frequency variation over full temperature range
- Optional external clock inputs of 32.768kHz or 19.2MHz

6 Electrical Characteristics

6.1 Gyroscope Specifications

VDD = 2.375V-3.465V, VLOGIC= 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
GYROSCOPE SENSITIVITY						
Full-Scale Range	FS_SEL=0		±250		º/s	
	FS_SEL=1		±500		º/s	
	FS_SEL=2		±1000		º/s	
	FS_SEL=3		±2000		º/s	
Gyroscope ADC Word Length			16		bits	
Sensitivity Scale Factor	FS_SEL=0		131		LSB/(º/s)	
	FS_SEL=1		65.5		LSB/(º/s)	
	FS_SEL=2		32.8		LSB/(º/s)	
	FS_SEL=3		16.4		LSB/(º/s)	
Sensitivity Scale Factor Tolerance	25°C	-3		+3	%	
Sensitivity Scale Factor Variation Over Temperature	-40°C to +85°C		±0.04		%/°C	
Nonlinearity	Best fit straight line; 25°C		0.2		%	
Cross-Axis Sensitivity			±2		%	
GYROSCOPE ZERO-RATE OUTPUT (ZRO)						
Initial ZRO Tolerance	Component level (25°C)		±20		º/s	
ZRO Variation Over Temperature	-40°C to +85°C		±20		º/s	
SELF-TEST RESPONSE	Change from factory trim	-14		14	%	
GYROSCOPE NOISE PERFORMANCE	**FS_SEL=0**					
Total RMS Noise	DLPFCFG=2 (92Hz)		0.06		º/s-rms	
Rate Noise Spectral Density	At 10Hz		0.005		º/s/ √ Hz	
GYROSCOPE MECHANICAL FREQUENCIES						
X-Axis		30	33	36	kHz	
Y-Axis		27	30	33	kHz	
Z-Axis		24	27	30	kHz	
LOW PASS FILTER RESPONSE	Programmable Range	5		256	Hz	
OUTPUT DATA RATE	Programmable	4		8,000	Hz	
GYROSCOPE START-UP TIME	**DLPFCFG=0**					
ZRO Settling	to ±1º/s of Final		30		ms	

6.2 Accelerometer Specifications

VDD = 2.375V-3.465V, VLOGIC= 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
ACCELEROMETER SENSITIVITY						
Full-Scale Range	AFS_SEL=0		±2		*g*	
	AFS_SEL=1		±4		*g*	
	AFS_SEL=2		±8		*g*	
	AFS_SEL=3		±16		*g*	
ADC Word Length	Output in two's complement format		16		bits	
Sensitivity Scale Factor	AFS_SEL=0		16,384		LSB/*g*	
	AFS_SEL=1		8,192		LSB/*g*	
	AFS_SEL=2		4,096		LSB/*g*	
	AFS_SEL=3		2,048		LSB/*g*	
Initial Calibration Tolerance			±3		%	
Sensitivity Change vs. Temperature	AFS_SEL=0, -40°C to +85°C		±0.02		%/°C	
Nonlinearity	Best Fit Straight Line		0.5		%	
ZERO-G OUTPUT						
Initial Calibration Tolerance	X and Y axes		±80		m*g*	
	Z axis		±150		m*g*	
Change over specified temperature – Component level -25°C to 85°C	X & Y Axis		±0.75		mg/°C	
	Z Axis		±1.50		mg/°C	
SELF-TEST RESPONSE	Change from factory trim	-14		14	%	
NOISE PERFORMANCE						
Power Spectral Density	X, Y & Z Axes, @10Hz, AFS_SEL=0 & ODR=1kHz		400		µg/ √Hz	
Total RMS Noise	AFS = 0 @100Hz		4		mg-rms	
LOW PASS FILTER RESPONSE	Programmable Range	5		260	Hz	
OUTPUT DATA RATE	Programmable Range	4		1,000	Hz	
INTELLIGENCE FUNCTION INCREMENT			32		m*g*/LSB	

6.3 Magnetometer Specifications

VDD = 2.375V-3.465V, VLOGIC= 1.8V±5% or VDD, T_A = 25°C

The information in the following table is from the AKM AK8975 datasheet.

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
MAGNETOMETER SENSITIVITY						
Full-Scale Range			±1200		µT	
ADC Word Length	Output in two's complement format		13		bits	
Sensitivity Scale Factor		0.285	0.3	0.315	µT /LSB	
ZERO-FIELD OUTPUT						
Initial Calibration Tolerance		-1000		1000	LSB	
SELF-TEST RESPONSE						
	X-axis	-100		100		
	Y-axis	-100		100	LSB	
	Z-axis	-1000		-300		

6.4 Electrical and Other Common Specifications

VDD = 2.375V-3.465V, VLOGIC= 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS		MIN	TYP	MAX	Units	Notes
TEMPERATURE SENSOR							
Range				-40 to +85		°C	
Sensitivity	Untrimmed			340		LSB/ºC	
Temperature Offset	35°C			-521		LSB	
Linearity	Best fit straight line (-40°C to +85°C)			±1		°C	
VDD POWER SUPPLY							
Operating Voltages			2.375		3.465	V	
Power Supply Ramp Rate	Monotonic ramp. Ramp rate is 10% to 90% of the final value				100	ms	
OPERATING CURRENT							
Normal Operating Current	Gyro at all rates	Gyro+Accel (Magnetometer and DMP disabled)		3.9		mA	
	Accel at 1kHz sample rate	Accel + Magnetometer (Gyro and DMP disabled)		900		µA	
	Magnetometer at 8Hz repetition rate	Magnetometer only (DMP, Gyro, and Accel disabled)		350		µA	
Accelerometer Low Power Mode Current	1.25 Hz update rate			10		µA	
	5 Hz update rate			20		µA	
	20 Hz update rate			70		µA	
	40 Hz update rate			140		µA	
Magnetometer Full Power Mode Current	100% Duty Cycle			6		mA	
Full-Chip Idle Mode Supply Current				6		µA	
VLOGIC REFERENCE VOLTAGE							
Voltage Range	VLOGIC must be ≤ VDD at all times		1.71		VDD	V	
Power Supply Ramp Rate	Monotonic ramp. Ramp rate is 10% to 90% of the final value				3	ms	
Normal Operating Current				100		µA	
TEMPERATURE RANGE							
Specified Temperature Range	Performance parameters are not applicable beyond Specified Temperature Range		-40		+85	°C	

6.5 Electrical Specifications, Continued

VDD = 2.375V-3.465V, VLOGIC= 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	Units	Notes
SERIAL INTERFACE						
I^2C Operating Frequency	All registers, Fast-mode All registers, Standard-mode			400 100	kHz kHz	
I^2C ADDRESS	AD0 = 0 AD0 = 1		1101000 1101001			
DIGITAL INPUTS (SDA, AD0, SCL, FSYNC, CLKIN)						
V_{IH}, High Level Input Voltage		0.7*VLOGIC			V	
V_{IL}, Low Level Input Voltage				0.3*VLOGIC	V	
C_I, Input Capacitance			< 5		pF	
DIGITAL OUTPUT (INT)						
V_{OH}, High Level Output Voltage	R_{LOAD}=1MΩ	0.9*VLOGIC			V	
V_{OL1}, LOW-Level Output Voltage	R_{LOAD}=1MΩ			0.1*VLOGIC	V	
V_{OL_INT1}, INT Low-Level Output Voltage	OPEN=1, 0.3mA sink Current			0.1	V	
Output Leakage Current	OPEN=1		100		nA	
t_{INT}, INT Pulse Width	LATCH_INT_EN=0		50		µs	
DIGITAL OUTPUT (CLKOUT)						
V_{OH}, High Level Output Voltage	RLOAD=1MΩ	0.9*VDD			V	
V_{OL1}, LOW-Level Output Voltage	RLOAD=1MΩ			0.1*VDD	V	

6.6 Electrical Specifications, Continued

Typical Operating Circuit of Section 7.2, VDD = 2.375V-3.465V, VLOGIC= 1.8V±5% or VDD, T_A = 25°C

Parameters	Conditions	Typical	Units	Notes
Primary I^2C I/O (SCL, SDA)				
VIL, LOW Level Input Voltage		-0.5V to 0.3*VLOGIC	V	
VIH, HIGH-Level Input Voltage		0.7*VLOGIC to VLOGIC + 0.5V	V	
Vhys, Hysteresis		0.1*VLOGIC	V	
V_{OL1}, LOW-Level Output Voltage	3mA sink current	0 to 0.4	V	
I_{OL}, LOW-Level Output Current	V_{OL} = 0.4V	3	mA	
	V_{OL} = 0.6V	5	mA	
Output Leakage Current		100	nA	
t_{of}, Output Fall Time from V_{IHmax} to V_{ILmax}	C_b bus capacitance in pF	20+0.1C_b to 250	ns	
C_I, Capacitance for Each I/O pin		< 10	pF	
Auxiliary I^2C I/O (ES_CL, ES_DA)				
V_{IL}, LOW-Level Input Voltage		-0.5 to 0.3*VDD	V	
V_{IH}, HIGH-Level Input Voltage		0.7*VDD to VDD+0.5V	V	
V_{hys}, Hysteresis		0.1*VDD	V	
V_{OL1}, LOW-Level Output Voltage	1mA sink current	0 to 0.4	V	
I_{OL}, LOW-Level Output Current	V_{OL} = 0.4V	1	mA	
	V_{OL} = 0.6V	1	mA	
Output Leakage Current		100	nA	
t_{of}, Output Fall Time from V_{IHmax} to V_{ILmax}	C_b bus cap. in pF	20+0.1C_b to 250	ns	
C_I, Capacitance for Each I/O pin		< 10	pF	

6.7 Electrical Specifications, Continued

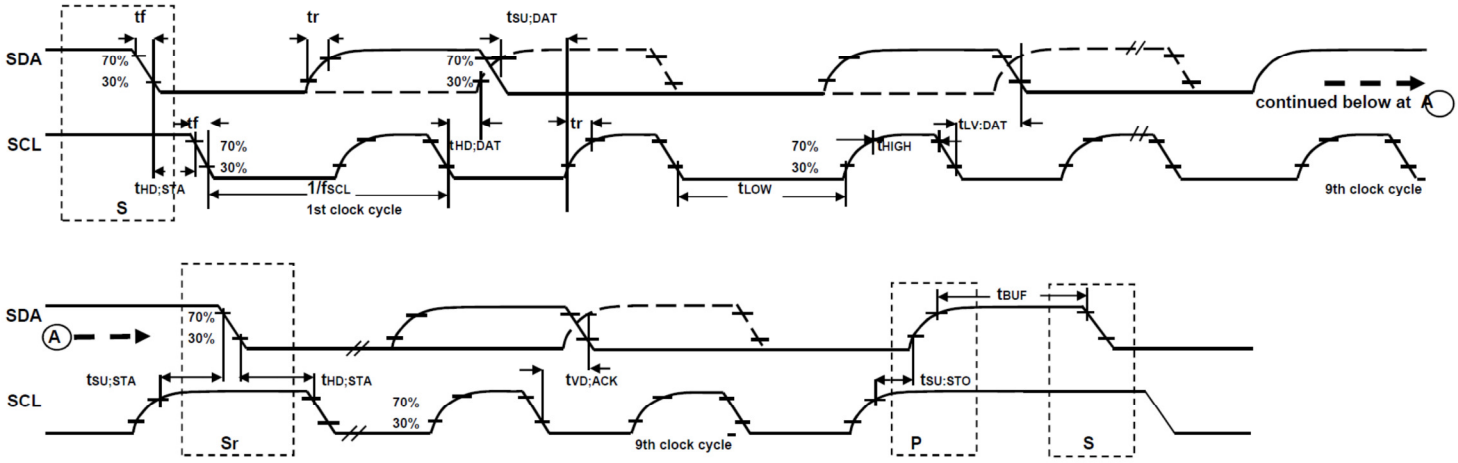
Typical Operating Circuit of Section 7.2, VDD = 2.375V-3.465V, VLOGIC= 1.8V±5% or VDD, T_A = 25°C

