



UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Proyecto de Fin de Grado en Ingeniería Informática

## **REPRESENTACIÓN DE LA ORIENTACIÓN DE UN SISTEMA ARTICULADO**

Jaime de Arcos Sánchez

Dirigido por: María Llanos Tobarra

Codirigido por: Francisco Domínguez



## **REPRESENTACIÓN DE LA ORIENTACIÓN DE UN SISTEMA ARTICULADO**

Proyecto de Fin de Grado en Ingeniería Informática

De modalidad específica

Jaime de Arcos Sánchez

Dirigido por: María Llanos Tobarra

Codirigido por: Francisco Domínguez

Fecha de lectura y defensa

## Resumen

---

El presente documento ilustra todo el proceso de investigación y desarrollo que ha sido necesario para elaborar un dispositivo electrónico capaz de transmitir a un ordenador la información necesaria para representar la orientación de cada una de las articulaciones de un ser humano.

El dispositivo consiste en una red de nodos con sensores conectados mediante un bus I<sup>2</sup>C en modo esclavo, y un módulo maestro que gestiona la red y se comunica con un PC por medio de cable o bluetooth. Cada nodo posee a su vez un array de sensores (acelerómetro, giroscopio y magnetómetro de 3 ejes).

El producto se ha enfocado como un periférico, no obstante, se han desarrollado varias aplicaciones para PC a modo de demostración de las capacidades del dispositivo.

La calidad del producto en los 4 meses de desarrollo ha superado con creces las expectativas iniciales, a modo de resumen los hitos alcanzados:

**Tiempo real:** se estableció un mínimo de 10 muestras por segundo, para que las aplicaciones de representación pudieran mostrar gráficos mínimamente fluidos, se han alcanzado medias de 20 muestras por segundo después de la optimización de código. Todo esto con microcontroladores MUY básicos de arquitectura 8-bit a 16Mhz

**Tamaño:** Los nodos hardware diseñados se adaptan a unas dimensiones perfectamente funcionales, algo muy notable para haberse desarrollado con equipación casera. El mismo diseño con herramientas industriales podría ser comparable al de una tarjeta SD.

**Precisión y robustez:** Con los procesos de filtrado, fusión y calibración, la precisión del sistema alcanza cotas muy aceptables para aplicaciones de entretenimiento, incluso para algunos software de vuelo de aviones radiocontrol, además, el sistema no acumula error, es recuperable y robusto ante interferencias y errores en las medidas gracias al filtro complementario.

**Desacoplamiento y modularidad:** el sistema se ha diseñado con vistas a nuevos proyectos, haciendo que ciertos módulos software crítico sean reutilizables, y desacoplados del hardware del sistema, utilizando interfaces y arquitectura en capas.

Por otro lado, y como es evidente, en 4 meses de desarrollo, el producto todavía tiene mucho margen de mejora y posibilidades de expansión, entre otras funcionalidades, quedan pendientes:

- **Cambiar filtro complementario por filtro de Kalman:** descartado del proyecto por su coste computacional, si consideramos un cambio por controladores de arquitectura de 32 bits sería una gran mejora.
- **Más Aplicaciones demostrativas:** al centrar todos los esfuerzos en la calidad y prestaciones del dispositivo, solo se han desarrollado algunas aplicaciones demostrativas, para comprobar la funcionalidad de las mismas. No obstante, el sistema ofrece la base para el desarrollo de este tipo de aplicaciones sobre cualquier motor de videojuegos tipo Blender.

## Palabras Clave

---

Estimación de la orientación, filtro complementario, acelerómetro, giróscopo, giroscopio, fusión de sensores, cuerpos articulados, bus i2c, Arduino, microcontroladores, C++, Python

# Attitude estimation of an articulated system

---

## Summary:

This document intends to illustrate the whole process of research and development that has been necessary to develop an electronic device capable of transmitting the information needed to a computer in order to represent the orientation of each of the joints of a human being. In other words, we are talking about a device that shows a complete and accurate virtual representation of the position of a human being in real-time.

The device consists in a mote network connected by an I2C bus in slave mode and a master module that manages the network and communicates with a PC via bluetooth. Each node has in turn an array of sensors (accelerometer, gyroscope and 3-axis magnetometer).

The product has been focused as a peripheral; however, several applications for PC have been developed as a demonstration of the capabilities of the device.