Parameters	Conditions	Min	Typical	Max	Units	Notes
INTERNAL CLOCK SOURCE	**CLK_SEL=0,1,2,3**					
Gyroscope Sample Rate, Fast	DLPFCFG=0 SAMPLERATEDIV = 0		8		kHz	
Gyroscope Sample Rate, Slow	DLPFCFG=1,2,3,4,5, or 6 SAMPLERATEDIV = 0		1		kHz	
Accelerometer Sample Rate			1		kHz	
Reference Clock Output	CLKOUTEN = 1		1.024		MHz	
Clock Frequency Initial Tolerance	CLK_SEL=0, 25°C	-5		+5	%	
	CLK_SEL=1,2,3; 25°C	-1		+1	%	
Frequency Variation over Temperature	CLK_SEL=0		-15 to +10		%	
	CLK_SEL=1,2,3		±1		%	
PLL Settling Time	CLK_SEL=1,2,3		1		ms	
EXTERNAL 32.768kHz CLOCK	**CLK_SEL=4**					
External Clock Frequency			32.768		kHz	
External Clock Allowable Jitter	Cycle-to-cycle rms		1 to 2		µs	
Gyroscope Sample Rate, Fast	DLPFCFG=0 SAMPLERATEDIV = 0		8.192		kHz	
Gyroscope Sample Rate, Slow	DLPFCFG=1,2,3,4,5, or 6 SAMPLERATEDIV = 0		1.024		kHz	
Accelerometer Sample Rate			1.024		kHz	
Reference Clock Output	CLKOUTEN = 1		1.0486		MHz	
PLL Settling Time			1		ms	
EXTERNAL 19.2MHz CLOCK	**CLK_SEL=5**					
External Clock Frequency			19.2		MHz	
Gyroscope Sample Rate	Full programmable range	3.9		8000	Hz	
Gyroscope Sample Rate, Fast Mode	DLPFCFG=0 SAMPLERATEDIV = 0		8		kHz	
Gyroscope Sample Rate, Slow Mode	DLPFCFG=1,2,3,4,5, or 6 SAMPLERATEDIV = 0		1		kHz	
Accelerometer Sample Rate			1		kHz	
Reference Clock Output	CLKOUTEN = 1		1.024		MHz	
PLL Settling Time			1		ms	

6.8 I²C Timing Characterization

Typical Operating Circuit of Section 7.2, VDD = 2.375V-3.465V, VLOGIC= 1.8V±5% or VDD, T_A = 25°C

Parameters	Conditions	Min	Typical	Max	Units	Notes
I²C TIMING	**I²C FAST-MODE**					
f_{SCL}, SCL Clock Frequency				400	kHz	
$t_{HD.STA}$, (Repeated) START Condition Hold Time		0.6			µs	
t_{LOW}, SCL Low Period		1.3			µs	
t_{HIGH}, SCL High Period		0.6			µs	
$t_{SU.STA}$, Repeated START Condition Setup Time		0.6			µs	
$t_{HD.DAT}$, SDA Data Hold Time		0			µs	
$t_{SU.DAT}$, SDA Data Setup Time		100			ns	
t_r, SDA and SCL Rise Time	C_b bus cap. from 10 to 400pF	20+0.1C_b		300	ns	
t_f, SDA and SCL Fall Time	C_b bus cap. from 10 to 400pF	20+0.1C_b		300	ns	
$t_{SU.STO}$, STOP Condition Setup Time		0.6			µs	
t_{BUF}, Bus Free Time Between STOP and START Condition		1.3			µs	
C_b, Capacitive Load for each Bus Line			< 400		pF	
$t_{VD.DAT}$, Data Valid Time				0.9	µs	
$t_{VD.ACK}$, Data Valid Acknowledge Time				0.9	µs	



I²C Bus Timing Diagram

6.9 Absolute Maximum Ratings

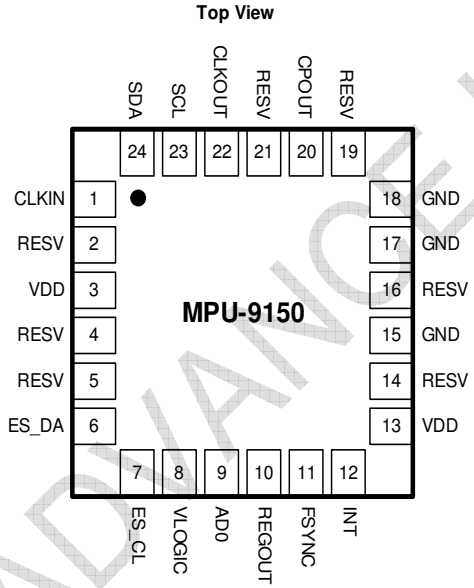
Stress above those listed as "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only and functional operation of the device at these conditions is not implied. Exposure to the absolute maximum ratings conditions for extended periods may affect device reliability.

Parameter	Rating
Supply Voltage, VDD	-0.5V to +6V
VLOGIC Input Voltage Level	-0.5V to VDD + 0.5V
REGOUT	-0.5V to 2V
Input Voltage Level (CLKIN, AUX_DA, AD0, FSYNC, INT, SCL, SDA)	-0.5V to VDD + 0.5V
CPOUT (2.5V ≤ VDD ≤ 3.6V)	-0.5V to 30V
Acceleration (Any Axis, unpowered)	10,000g for 0.2ms
Operating Temperature Range	-40°C to +105°C
Storage Temperature Range	-40°C to +125°C
Electrostatic Discharge (ESD) Protection	2kV (HBM); 200V (MM)
Latch-up	JEDEC Class II (2),125°C ±100mA

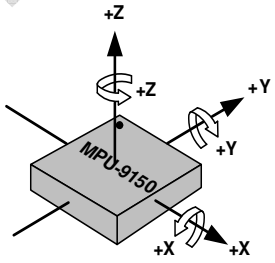
7 Applications Information

7.1 Pin Out and Signal Description

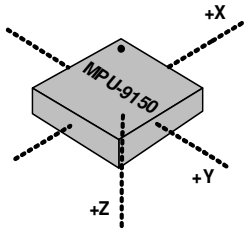
Pin Number	Pin Name	Pin Description
1	CLKIN	Optional external reference clock input. Connect to GND if unused.
6	ES_DA	Auxiliary I^2C master serial data
7	ES_CL	Auxiliary I^2C Master serial clock
8	VLOGIC	Digital I/O supply voltage
9	AD0	I^2C Slave Address LSB (AD0)
10	REGOUT	Regulator filter capacitor connection
11	FSYNC	Frame synchronization digital input. Connect to GND if unused.
12	INT	Interrupt digital output (totem pole or open-drain)
3, 13	VDD	Power supply voltage and Digital I/O supply voltage
15, 17,18	GND	Power supply ground
20	CPOUT	Charge pump capacitor connection
22	CLKOUT	System clock output
23	SCL	I^2C serial clock (SCL)
24	SDA	I^2C serial data (SDA)
2, 4, 5, 14, 16, 19, 21	RESV	Reserved. Do not connect.



LGA Package
24-pin, 4mm x 4mm x 1mm

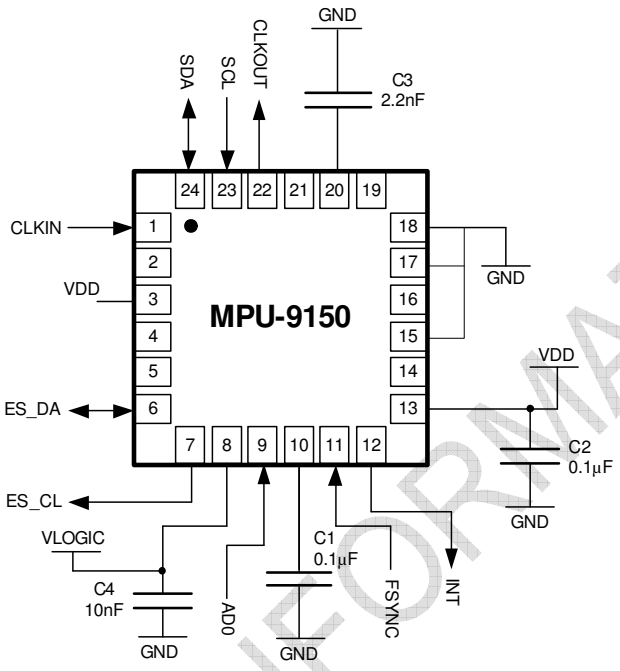


Orientation of Axes of Sensitivity and Polarity of Rotation for Acce& Gyro



Orientation of Axes of Sensitivity for Magnetometer

7.2 Typical Operating Circuit

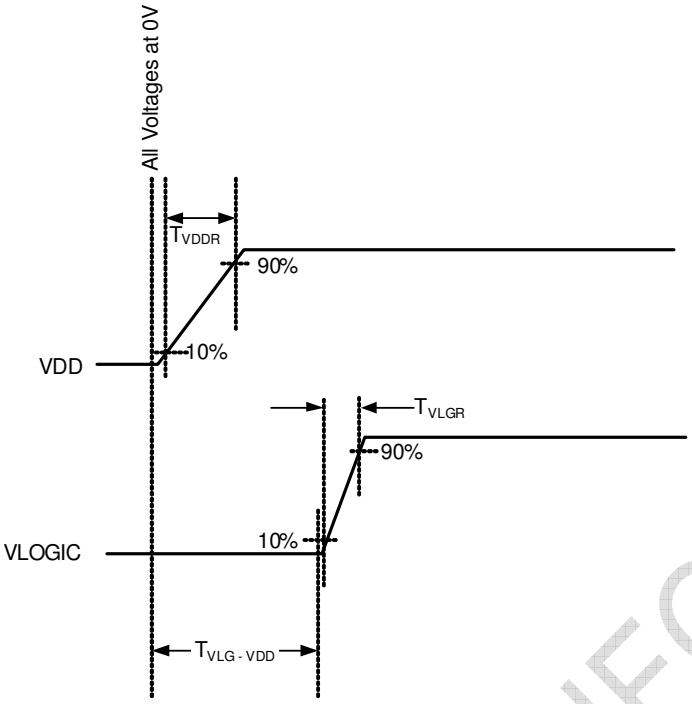


Typical Operating Circuits

7.3 Bill of Materials for External Components

Component	Label	Specification	Quantity
Regulator Filter Capacitor (Pin 10)	C1	Ceramic, X7R, 0.1µF ±10%, 2V	1
VDD Bypass Capacitor (Pin 13)	C2	Ceramic, X7R, 0.1µF ±10%, 4V	1
Charge Pump Capacitor (Pin 20)	C3	Ceramic, X7R, 2.2nF ±10%, 50V	1
VLOGIC Bypass Capacitor (Pin 8)	C4*	Ceramic, X7R, 10nF ±10%, 4V	1

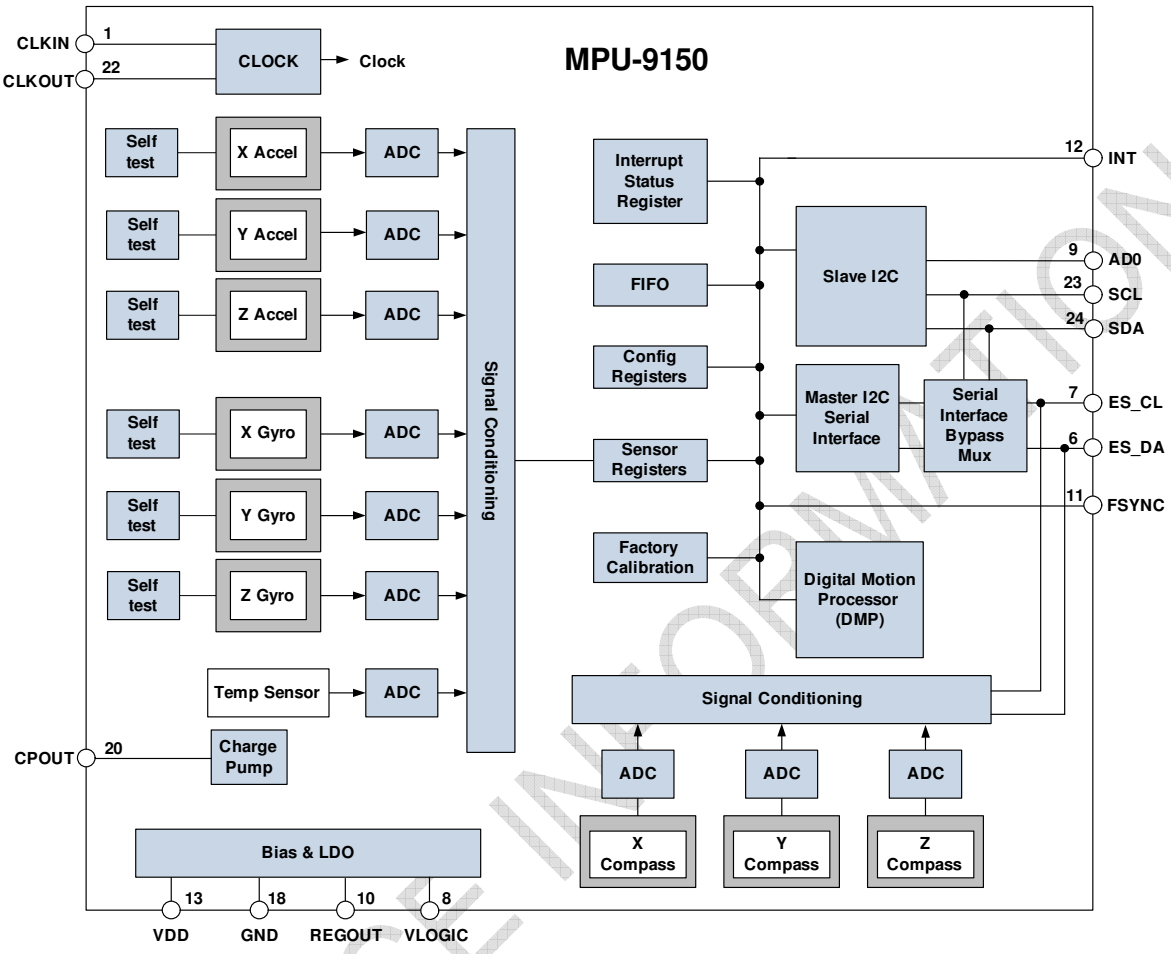
7.4 Recommended Power-on Procedure



Power-Up Sequencing

1. VLOGIC amplitude must always be ≤VDD amplitude
2. T_{VDDR} is VDD rise time: Time for VDD to rise from 10% to 90% of its final value
3. T_{VDDR} is ≤100msec
4. T_{VLGR} is VLOGIC rise time: Time for VLOGIC to rise from 10% to 90% of its final value
5. T_{VLGR} is ≤3msec
6. $T_{VLG-VDD}$ is the delay from the start of VDD ramp to the start of VLOGIC rise
7. $T_{VLG-VDD}$ is ≥0ms;
8. VDD and VLOGIC must be monotonic ramps

7.5 Block Diagram



7.6 Overview

The MPU-9150 is comprised of the following key blocks and functions:

- Three-axis MEMS rate gyroscope sensor with 16-bit ADCs and signal conditioning
- Three-axis MEMS accelerometer sensor with 16-bit ADCs and signal conditioning
- Three-axis MEMS magnetometer sensor with 13-bit ADCs and signal conditioning
- Digital Motion Processor (DMP) engine
- Primary I^2C serial communications interface
- Auxiliary I^2C serial interface for 3rd party sensors
- Clocking
- Sensor Data Registers
- FIFO
- Interrupts
- Digital-Output Temperature Sensor
- Gyroscope, Accelerometer and Magnetometer Self-test
- Bias and LDO
- Charge Pump

7.7 Three-Axis MEMS Gyroscope with 16-bit ADCs and Signal Conditioning

The MPU-9150 includes a 3-Axis vibratory MEMS rate gyroscope, which detect rotations about the X-, Y-, and Z- Axes. When the gyro is are rotated about any of the sense axes, the Coriolis Effect causes a vibration that is detected by a capacitive pickoff. The resulting signal is amplified, demodulated, and filtered to produce a voltage that is proportional to the angular rate. This voltage is digitized using individual on-chip 16-bit Analog-to-Digital Converters (ADCs) to sample each axis. The full-scale range of the gyro sensor may be digitally programmed to ±250, ±500, ±1000, or ±2000 degrees per second (dps). The ADC sample rate is programmable from 8,000 samples per second, down to 3.9 samples per second, and user-selectable low-pass filters enable a wide range of cut-off frequencies.

7.8 Three-Axis MEMS Accelerometer with 16-bit ADCs and Signal Conditioning

The MPU-9150's 3-axis accelerometer uses separate proof masses for each axis. Acceleration along a particular axis induces displacement on the corresponding proof mass, and capacitive sensors detect the displacement differentially. The MPU-9150's architecture reduces the accelerometer's susceptibility to fabrication variations as well as to thermal drift. When the device is placed on a flat surface, it will measure 0g on the X- and Y-axes and +1g on the Z-axis. The accelerometer's scale factor is calibrated at the factory and is nominally independent of supply voltage. Each sensor has a dedicated sigma-delta ADC for providing digital outputs. The full scale range of the digital output can be adjusted to ±2*g*, ±4*g*, ±8*g*, or ±16*g*.

7.9 Three-Axis MEMS Magnetometer with 13-bit ADCs and Signal Conditioning

The 3-axis magnetometer uses highly sensitive Hall sensor technology. The compass portion of the IC incorporates magnetic sensors for detecting terrestrial magnetism in the X-, Y-, and Z- Axes, a sensor driving circuit, a signal amplifier chain, and an arithmetic circuit for processing the signal from each sensor. Each ADC has a 13-bit resolution and a full scale range of ±1200 µT.

7.10 Digital Motion Processor

The embedded Digital Motion Processor (DMP) is located within the MPU-9150 and offloads computation of motion processing algorithms from the host processor. The DMP acquires data from accelerometers, gyroscopes, magnetometers and additional 3rd party sensors such as pressure sensors, and processes the data. The resulting data can be read from the DMP's registers, or can be buffered in a FIFO. The DMP has access to one of the MPU's external pins, which can be used for generating interrupts.

The purpose of the DMP is to offload both timing requirements and processing power from the host processor. Typically, motion processing algorithms should be run at a high rate, often around 200Hz, in order to provide accurate results with low latency. This is required even if the application updates at a much lower rate; for example, a low power user interface may update as slowly as 5Hz, but the motion processing should still run at 200Hz. The DMP can be used as a tool in order to minimize power, simplify timing, simplify the software architecture, and save valuable MIPS on the host processor for use in the application.

7.11 Primary I^2C

The MPU-9150 communicates to a system processor using an I^2C serial interface. The MPU-9150 always acts as a slave when communicating to the system processor. The logic level for communications to the master is set by the voltage on the VLOGIC pin. The LSB of the of the I^2C slave address is set by pin 9 (AD0).

7.12 Auxiliary I²C Serial Interface

The MPU-9150 has an auxiliary I²C bus for communicating to off-chip sensors. This bus has two operating modes:

- I²C Master Mode: The MPU-9150 acts as a master to any external sensors connected to the auxiliary I²C bus
- Pass-Through Mode: The MPU-9150 directly connects the primary and auxiliary I²C buses together, allowing the system processor to directly communicate with any external sensors.

Auxiliary I²C Bus Modes of Operation:

- I²C Master Mode: Allows the MPU-9150 to directly access the data registers of external digital sensors, such as a pressure sensor. In this mode, the MPU-9150 directly obtains data from auxiliary sensors, allowing the on-chip DMP to generate sensor fusion data without intervention from the system applications processor.

  For example, In I²C Master mode, the MPU-9150 can be configured to perform burst reads, returning the following data from a triple-Axis external sensor:

  - X-Axis data (2 bytes)
  - Y-Axis data (2 bytes)
  - Z-Axis data (2 bytes)

- The I²C Master can be configured to read up to 24 bytes from up to 3 auxiliary sensors. A fourth sensor can be configured to work single byte read/write mode.

- Pass-Through Mode: Allows an external system processor to act as master and directly communicate to the external sensors connected to the auxiliary I²C bus pins (ES_DA and ESCL). In this mode, the auxiliary I²C bus control logic (3rd-party sensor interface block) of the MPU-9150 is disabled, and the auxiliary I²C pins ES_DA and ES_CL (Pins 6 and 7) are connected to the main I²C bus (Pins 23 and 24) through analog switches.

  Pass-Through Mode is useful for configuring the external sensor, or for keeping the MPU-9150 in a low-power mode when only the external sensors are used. In Pass-Through Mode the system processor can still access MPU-9150 data through the I²C interface.

Auxiliary I²C Bus IO Logic Level

The logic level of the auxiliary I²C bus is VDD.

For further information regarding the MPU-9150's logic level, please refer to Section 10.2.

7.13 Self-Test

Please refer to the register map document for more details on self-test.

Self-test allows for the testing of the mechanical and electrical portions of the sensors. The self-test for each measurement axis can be activated by controlling the bits of the Gyro and Accel control registers.

When self-test is activated, the electronics cause the sensors to be actuated and produce an output signal. The output signal is used to observe the self-test response.

The self-test response is defined as follows:

Self-test response = Sensor output with self-test enabled – Sensor output without self-test enabled

The self-test response for each accelerometer axis is defined in the accelerometer specification table (Section 6.2), while that for each gyroscope axis is defined in the gyroscope specification table (Section 6.1).

When the value of the self-test response is within the min/max limits of the product specification, the part has passed self-test. When the self-test response exceeds the min/max values, the part is deemed to have failed self-test. Code for operating self-test code is included within the MotionApps software provided by InvenSense.

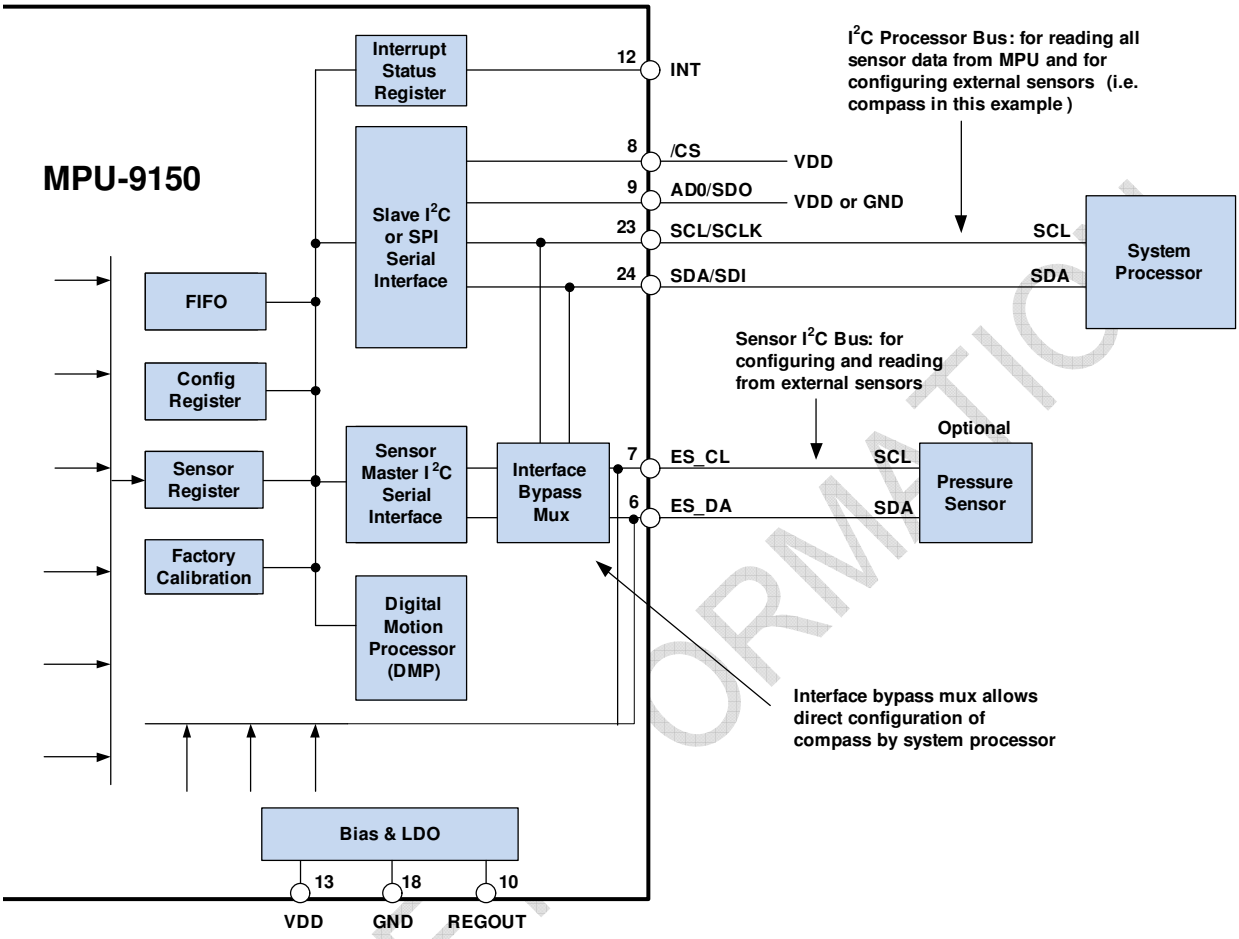
For Magnetometer self-test information please refer to "App Note – MPU-9150 Factory Self-Test for Magnetometer."