Product quality has, by far, exceeded the initial expectations in the 4 months of development. We offer a summary of the accomplishments:

**-Real-Time:** A minimum of 10 samples per second was established, so the applications of representation could show minimally fluid graphics. An average of 20 samples per second has been achieved after the code optimization. All this with very basic 8-bit to 16 Mhz architecture microcontrollers.

**-Size:** The hardware nodes designed suit perfectly functional dimensions, something very remarkable despite being developed with homemade equipment. The use of Industrial tools could equate it to that of an SD card.

**-Accuracy and hardness:** With the filtering, merger and calibration processes, system accuracy reaches very acceptable levels for entertainment applications, even for some radio control aircraft software. In addition, the system does not accumulate errors; it is recoverable and robust against interferences and measurement errors due to the complementary filter.

**-Decoupling and modularity:** the system is designed with a view to new projects, making certain critical software modules reusable and uncoupled from hardware system, using interfaces and layered architecture.

On the other hand, as is evident in 4 months of development, the product still has much room for improvement and expansion possibilities.

Among other features, the next remain to be done:

- **Change complementary filter for a Kalman filter:** it has been considered the great achievement in the theory of estimation of the twentieth century, discarded from the project due to its computational cost, but if we changed from an Atmel based on chips controller with 8-bit to 16Mhz architecture to a new ESP8266 (32bit 80MHz), it would be perfectly supported.

- **More Demonstrative Applications:** initially, a goal was to develop an application similar to a video game that could show the benefits of the system, but the lack of time forced to make a decision. However, the system provides the basics for the development of these applications on any type Blender game engine.

## Keywords:

Orientation estimation, complementary filter, accelerometer, giroscope, magnetometer, sensor fusión, body joins, i2c bus, Arduino, microcontrollers, c++,Python, virtual reality.

# Índice

---

1.	INTRODUCCIÓN. ....	13
2.	ESTADO DEL ARTE. ....	15
3.	OBJETIVOS Y ESPECIFICACIONES .....	17
4.	METODOLOGÍA. ....	21
5.	HERRAMIENTAS MATEMÁTICAS Y TEORÍA. ....	23
5.1.	<i>Sistemas de representación matemática de las rotaciones. ....</i>	<i>23</i>
5.1.1.	Ángulos Euler. ....	24
-	Ángulos EULER (Z,X,Z): .....	24
-	ROLL, PITCH, YAW: .....	25
-	VENTAJAS y DESVENTAJAS: .....	26
5.1.2.	Matriz de Rotación. ....	26
-	Composición de Rotaciones: .....	26
-	Ventajas e inconvenientes: .....	27
5.1.3.	Cuaterniones. ....	27
5.2.	<i>Teoría de Sensores Inerciales. ....</i>	<i>29</i>
5.2.1.	Acelerómetros. ....	29
-	Características .....	30
-	Numero de ejes .....	30
-	Tecnología MEMS .....	30
-	Rango y precisión .....	31
-	Interfaz .....	31
-	Calibración de acelerómetros: .....	32
-	Offset .....	32
-	Factor de escala (Ganancia) .....	34
5.2.2.	Giróscopo. ....	35
-	Características .....	36
-	Numero de Ejes .....	36
-	Tecnología MEMS .....	36
-	Interfaz .....	36
-	Rango y precisión .....	36
-	Calibracion de Giroscopos. ....	36
-	Offset .....	36
-	Ganancia .....	37
5.2.3.	MPU-6050: Acelerómetro + Giróscopo. ....	37
-	Características Acelerómetro: .....	38
-	Características Giroscopio. ....	38
-	Características adicionales .....	38
-	GY-521 BreadBoard .....	38

- Configuración y uso: .....	39
- Test de conexión.....	39
- Inicialización .....	40
- Configuración de Rangos .....	40
- Adquisición de datos. ....	41
5.2.4. Magnetómetro. ....	42
- Campo Magnético terrestre. ....	42
- Norte Magnético y Norte Geográfico. ....	43
- Campo magnético como brújula. ....	44
- Magnetómetros para calcular rotaciones. ....	45
- Características de los magnetómetros .....	45
- Numero de ejes .....	45
- Tecnología MEMS.....	46
- Rango y precisión.....	46
- Interfaz .....	46
5.2.5. Magnetómetro HMC5883L.....	46
- Características: .....	47
- GY-271 BreadBoard .....	47
- Configuración y uso. ....	48
- Modo de operación. ....	48
5.3. Microcontrolador. ....	49
5.3.1. Características. ....	49
5.3.2. Modelos Considerados .....	50
- Arduino Pro Mini (Atmel 328p 16Mhz): .....	50
- ESP8266:.....	50
5.4. Protocolos de comunicación. ....	51
5.4.1. Puerto Serie RS-232.....	51
- Puerto Serie Arduino. ....	52
5.4.2. Bluetooth.....	53
- Hc-05 Modulo Bluetooth .....	53
- Pinout .....	54
- Funcionamiento.....	54
5.4.3. Protocolo I <sup>2</sup> C.....	55
6. MONTAJE DEL PROTOTIPO Y ADQUISICIÓN DE DATOS: .....	57
6.1. Hardware. ....	58
6.2. Módulos Software para Nodo Esclavo: .....	58
6.3. Aplicación de pruebas: sensoresEnBruto.py .....	61
7. TÉCNICAS DE FILTRADO Y FUSIÓN. ....	65
7.1.1. Estimación de la orientación con acelerómetro y magnetómetro. ....	65
7.1.2. Matriz de Rotación con giroscopio. ....	70
7.1.3. Velocidad Angular a partir del acelerómetro y magnetómetro.....	71
7.1.4. Filtro complementario.....	72



7.1.5.	Corrección de ortogonalidad de Matriz de rotación. ....	74
7.1.6.	Aplicación de prueba: Avion.py.....	76
8.	REDES DE COMUNICACIÓN.....	77
8.1.	<i>Comunicación Sensores-&gt;Arduino. (I<sup>2</sup>C Software)</i> .....	77
8.2.	<i>Comunicación Nodos-Eslavos-&gt;Nodo-Maestro</i> .....	78
8.3.	<i>Comunicación Maestro-PC</i> .....	79
8.4.	<i>Aplicación de prueba: Esqueleto.py.</i> .....	79
9.	ELEMENTOS ADICIONALES .....	83
9.1.	<i>Hardware:</i> .....	83
9.1.1.	Circuito Impreso (PCB). ....	83
-	Diseño del circuito Impreso.....	83
-	Realización de las placas por insolación .....	84
9.1.2.	Cableado.....	85
9.1.3.	Fijaciones:.....	85
9.2.	<i>Software:</i> .....	86
9.2.1.	Configuración del entorno de desarrollo.....	86
-	Java SDK 1.7.....	86
-	Eclipse.....	86
-	Plugin: PyDev.....	86
-	Python 2.7 (32Bits) .....	87
-	Librería vPython .....	87
-	Librería PySerial .....	87
-	Control de Versiones, SVN Subversion y Tortoise SVN.....	87
9.2.2.	Diagrama UML del Firmware para Arduino.....	88
10.	CONCLUSIONES .....	89
11.	BIBLIOGRAFÍA Y REFERENCIAS.....	91
12.	SIGLAS, ABREVIATURAS Y ACRÓNIMOS.....	95

## Índice de Figuras

---

Figura 1: Sistemas de referencias local y global.....	24
Figura 2: Representación gráfica de los ángulos de EULER.....	25
Figura 3: Representación gráfica EULER (roll,pitch,yaw) .....	25
Figura 4: Ejes de un sensor .....	30
Figura 5: Sistema MEMS para la aceleración en 1 eje.....	30
Figura 6: Señales PWM.....	32
Figura 7: Error de ganancia de un acelerómetro .....	33
Figura 8: Representación gráfica del offset de un acelerómetro.....	34
Figura 9: Representación gráfica del offset.....	36
Figura 10: Chip MPU6050.....	37
Figura 11: GY-521 .....	39
Figura 12: Registros de datos del MPU6050 .....	42
Figura 13: Líneas de Campo magnético de la tierra .....	43
Figura 14: Declinación Magnética .....	44
Figura 15: ejes de un magnetómetro .....	46
Figura 16: Chip HMC5883L .....	47
Figura 17 : GY-271 .....	48
Figura 18 : Arduino Pro Mini 5v 16Mhz.....	50
Figura 19 : ESP8266 .....	51
Figura 20 : Teensy 3.1.....	51
Figura 21: Conector DB9, para puerto serie.....	52
Figura 22 : Cable FTDI TTL-232R-5V .....	53
Figura 23 : Hc-05 modulo Bluetooth .....	53
Figura 24: comandos AT al módulo Bluetooth Hc-05 .....	55
Figura 25 : Componentes necesarios para montaje del nodo prototipo .....	57
Figura 26: Diseño del prototipo de nodo en fritzing .....	57
Figura 27: Prototipo en protoboard de nodo esclavo.....	58
Figura 28 : Diagrama UML de clase para el módulo MPU6050.....	59
Figura 29 : Diagrama UML de clase para el modulo del magnetómetro .....	60
Figura 30 : Sistema Local y sistema Global.....	66