7.14 MPU-9150 Solution for 10-Axis Sensor Fusion Using I^2C Interface

In the figure below, the system processor is an I^2C master to the MPU-9150. In addition, the MPU-9150 is an I^2C master to the optional external pressure sensor. The MPU-9150 has limited capabilities as an I^2C Master, and depends on the system processor to manage the initial configuration of any auxiliary sensors. The MPU-9150 has an interface bypass multiplexer, which connects the system processor I^2C bus pins 23 and 24 (SDA and SCL) directly to the auxiliary sensor I^2C bus pins 6 and 7 (ES_DA and ES_CL).

Once the auxiliary sensors have been configured by the system processor, the interface bypass multiplexer should be disabled so that the MPU-9150 auxiliary I^2C master can take control of the sensor I^2C bus and gather data from the auxiliary sensors.

For further information regarding I^2C master control, please refer to Section 10.



7.15 Procedure for Directly Accessing the AK8975 3-Axis Compass

The AK8975 3-Axis Compass is connected to the MPU-9150 through the MPU's Auxiliary I^2C Bus. In order to access this compass directly, the MPU-9150 should be put into Pass-Through Mode.

For further information regarding MPU-9150 Pass-Through Mode, please refer to Section 7.12.

The slave address for AK8975 is 0x0C or 12 decimal.

The MPU-9150 pin configuration for Direct Access to the AK8975 is described in the table below.

Pin Configuration for Direct Access to AK8975 3-Axis Compass

Pin Number	Pin Name	Pin Description
1	CLKIN	Inactive. Connect to GND.
6	ES_DA	Active. Leave as NC. (provision for option external pull-up resistor to VDD)
7	ES_CL	Active. Leave as NC. (provision for option external pull-up resistor to VDD)
8	VLOGIC	Active. Digital I/O supply voltage.
9	AD0	Active. Connect to GND.
10	REGOUT	Active. Connect a 100nF bypass capacitor on the board.
11	FSYNC	Inactive. Connect to GND.
12	INT	Inactive. Leave as NC.
3, 13	VDD	Power supply voltage and Digital I/O supply voltage
15, 17,18	GND	Power supply ground.
20	CPOUT	Active. Connect a 10nF bypass capacitor on the board.
22	CLKOUT	Inactive. Leave as NC.
23	SCL	Active. I^2C serial clock (SCL)
24	SDA	Active. I^2C serial data (SDA)
2, 4, 5, 14, 16, 19, 21	RESV	Reserved. Do not connect.

For detailed information regarding the Register Map of the AK8975, please refer to the MPU-9150 Register Map and Register Descriptions document.

7.16 Internal Clock Generation

The MPU-9150 has a flexible clocking scheme, allowing a variety of internal or external clock sources to be used for the internal synchronous circuitry. This synchronous circuitry includes the signal conditioning and ADCs, the DMP, and various control circuits and registers. An on-chip PLL provides flexibility in the allowable inputs for generating this clock.

Allowable internal sources for generating the internal clock are:

- An internal relaxation oscillator
- Any of the X, Y, or Z gyros (MEMS oscillators with a variation of ±1% over temperature)

Allowable external clocking sources are:

- 32.768kHz square wave
- 19.2MHz square wave

Selection of the source for generating the internal synchronous clock depends on the availability of external sources and the requirements for power consumption and clock accuracy. These requirements will most likely vary by mode of operation. For example, in one mode, where the biggest concern is power consumption, the user may wish to operate the Digital Motion Processor of the MPU-9150 to process accelerometer data, while keeping the gyros and magnetometer off. In this case, the internal relaxation oscillator is a good clock choice. However, in another mode, where the gyros are active, selecting the gyros as the clock source provides for a more accurate clock source.

Clock accuracy is important, since timing errors directly affect the distance and angle calculations performed by the Digital Motion Processor (and by extension, by any processor).

There are also start-up conditions to consider. When the MPU-9150 first starts up, the device uses its internal clock until programmed to operate from another source. This allows the user, for example, to wait for the MEMS oscillators to stabilize before they are selected as the clock source.

7.17 Sensor Data Registers

The sensor data registers contain the latest gyro, accelerometer, magnetometer and temperature measurement data. They are read-only registers, and are accessed via the serial interface. Data from these registers may be read anytime. However, the interrupt function may be used to determine when new data is available.

For a table of interrupt sources please refer to Section 8.

7.18 FIFO

The MPU-9150 contains a 1024-byte FIFO register that is accessible via the Serial Interface. The FIFO configuration register determines which data is written into the FIFO. Possible choices include gyro data, accelerometer data, temperature readings, auxiliary sensor readings, and FSYNC input. A FIFO counter keeps track of how many bytes of valid data are contained in the FIFO. The FIFO register supports burst reads. The interrupt function may be used to determine when new data is available.

For further information regarding the FIFO, please refer to the MPU-9150 Register Map and Register Descriptions document.

7.19 Interrupts

Interrupt functionality is configured via the Interrupt Configuration register. Items that are configurable include the INT pin configuration, the interrupt latching and clearing method, and triggers for the interrupt. Items that can trigger an interrupt are (1) new data is available to be read (from the FIFO and Data registers); (2) accelerometer event interrupts; and (3) the MPU-9150 did not receive an acknowledge from an auxiliary sensor on the secondary I^2C bus. The interrupt status can be read from the Interrupt Status register.

For further information regarding interrupts, please refer to the MPU-9150 Register Map and Register Descriptions document.

For information regarding the MPU-9150's accelerometer event interrupts, please refer to Section 8.

7.20 Digital-Output Temperature Sensor

An on-chip temperature sensor and ADC are used to measure the MPU-9150 die temperature. The readings from the ADC can be read from the FIFO or the Sensor Data registers.

7.21 Bias and LDO

The bias and LDO section generates the internal supply and the reference voltages and currents required by the MPU-9150. Its two inputs are an unregulated VDD and a VLOGIC logic reference supply voltage. The LDO output is bypassed by a capacitor at REGOUT. For further details on the capacitor, please refer to the Bill of Materials for External Components (Section 7.3).

7.22 Charge Pump

An on-board charge pump generates the high voltage required for the MEMS oscillators. Its output is bypassed by a capacitor at CPOUT. For further details on the capacitor, please refer to the Bill of Materials for External Components (Section 7.3).

8 Programmable Interrupts

The MPU-9150 has a programmable interrupt system which can generate an interrupt signal on the INT pin. Status flags indicate the source of an interrupt. Interrupt sources may be enabled and disabled individually.

Table of Interrupt Sources

Interrupt Name	Module
Motion Detection	Motion
FIFO Overflow	FIFO
Data Ready	Sensor Registers
I^2C Master errors: Lost Arbitration, NACKs	I^2C Master
I^2C Slave 4	I^2C Master

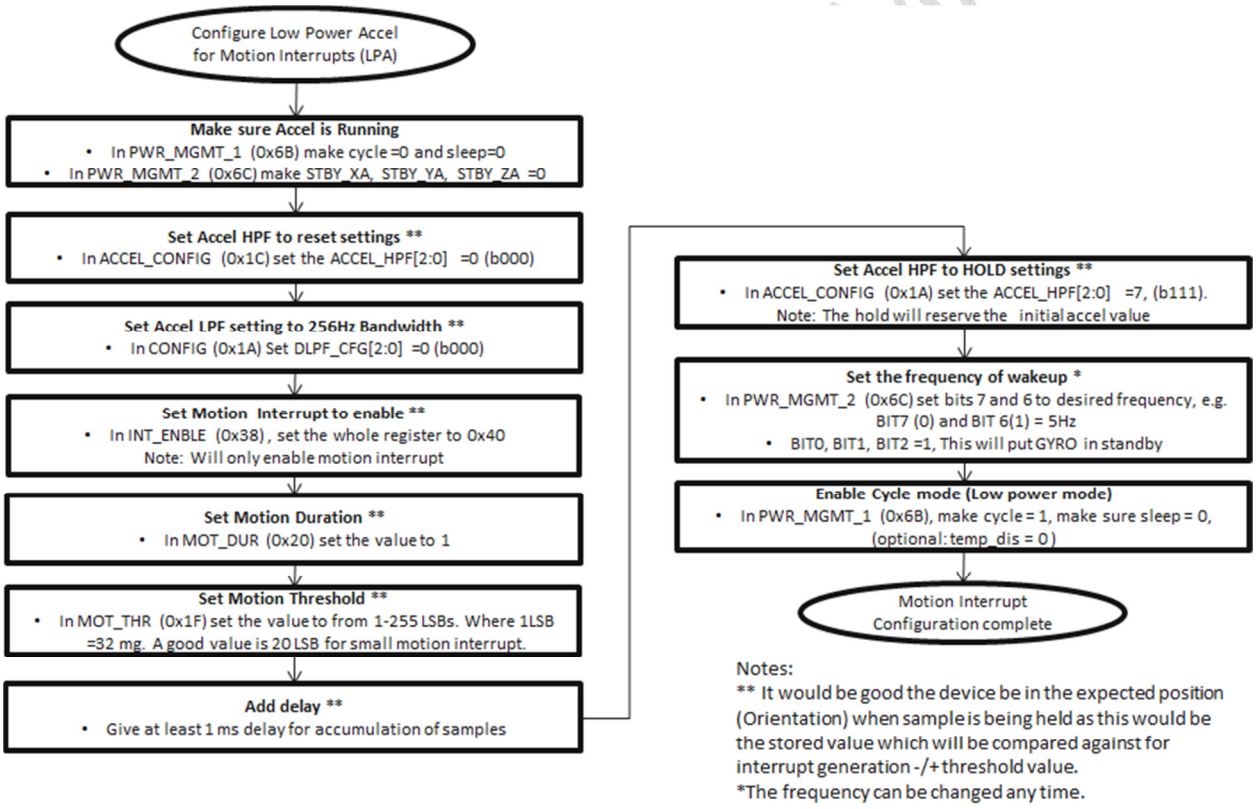
For information regarding the interrupt enable/disable registers and flag registers, please refer to the MPU-9150 Register Map and Register Descriptions document. Some interrupt sources are explained below.