Figura 31: Gravedad (0,0,-1) medida en Sistema Global.....	66
Figura 32: Gravedad medida en sistema Local (acelerómetro) .....	67
Figura 33: Producto Vectorial magnetómetro y acelerómetro (OESTE).....	68
Figura 34: Producto vectorial Acelerómetro y Oeste (amarillo) .....	68
Figura 35: cálculo de DCM(t+dt) a partir de la velocidad angular w.....	71
Figura 36: Velocidad angular a partir de los dos vectores desplazados.....	72
Figura 37: corrección ortogonalidad de dos vectores.....	74
Figura 38: corrección de la ortogonalidad repartiendo el error .....	75
Figura 39: Aplicación Avion.py .....	76
Figura 40: Redes de comunicación del dispositivo.....	77
Figura 41: esquema bus i2c sensores-Arduino.....	78

## Índice de Tablas

---

Tabla 1 : Comandos que el periférico debe aceptar .....	19
Tabla 2: Descripción del byte (registro) 27 .....	40
Tabla 3: Selección de Rango del giroscopio .....	40
Tabla 4 : Registro 28: configuración para el acelerómetro. ....	41
Tabla 5 : Selección del rango del acelerómetro. ....	41
Tabla 6: Conexión de los pines de los componentes .....	58
Tabla 7: Cálculo de sistema Global a partir de datos de los sensores .....	69



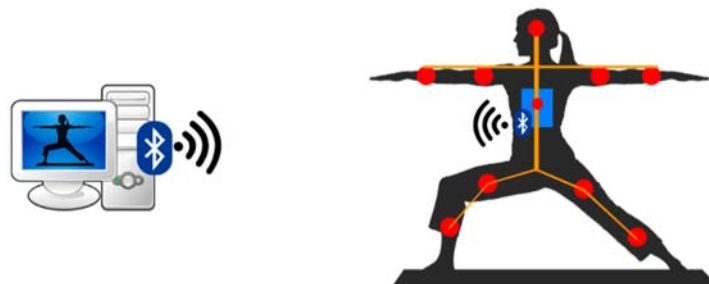
# 1. Introducción.

---

El presente Proyecto Fin de Grado ilustra todo el proceso de investigación y desarrollo que ha sido necesario para elaborar un dispositivo electrónico capaz de transmitir a un ordenador la información necesaria para representar la orientación de cada una de las articulaciones de un ser humano.

El dispositivo consiste en una red de nodos con sensores, gestionada por un nodo maestro que se conectará al PC mediante cable o bluetooth.

El dispositivo se va a desarrollar como un periférico, centrando el esfuerzo en éste, no obstante, se van a desarrollar varias aplicaciones para PC, para poder probar su funcionamiento y presentar los resultados.



La motivación principal para realizar este proyecto ha sido poder investigar e implementar dispositivos y técnicas para conocer la orientación de un cuerpo a través de sensores sin referencias externas, con el fin de obtener un conocimiento y una experiencia para en un futuro, poder estudiar técnicas de control automático como PID, y poder implementarlas en otras materias como el desarrollo de aviones no tripulados (UAVs).



## 2. Estado del Arte.

---

El seguimiento del movimiento humano es un campo muy de actualidad en relación a la interacción hombre-máquina, entre otros motivos, muy propiciado, por las nuevas tecnologías MEMS que han dado lugar a sensores de movimiento asequibles, precisos y de muy reducidas dimensiones.

La mayoría de las aplicaciones se concentran en la realidad virtual, como en los videojuegos, de hecho, las últimas **consolas de videojuegos** poseen periféricos de control que incluyen acelerómetros y giroscopios para controlar el movimiento de los personajes, en este campo, destaca por ser pionera, Wii de la marca nipona Nintendo.

Además de los videojuegos, existen multitud aplicaciones, como la creación de películas, en las que se pueden diseñar modelos 3d de personajes y generar las animaciones y movimientos imitando los de un actor con estos dispositivos. En el deporte, se han implementado aplicaciones para mejorar la técnica de los usuarios, como por ejemplo en el golf, un ordenador puede comparar el swing de un jugador con el prototipo del swing perfecto y darnos consejos para mejorar.