8.1 Motion Interrupt

The MPU-9150 provides Motion detection capability. Accelerometer measurements are passed through a configurable digital high pass filter (DHPF) in order to eliminate bias due to gravity. A qualifying motion sample is one where the high passed sample from any axis has an absolute value exceeding a user-programmable threshold. A counter increments for each qualifying sample, and decrements for each non-qualifying sample. Once the counter reaches a user-programmable counter threshold, a motion interrupt is triggered. The axis and polarity which caused the interrupt to be triggered is flagged in the MOT_DETECT_STATUS register.

Motion detection has a configurable acceleration threshold MOT_THR specified in 1 m*g* increments. The counter threshold MOT_DUR is specified in 1 ms increments. The decrement rate for non-qualifying samples is also configurable. The MOT_DETECT_CTRL register allows the user to specify whether a non-qualifying sample makes the counter reset to zero, or decrement in steps of 1, 2, or 4.

The flow chart below explains how the motion interrupt should be used. Please refer to the MPU-9150 Register Map and Register Descriptions document for descriptions of the registers referenced in the flow chart.



9 Digital Interface

9.1 I²C Serial Interface

The internal registers and memory of the MPU-9150 can be accessed using either I²C at 400 kHz.

Serial Interface

Pin Number	Pin Name	Pin Description
8	VLOGIC	Digital I/O supply voltage. VLOGIC must be ≤ VDD at all times.
9	AD0	I²C Slave Address LSB
23	SCL	I²C serial clock
24	SDA	I²C serial data

9.2 I²C Interface

I²C is a two-wire interface comprised of the signals serial data (SDA) and serial clock (SCL). In general, the lines are open-drain and bi-directional. In a generalized I²C interface implementation, attached devices can be a master or a slave. The master device puts the slave address on the bus, and the slave device with the matching address acknowledges the master.

The MPU-9150 always operates as a slave device when communicating to the system processor, which thus acts as the master. SDA and SCL lines typically need pull-up resistors to VDD. The maximum bus speed is 400 kHz.

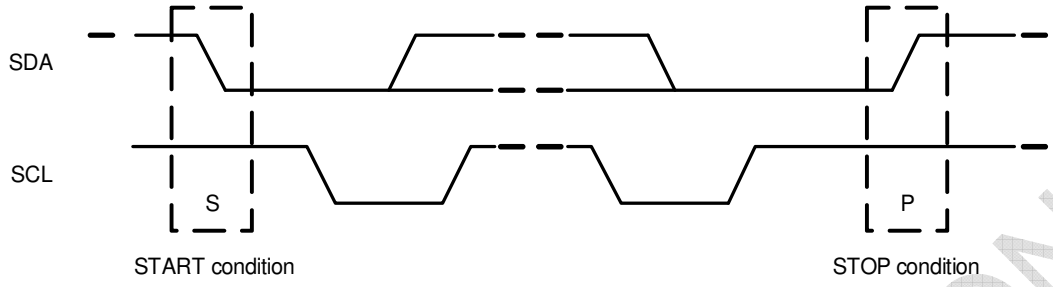
The slave address of the MPU-9150 is b110100X which is 7 bits long. The LSB bit of the 7 bit address is determined by the logic level on pin AD0. This allows two MPU-9150s to be connected to the same I²C bus. When used in this configuration, the address of the one of the devices should be b1101000 (pin AD0 is logic low) and the address of the other should be b1101001 (pin AD0 is logic high).

9.3 I²C Communications Protocol

START (S) and STOP (P) Conditions

Communication on the I²C bus starts when the master puts the START condition (S) on the bus, which is defined as a HIGH-to-LOW transition of the SDA line while SCL line is HIGH (see figure below). The bus is considered to be busy until the master puts a STOP condition (P) on the bus, which is defined as a LOW to HIGH transition on the SDA line while SCL is HIGH (see figure below).

Additionally, the bus remains busy if a repeated START (Sr) is generated instead of a STOP condition.

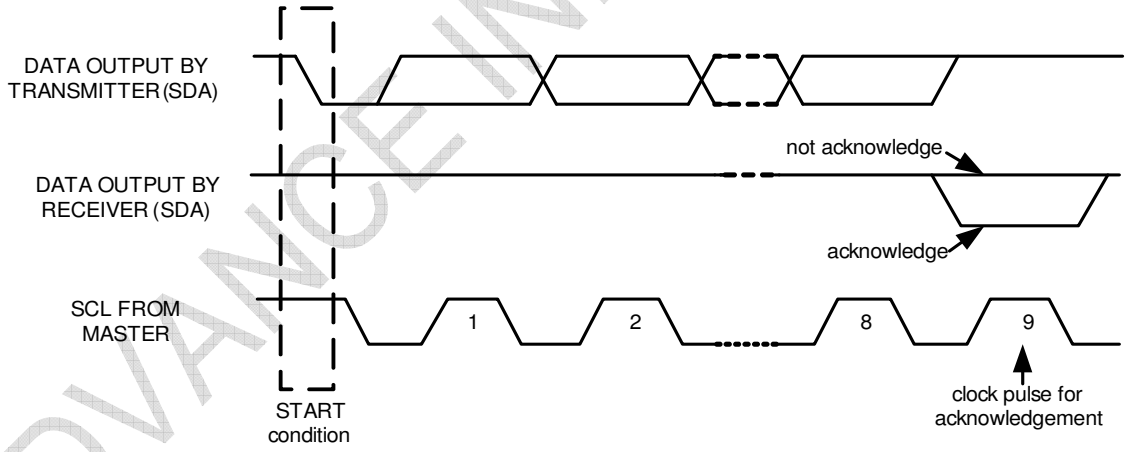


START and STOP Conditions

Data Format / Acknowledge

I^2C data bytes are defined to be 8-bits long. There is no restriction to the number of bytes transmitted per data transfer. Each byte transferred must be followed by an acknowledge (ACK) signal. The clock for the acknowledge signal is generated by the master, while the receiver generates the actual acknowledge signal by pulling down SDA and holding it low during the HIGH portion of the acknowledge clock pulse.

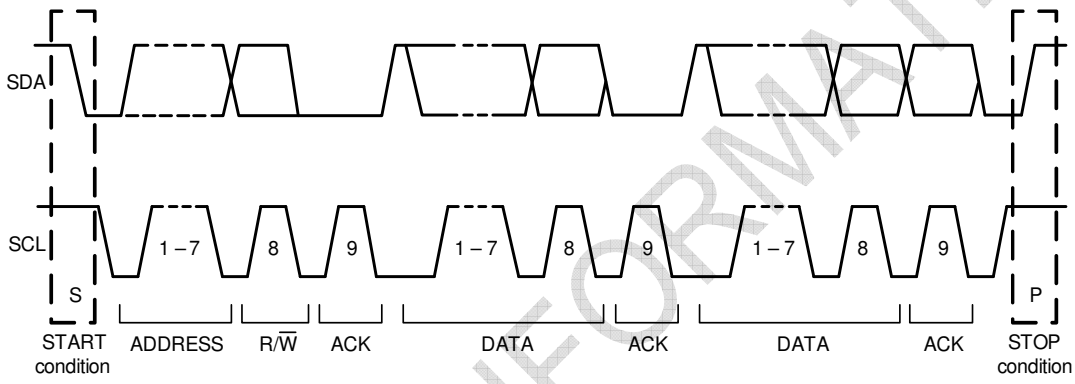
If a slave is busy and cannot transmit or receive another byte of data until some other task has been performed, it can hold SCL LOW, thus forcing the master into a wait state. Normal data transfer resumes when the slave is ready, and releases the clock line (refer to the following figure).



Acknowledge on the I^2C Bus

Communications

After beginning communications with the START condition (S), the master sends a 7-bit slave address followed by an 8th bit, the read/write bit. The read/write bit indicates whether the master is receiving data from or is writing to the slave device. Then, the master releases the SDA line and waits for the acknowledge signal (ACK) from the slave device. Each byte transferred must be followed by an acknowledge bit. To acknowledge, the slave device pulls the SDA line LOW and keeps it LOW for the high period of the SCL line. Data transmission is always terminated by the master with a STOP condition (P), thus freeing the communications line. However, the master can generate a repeated START condition (Sr), and address another slave without first generating a STOP condition (P). A LOW to HIGH transition on the SDA line while SCL is HIGH defines the stop condition. All SDA changes should take place when SCL is low, with the exception of start and stop conditions.



Complete I²C Data Transfer

To write the internal MPU-9150 registers, the master transmits the start condition (S), followed by the I²C address and the write bit (0). At the 9th clock cycle (when the clock is high), the MPU-9150 acknowledges the transfer. Then the master puts the register address (RA) on the bus. After the MPU-9150 acknowledges the reception of the register address, the master puts the register data onto the bus. This is followed by the ACK signal, and data transfer may be concluded by the stop condition (P). To write multiple bytes after the last ACK signal, the master can continue outputting data rather than transmitting a stop signal. In this case, the MPU-9150 automatically increments the register address and loads the data to the appropriate register. The following figures show single and two-byte write sequences.

Single-Byte Write Sequence

Master	S	AD+W		RA		DATA		P
Slave			ACK		ACK		ACK	

Burst Write Sequence

Master	S	AD+W		RA		DATA		DATA		P
Slave			ACK		ACK		ACK		ACK	

To read the internal MPU-9150 registers, the master sends a start condition, followed by the I^2C address and a write bit, and then the register address that is going to be read. Upon receiving the ACK signal from the MPU-9150, the master transmits a start signal followed by the slave address and read bit. As a result, the MPU-9150 sends an ACK signal and the data. The communication ends with a not acknowledge (NACK) signal and a stop bit from master. The NACK condition is defined such that the SDA line remains high at the 9^{th} clock cycle. The following figures show single and two-byte read sequences.

Single-Byte Read Sequence

Master	S	AD+W		RA		S	AD+R			NACK	P
Slave			ACK		ACK			ACK	DATA		

Burst Read Sequence

Master	S	AD+W		RA		S	AD+R			ACK		NACK	P
Slave			ACK		ACK			ACK	DATA		DATA		

9.4 I^2C Terms

Signal	Description
S	Start Condition: SDA goes from high to low while SCL is high
AD	Slave I^2C address
W	Write bit (0)
R	Read bit (1)
ACK	Acknowledge: SDA line is low while the SCL line is high at the 9^{th} clock cycle
NACK	Not-Acknowledge: SDA line stays high at the 9^{th} clock cycle
RA	MPU-9150 internal register address
DATA	Transmit or received data
P	Stop condition: SDA going from low to high while SCL is high

10 Serial Interface Considerations

10.1 MPU-9150 Supported Interfaces

The MPU-9150 supports I^2C communications.

10.2 Logic Levels

The MPU-9150's I/O logic levels are set to be either VDD or VLOGIC, as shown in the table below.

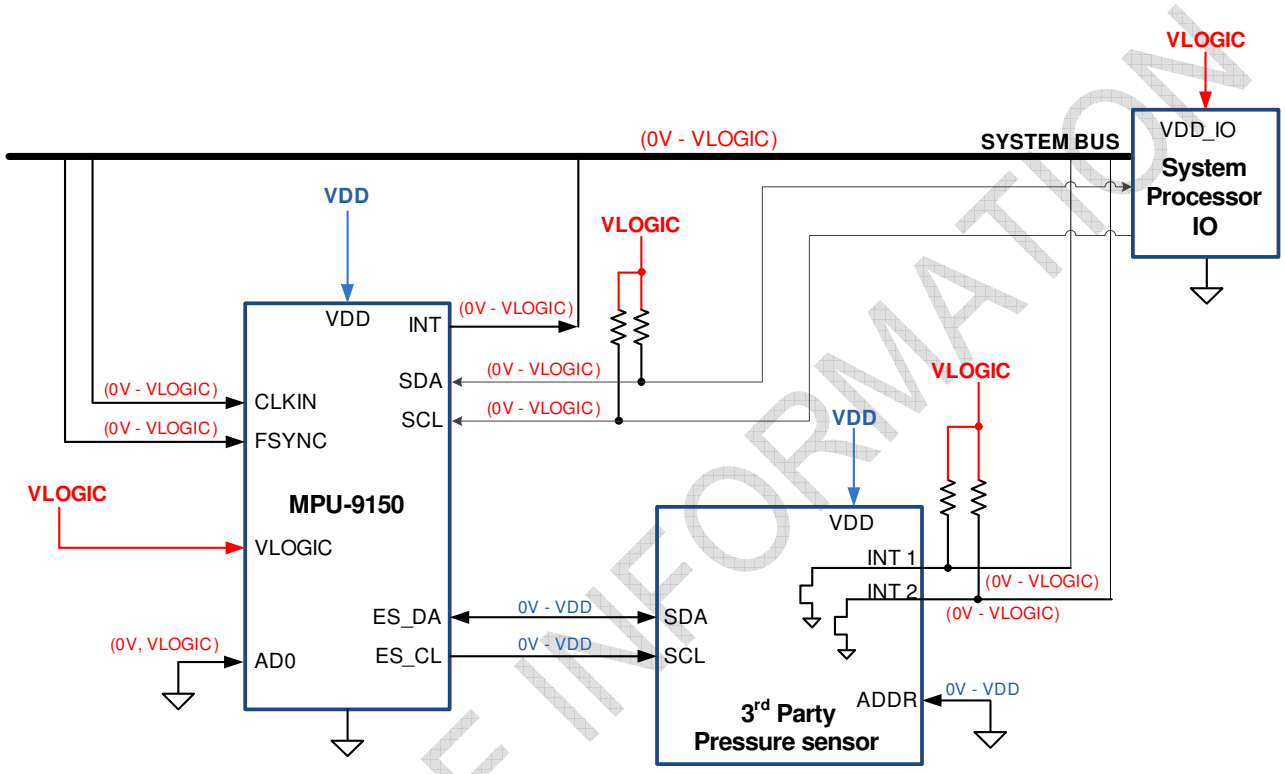
I/O Logic Levels

MICROPROCESSOR LOGIC LEVELS (Pins: SDA, SCL, AD0, CLKIN, INT)	**AUXILIARY LOGIC LEVELS** (Pins: ES_DA, ES_CL)
VLOGIC	VDD

VLOGIC may be set to be equal to VDD or to another voltage. However, VLOGIC must be ≤ VDD at all times. VLOGIC is the power supply voltage for the microprocessor system bus and VDD is the supply for the auxiliary I^2C bus, as shown in the figure of Section 10.3.

10.3 Logic Levels Diagram

The figure below depicts a sample circuit with a third party pressure sensor attached to the auxiliary I^2C bus. It shows logic levels and voltage connections. Note: Actual configuration will depend on the auxiliary sensors used.



I/O Levels and Connections

Notes:

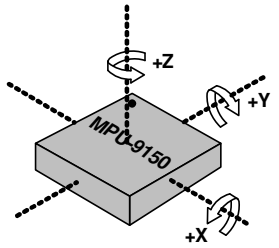
1. The IO voltage levels of ES_DA and ES_CL are set relative to VDD.
2. Third-party auxiliary device logic levels are referenced to VDD. Setting INT1 and INT2 to open drain configuration provides voltage compatibility when VDD ≠ VLOGIC. When VDD = VLOGIC, INT1 and INT2 may be set to push-pull outputs, and external pull-up resistors are not needed.
3. CLKOUT is referenced to VDD.
4. All other MPU-9150 logic IO is always referenced to VLOGIC.

11 Assembly

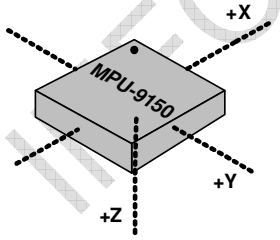
This section provides general guidelines for assembling InvenSense Micro Electro-Mechanical Systems (MEMS) gyros packaged in Lead Grid Array package (LGA) surface mount integrated circuits.

11.1 Orientation of Axes

The diagram below shows the orientation of the axes of sensitivity and the polarity of rotation. Note the pin 1 identifier (•) in the figure.

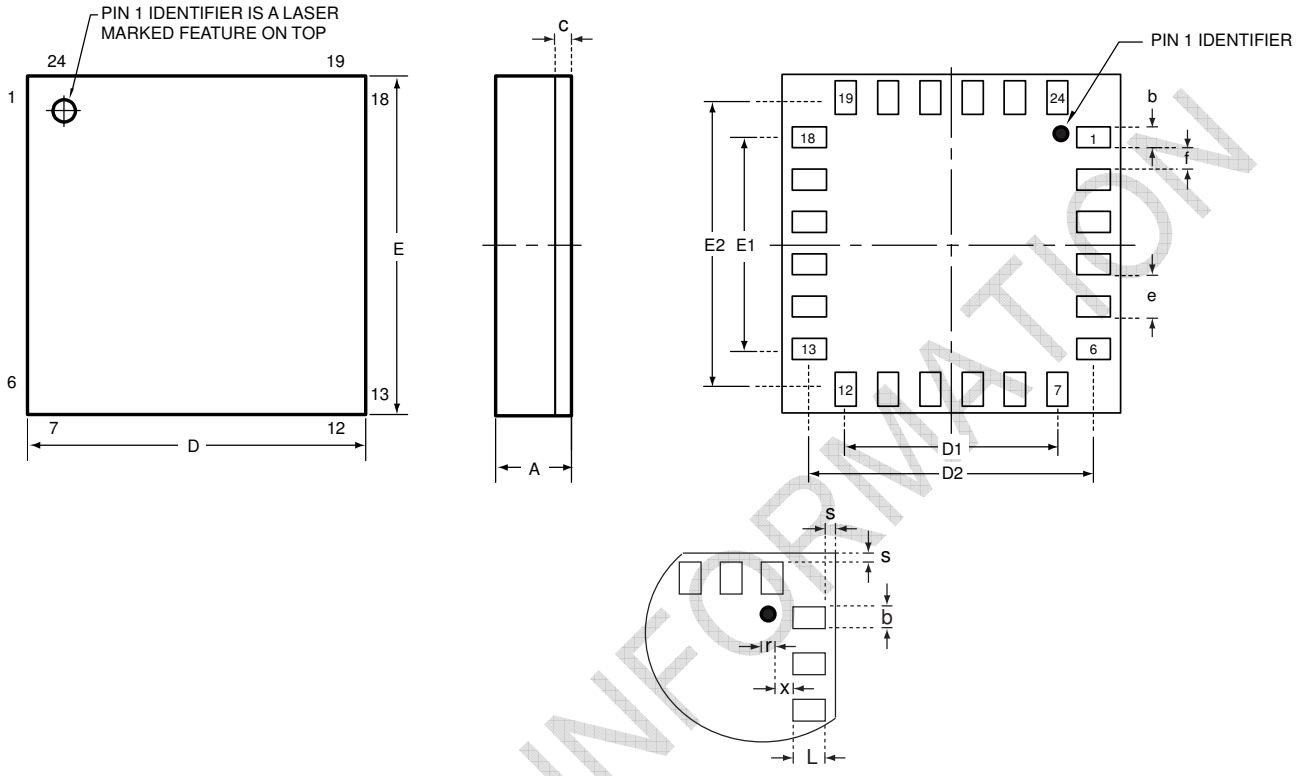


Orientation of Axes of Sensitivity and Polarity of Rotation for Gyroscopes and Accelerometers



Orientation of Axes of Sensitivity for Compass

11.2 Package Dimensions

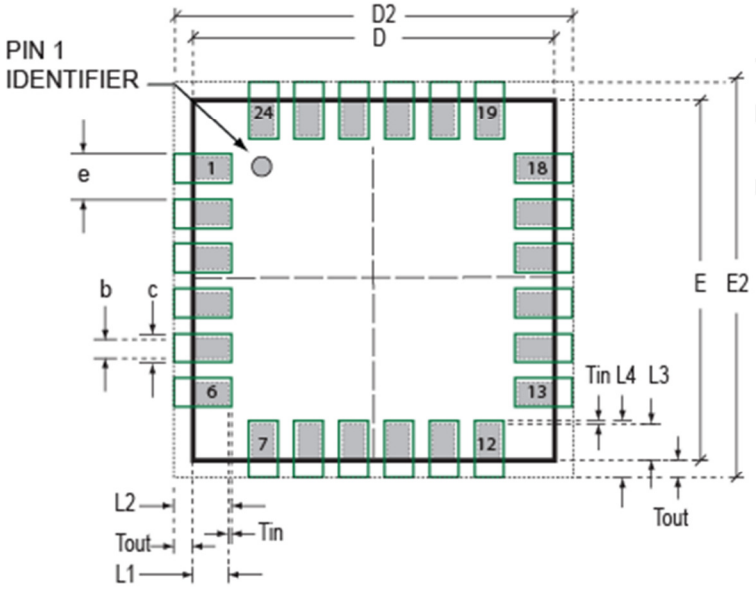


SYMBOLS	DIMENSIONS IN MILLIMETERS		
	MIN.	NOM.	MAX.
A	0.90	1.00	1.10
c	0.106	0.136	0.166
D	3.90	4.00	4.10
E	3.90	4.00	4.10
D1/E1	---	2.50	---
D2/E2	---	3.41	---
e	---	0.50	---
b	0.22	0.25	0.28
f	0.22	0.25	0.28
L	0.32	0.35	0.38
s	---	0.12	---
x	0.32	0.35	0.38

11.3 PCB Design Guidelines:

The Pad Diagram using a JEDEC type extension with solder rising on the outer edge is shown below. The Pad Dimensions Table shows pad sizing (mean dimensions) recommended for the MPU-9150 product.

JEDEC type extension with solder rising on outer edge



PCB Lay-out Diagram

SYMBOLS	DIMENSIONS IN MILLIMETERS	NOM
	Nominal Package I/O Pad Dimensions	
e	Pad Pitch	0.50
b	Pad Width	0.25
L1	Pad Length	0.35
L3	Pad Length	0.40
D	Package Width	4.00
E	Package Length	4.00
	I/O Land Design Dimensions (Guidelines)	
D2	I/O Pad Extent Width	4.80
E2	I/O Pad Extent Length	4.80
c	Land Width	0.35
Tout	Outward Extension	0.40
Tin	Inward Extension	0.05
L2	Land Length	0.80
L4	Land Length	0.85

PCB Dimensions Table (for PCB Lay-out Diagram)

11.4 Assembly Precautions

11.4.1 Surface Mount Guidelines

InvenSense MEMS motion sensors are sensitive to mechanical stress coming from the printed circuit board (PCB). This PCB stress can be minimized by adhering to certain design rules.

When using MEMS components in plastic packages, PCB mounting and assembly can cause package stress. This package stress in turn can affect the output offset and its value over a wide range of temperatures. This stress is caused by the mismatch between the Coefficient of Linear Thermal Expansion (CTE) of the package material and the PCB. Care must be taken to avoid package stress due to mounting.

Traces connected to pads should be as symmetric as possible. Maximizing symmetry and balance for pad connection will help component self alignment and will lead to better control of solder paste reduction after reflow.

Any material used in the surface mount assembly process of the MEMS product should be free of restricted RoHS elements or compounds. Pb-free solders should be used for assembly.

11.4.2 Exposed Die Pad Precautions

The MPU-9150 has very low active and standby current consumption. The exposed center die pad is not required for heat sinking, and should not be soldered to the PCB. Under-fill should also not be used. Failure to adhere to this rule can induce performance changes due to package thermo-mechanical stress. There is no electrical connection between the pad and the CMOS.