Existen multitud de dispositivos que han propiciado el uso de estos sensores, que miden la orientación de un objeto. Los **teléfonos móviles** integran acelerómetros para medir la orientación de la pantalla; Los **UAVs, (vehículos aéreos no tripulados)**, coloquialmente conocidos como drones, son otro ejemplo de dispositivo que hace uso de estos sensores, siendo en este caso críticos para su funcionamiento, los más avanzados hacen uso de acelerómetros, giroscopios, magnetómetros, sistemas GPS y cámaras. Los robots usualmente los necesitan para navegación, siendo cada vez más usual ver como sustituyen o combinan con los clásicos encoders de motores.

Todo indica que el uso de este tipo de sensores va a seguir en aumento, ya que cada vez son más las aplicaciones que requieren sus bondades.

## Representación de la orientación de un sistema articulado

Respecto a la representación de las articulaciones de un cuerpo humano, actualmente hay muchas empresas desarrollando prototipos similares al que este proyecto aspira a desarrollar.

La empresa Prio-VR ([priovr.com](http://priovr.com)) actualmente está desarrollando un traje de realidad virtual para su uso en el mercado de videoconsolas, y a pesar de que aún no se está distribuyendo, ya incluye avances muy significativos en 3D tracking, con lo que además de la orientación de las articulaciones, también es capaz de calcular la posición global del cuerpo y sus desplazamientos.



# 3. Objetivos y especificaciones

La funcionalidad que se desea desarrollar para el periférico debe estar orientada a los potenciales desarrolladores de aplicaciones o videojuegos que quieran hacer uso del traje, de modo que es necesario definir una interfaz con el periférico, lo más desacoplada posible, de modo que los desarrolladores que hagan uso de ella no deban conocer la implementación y sea lo más transparente posible para ellos.

La funcionalidad que el dispositivo va proporcionar será la siguiente:

- **Conectividad:** el dispositivo se conectará al PC mediante cable o bluetooth, en cualquiera de los dos casos el PC lo reconocerá como un puerto serie adicional, a través del cual el desarrollador podrá conectarse mediante cualquiera de las múltiples librerías que existen para los diferentes lenguajes de programación. Los parámetros de conexión al puerto serie vienen definidos por el dispositivo y no podrán modificarse. La velocidad de la comunicación (conocida en inglés como baudrate) será de 250.000 baudios para el caso de conexión cableada y de 230.400 baudios para el caso de la conexión bluetooth, los demás parámetros serán por defecto.
- **Comunicación:** el dispositivo actúa en modo servidor, bajo el paradigma cliente-servidor, donde el periférico siempre está a la espera de comandos del cliente que los consume. Los comandos aceptados por el periférico definen todas sus posibilidades, pero dejan la puerta abierta a posibles actualizaciones de firmware únicamente añadiendo más comandos.
- **Configuración:** a través de los comandos el desarrollador de aplicaciones debe tener acceso a cierta configuración interna del periférico como son valores de calibración, valores en bruto, posición base del sensor...

- **Datos:** los datos proporcionados por el periférico dependen de la entrada, pero se transmitirán como una cadena de caracteres por el puerto serie. En el caso de las orientaciones de cada nodo siempre se servirán en forma de matriz de rotación, se deja abierta la posibilidad de ampliar la funcionalidad en versiones futuras a representación en cuaterniones.
- **Comandos:** la siguiente tabla 1 describe los comandos que debe aceptar el dispositivo, la respuesta esperada, y el formato de las mismas. A modo de capa de seguridad, el dispositivo debe borrar el buffer de entrada después de cada carácter recibido y no aceptar nada que se salga de la siguiente tabla, para evitar posibles hackeos por desbordamiento de buffer o fallos del dispositivo ante problemas de conexión.

CMD	Respuesta
<b>0-20</b>	<b>Matriz de Rotación Esclavos (num):</b> Devuelve la Matriz de rotación del nodo cuyo número sea el de la llamada <b>(9 valores separados por comas)</b> <i>el 0 siempre es el maestro</i>
<b>a</b>	<b>Matriz de Rotación nodo Maestro:</b> Devuelve la matriz de rotación del nodo maestro <i>(9 valores separados por comas)</i>
<b>b</b>	<b>Valores sensores Procesados Nodo Maestro :</b> Devuelve las magnitudes físicas calculadas para cada sensor en los 3 ejes (gyro-> grados/seg, acelerómetro -> fuerzas G , magnetómetro -> miliGauss) <i>(9 valores separados por comas)</i>
<b>c</b>	<b>Valores en Bruto nodo Maestro :</b> Devuelve los valores en bruto que se del conversor ADC de los sensores en binario para cada sensor en los 3 ejes <i>(9 valores separados por comas)</i>
<b>d</b>	<b>Imprimir memoria :</b> Devuelve por pantalla los valores de configuración almacenados en la EEPROM