11.4.3 Trace Routing

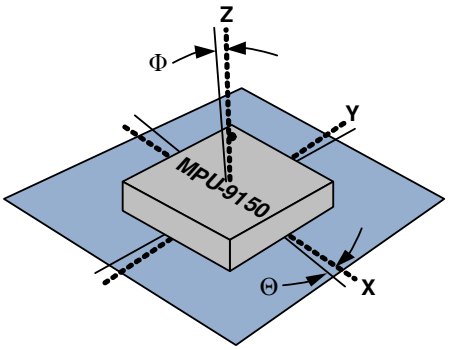
Routing traces or vias under the gyro package such that they run under the exposed die pad is prohibited. Routed active signals may harmonically couple with the gyro MEMS devices, compromising gyro response. These devices are designed with the drive frequencies as follows: X = 33±3kHz, Y = 30±3kHz, and Z=27±3kHz. To avoid harmonic coupling don't route active signals in non-shielded signal planes directly below, or above the gyro package. Note: For best performance, design a ground plane under the e-pad to reduce PCB signal noise from the board on which the gyro device is mounted. If the gyro device is stacked under an adjacent PCB board, design a ground plane directly above the gyro device to shield active signals from the adjacent PCB board.

11.4.4 Component Placement

Do not place large insertion components such as keyboard or similar buttons, connectors, or shielding boxes at a distance of less than 6 mm from the MEMS gyro. Maintain generally accepted industry design practices for component placement near the MPU-9150 to prevent noise coupling and thermo-mechanical stress.

11.4.5 PCB Mounting and Cross-Axis Sensitivity

Orientation errors of the gyroscope and accelerometer mounted to the printed circuit board can cause cross-axis sensitivity in which one gyro or accel responds to rotation or acceleration about another axis, respectively. For example, the X-axis gyroscope may respond to rotation about the Y or Z axes. The orientation mounting errors are illustrated in the figure below.



Package Gyro & Accel Axes (----------) Relative to PCB Axes (——) with Orientation Errors (Θ and Φ)

The table below shows the cross-axis sensitivity as a percentage of the specified gyroscope or accelerometer's sensitivity for a given orientation error, respectively.

Cross-Axis Sensitivity vs. Orientation Error

Orientation Error (θ or Φ)	Cross-Axis Sensitivity (sinθ or sinΦ)
0º	0%
0.5º	0.87%
1º	1.75%

The specifications for cross-axis sensitivity in Section 6.1 and Section 6.2 include the effect of the die orientation error with respect to the package.

11.4.6 MEMS Handling Instructions

MEMS (Micro Electro-Mechanical Systems) are a time-proven, robust technology used in hundreds of millions of consumer, automotive and industrial products. MEMS devices consist of microscopic moving mechanical structures. They differ from conventional IC products, even though they can be found in similar packages. Therefore, MEMS devices require different handling precautions than conventional ICs prior to mounting onto printed circuit boards (PCBs).

The MPU-9150 has been qualified to a shock tolerance of 10,000*g*. InvenSense packages its gyroscopes as it deems proper for protection against normal handling and shipping. It recommends the following handling precautions to prevent potential damage.

- Do not drop individually packaged gyroscopes, or trays of gyroscopes onto hard surfaces. Components placed in trays could be subject to *g*-forces in excess of 10,000*g* if dropped.
- Printed circuit boards that incorporate mounted gyroscopes should not be separated by manually snapping apart. This could also create *g*-forces in excess of 10,000*g.*

- Do not clean MEMS gyroscopes in ultrasonic baths. Ultrasonic baths can induce MEMS damage if the bath energy causes excessive drive motion through resonant frequency coupling.

11.4.7 ESD Considerations

Establish and use ESD-safe handling precautions when unpacking and handling ESD-sensitive devices.

- Store ESD sensitive devices in ESD safe containers until ready for use. The Tape-and-Reel moisture-sealed bag is an ESD approved barrier. The best practice is to keep the units in the original moisture sealed bags until ready for assembly.

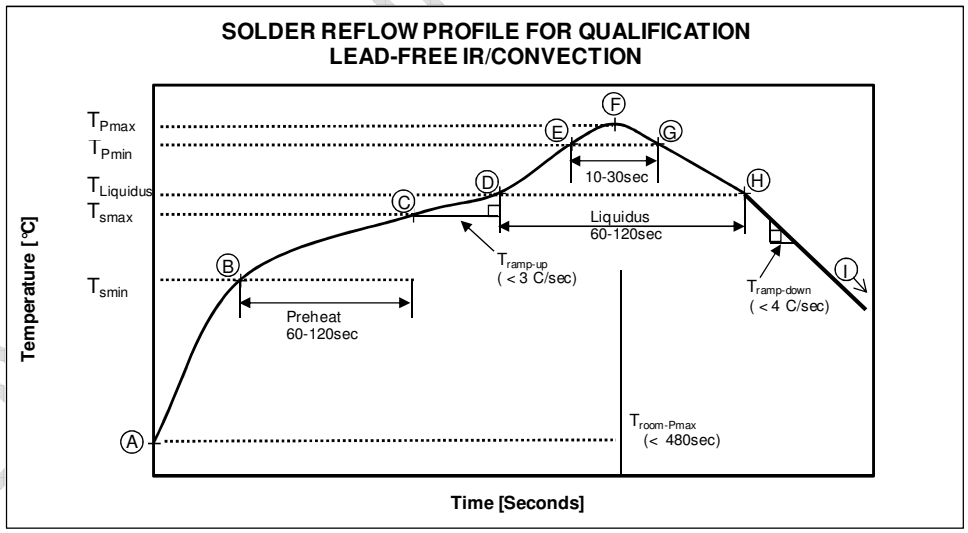
Restrict all device handling to ESD protected work areas that measure less than 200V static charge. Ensure that all workstations and personnel are properly grounded to prevent ESD.

11.5 Reflow Specification

Qualification Reflow: The MPU-9150 was qualified in accordance with IPC/JEDEC J-STD-020D.01. This standard classifies proper packaging, storage and handling in order to avoid subsequent thermal and mechanical damage during the solder reflow attachment phase of PCB assembly.

The qualification preconditioning process specifies a sequence consisting of a bake cycle, a moisture soak cycle (in a temperature humidity oven), and three consecutive solder reflow cycles, followed by functional device testing.

The peak solder reflow classification temperature requirement for package qualification is (260 +5/-0°C) for lead-free soldering of components measuring less than 1.6 mm in thickness. The qualification profile and a table explaining the set-points are shown below:



Temperature Set Points Corresponding to Reflow Profile Above

Step	Setting	CONSTRAINTS		
		Temp (°C)	Time (sec)	Max. Rate (°C/sec)
A	T_{room}	25		
B	T_{Smin}	150		
C	T_{Smax}	200	$60 < t_{BC} < 120$	
D	$T_{Liquidus}$	217		$r_{(TLiquidus-TPmax)} < 3$
E	$T_{Pmin\ [255°C,\ 260°C]}$	255		$r_{(TLiquidus-TPmax)} < 3$
F	$T_{Pmax\ [260°C,\ 265°C]}$	260	$t_{AF} < 480$	$r_{(TLiquidus-TPmax)} < 3$
G	$T_{Pmin\ [255°C,\ 260°C]}$	255	$10 < t_{EG} < 30$	$r_{(TPmax-TLiquidus)} < 4$
H	$T_{Liquidus}$	217	$60 < t_{DH} < 120$	
I	T_{room}	25		

Notes: Customers must never exceed the Classification temperature (T_{Pmax} = 260°C).
All temperatures refer to the topside of the QFN package, as measured on the package body surface.

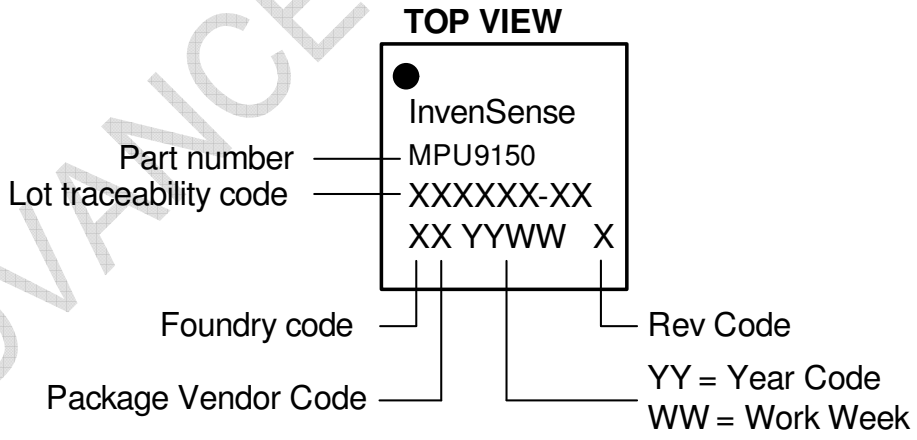
Production Reflow: Check the recommendations of your solder manufacturer. For optimum results, use lead-free solders that have lower specified temperature profiles (Tp_{max} ~ 235°C). Also use lower ramp-up and ramp-down rates than those used in the qualification profile. Never exceed the maximum conditions that we used for qualification, as these represent the maximum tolerable ratings for the device.

11.6 Storage Specifications

The storage specification of the MPU-9150 conforms to IPC/JEDEC J-STD-020D.01 Moisture Sensitivity Level (MSL) 3.

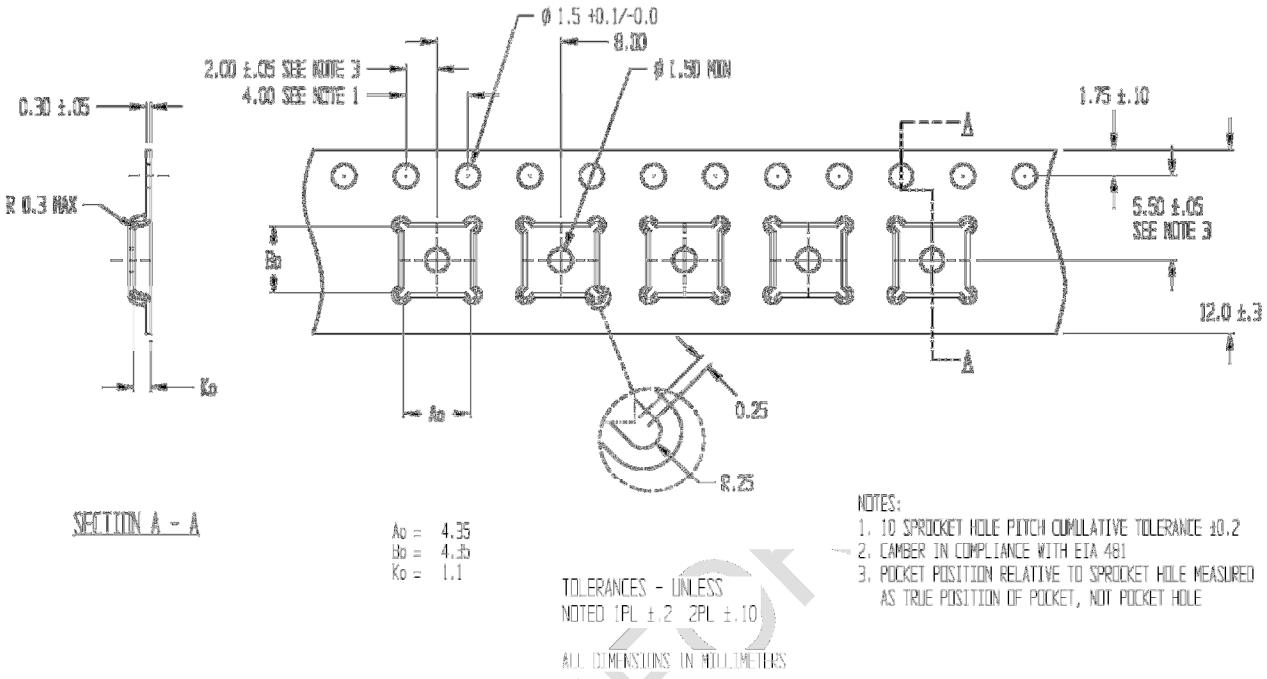
Calculated shelf-life in moisture-sealed bag	12 months -- Storage conditions: <40°C and <90% RH
After opening moisture-sealed bag	168 hours -- Storage conditions: ambient ≤30°C at 60%RH

11.7 Package Marking Specification

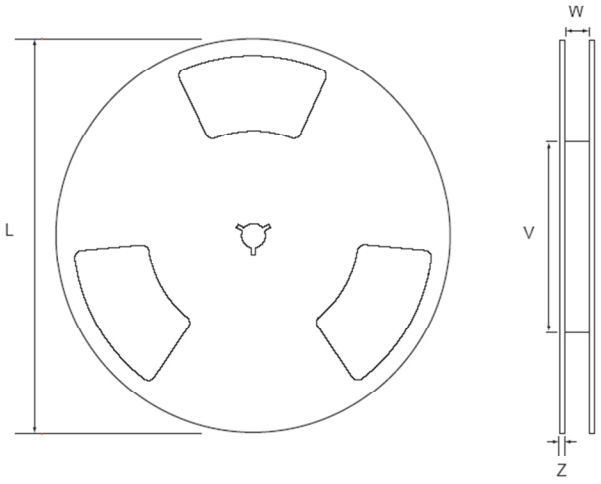


Package Marking Specification

11.8 Tape & Reel Specification



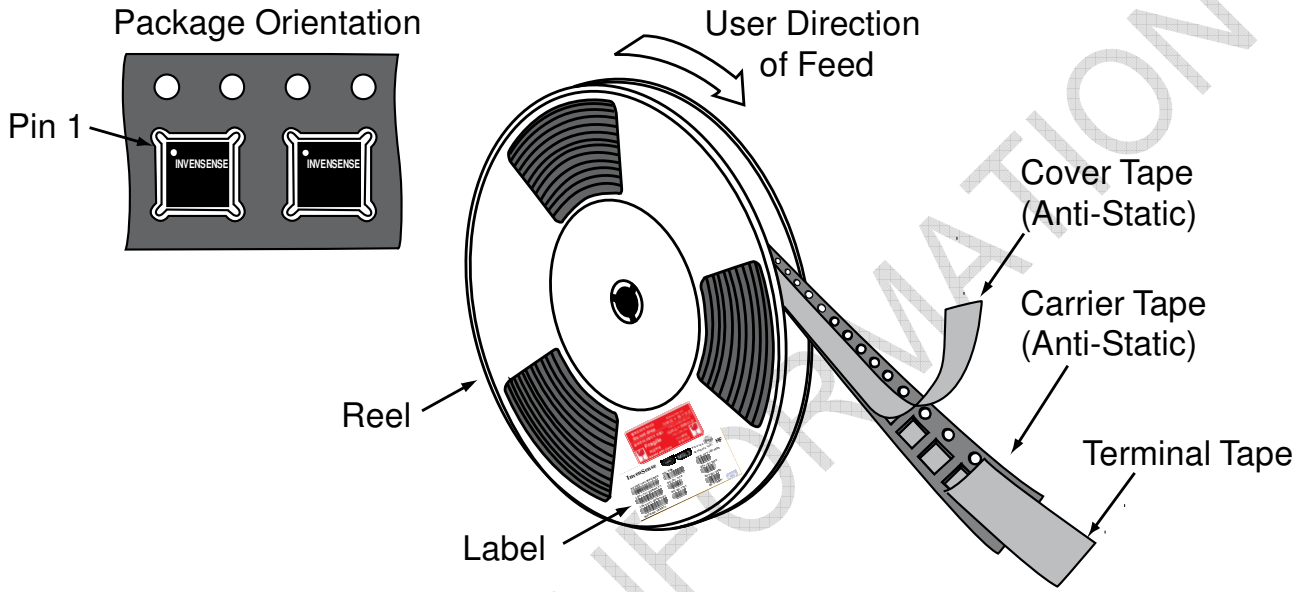
Tape Dimensions



Reel Outline Drawing

Reel Dimensions and Package Size

PACKAGE SIZE	REEL (mm)			
	L	V	W	Z
4x4	330	100	13.2	2.2



Tape and Reel Specification

Reel Specifications

Quantity Per Reel	5,000
Reels per Box	1
Boxes Per Carton (max)	5
Pcs/Carton (max)	25,000

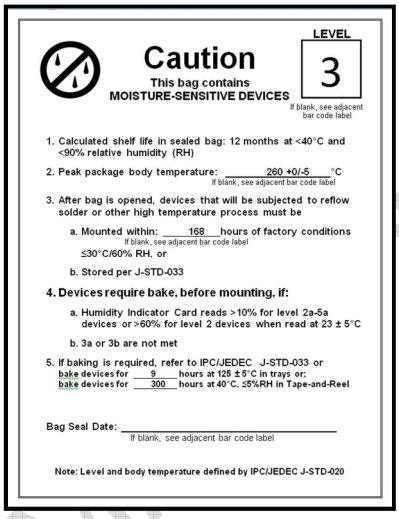
11.9 Label

Location of Label

11.10 Packaging

REEL – with Barcode & Caution labels

Vacuum-Sealed Moisture Barrier Bag with ESD, MSL3, Caution, and Barcode Labels



MSL3 Label

Caution Label

ESD Label

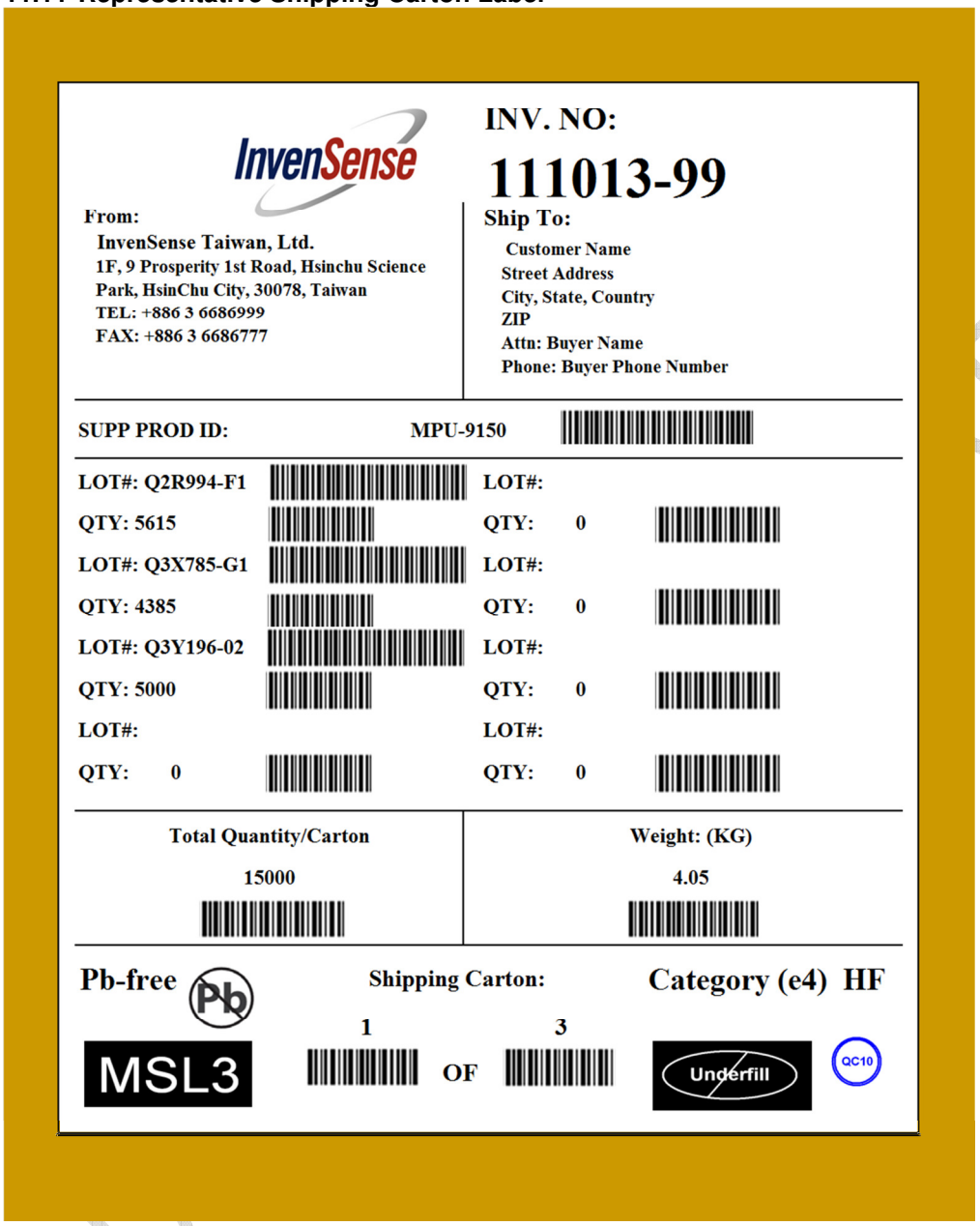
Inner Bubble Wrap

Pizza Box

Pizza Boxes Placed in Foam-Lined Shipper Box

Outer Shipper Label

11.11 Representative Shipping Carton Label



12 Reliability

12.1 Qualification Test Policy

InvenSense's products complete a Qualification Test Plan before being released to production. The Qualification Test Plan for the MPU-9150 followed the JESD 47H.01 Standards, "Stress-Test-Driven Qualification of Integrated Circuits," with the individual tests described below.

12.2 Qualification Test Plan

Accelerated Life Tests

TEST	Method/Condition	Lot Quantity	Sample / Lot	Acc / Reject Criteria
(HTOL/LFR) High Temperature Operating Life	JEDEC JESD22-A108D, Dynamic, 3.63V biased, Tj>125°C [read-points 168, 500, 1000 hours]	3	77	(0/1)
(HAST) Highly Accelerated Stress Test (1)	JEDEC JESD22-A118A Condition A, 130°C, 85%RH, 33.3 psia., unbiased, [read-point 96 hours]	3	77	(0/1)
(HTS) High Temperature Storage Life	JEDEC JESD22-A103D, Cond. A, 125°C Non-Bias Bake [read-points 168, 500, 1000 hours]	3	77	(0/1)

Device Component Level Tests

TEST	Method/Condition	Lot Quantity	Sample / Lot	Acc / Reject Criteria
(ESD-HBM) ESD-Human Body Model	JEDEC JS-001-2010, (1.5KV)	1	3	(0/1)
(ESD-MM) ESD-Machine Model	JEDEC JESD22-A115C, (200V)	1	3	(0/1)
(LU) Latch Up	JEDEC JESD-78D Class II (2), 125°C; ±100mA	1	6	(0/1)
(MS) Mechanical Shock	JEDEC JESD22-B104C, Mil-Std-883, Method 2002.5, Cond. E, 10,000*g's*, 0.2ms, ±X, Y, Z – 6 directions, 5 times/direction	3	5	(0/1)
(VIB) Vibration	JEDEC JESD22-B103B, Variable Frequency (random), Cond. B, 5-500Hz, X, Y, Z – 4 times/direction	1	5	(0/1)
(TC) Temperature Cycling (1)	JEDEC JESD22-A104D Condition N [-40°C to +85°C], Soak Mode 2 [5'], 100 cycles	3	77	(0/1)

Board Level Tests

TEST	Method/Condition	Lot Quantity	Sample / Lot	Acc / Reject Criteria
(BMS) Board Mechanical Shock	JEDEC JESD22-B104C,Mil-Std-883, Method 2002.5, Cond. E, 10000*g's*, 0.2ms, +-X, Y, Z – 6 directions, 5 times/direction	1	5	(0/1)
(BTC) Board Temperature Cycling (1)	JEDEC JESD22-A104D Condition N [-40°C to +85°C], Soak mode 2 [5'], 100 cycles	1	40	(0/1)

(1) Tests are preceded by MSL3 Preconditioning in accordance with JEDEC JESD22-A113F

13 Environmental Compliance

The MPU-9150 is RoHS and Green compliant.

The MPU-9150 is in full environmental compliance as evidenced in report HS-MPU-9150, Materials Declaration Data Sheet.

Environmental Declaration Disclaimer:

InvenSense believes this environmental information to be correct but cannot guarantee accuracy or completeness. Conformity documents for the above component constitutes are on file. InvenSense subcontracts manufacturing and the information contained herein is based on data received from vendors and suppliers, which has not been validated by InvenSense.