### 3. Objetivos y especificaciones.

<b>e</b>	<b>Calibración MPU:</b> Inicia el proceso para calibrar offset los sensores
<b>f</b>	<b>Calibración HMC (1):</b> Inicia el proceso para calibrar ganancia de los sensores
<b>g</b>	<b>Calibración HMC (2):</b> Inicia el proceso para calibrar offset de los sensores
<b>h</b>	<b>EEPROM init:</b> Resetea la EEPROM y carga valores por defecto de configuración
<b>i</b>	<b>EEPROM zero:</b> Borra por completo la EEPROM
<b>j</b>	<b>Cambiar dir I2C:</b> Cambia el valor de la dirección del nodo en la red
<b>K</b>	<b>Solo giróscopo:</b> Configuración para funcionar solo con giróscopo
<b>l</b>	<b>Solo acelerómetro:</b> Configuración para funcionar solo con acelerómetro
<b>m</b>	<b>Solo magnetómetro:</b> Configuración para funcionar solo con magnetómetro
<b>n</b>	<b>Configuración estándar:</b> Utiliza todos los sensores, funcionamiento por defecto.

*Tabla 1 : Comandos que el periférico debe aceptar*

Esta tabla representa además de la interfaz o protocolo con el que el desarrollador de futuras aplicaciones tendrá que comunicarse con el periférico, una oportunidad para desarrollar según el **patrón TDD (Test-driven development)**, o desarrollo guiado por pruebas, ya que se va a desarrollar aplicaciones demo, que sirvan a la vez como demostración de las capacidades del producto, como para realizar las pruebas de los diferentes módulos software y se basaran en esta interfaz.

## Representación de la orientación de un sistema articulado

Esta manera de trabajo nos asegura, en primer lugar que se han entendido los requisitos, ya que de otra manera, no se puede escribir el código necesario para la prueba, y por otra parte, teniendo las pruebas, es más fácil la integración.

# 4. Metodología.

Para el desarrollo del periférico se ha optado por la realización en etapas, bien definidas, con unos objetivos a alcanzar en cada iteración o etapa, haciendo del desarrollo una tarea incremental y consiguiendo que tanto la programación como la integración sea lo más sencilla posible. Las etapas son:

- **Herramientas matemáticas y teoría:**

Proceso de investigación para adquirir las competencias teóricas necesarias para poder afrontar el problema, sistemas de representación matemática de las rotaciones de un cuerpo, tipos de sensores, filtrado de señales, fusión de sensores,... Es la etapa compleja, ya supone un gran esfuerzo de estudio e investigación en muy diversas áreas. De esta etapa se espera conseguir una base sólida de conocimientos sobre la que empezar a desarrollar el dispositivo.

- **Montaje del prototipo de nodo:**

- Montaje de un nodo en protoboard: un Arduino pro mini (microcontrolador) conectado a los sensores (acelerómetro, magnetómetro y giróscopo) MPU6050 y un HMC5883L.
- Módulos Software necesarios para:
  - Recuperar la información de los sensores.
  - Procesar y convertir esos datos en bruto a magnitudes físicas.
  - Protocolo de comunicación para que responda a comandos.
- Pruebas: la aplicación de demostración sensoresEnBruto.py está programada para comprobar la funcionalidad de esta etapa. Muestra gráficamente los vectores en un sistema de coordenadas.

- **Cálculo de la orientación, técnicas de filtrado y fusión:**

En esta etapa el objetivo principal es programar el algoritmo que sea capaz de inferir la orientación del nodo a partir de la información de los sensores y ponerla a disposición representándola en forma de matriz de rotación, para ello es necesario utilizar técnicas de fusión de sensores para hacer uso de la información que proporciona cada uno de ellos y por último aplicar el conocimiento que tenemos de las fortalezas, debilidades y fuentes de ruido e interferencias de cada sensor para filtrar las señales y obtener una salida lo más robusta y limpia posible.

### - **Red de comunicación y nodo maestro:**

En esta etapa el objetivo es realizar el montaje de varios nodos como el del apartado anterior y diseñar un nodo maestro que sea capaz de comunicarse con los demás para solicitarles la información de su orientación, de modo que este nodo reúna la información de toda la red y sea capaz de transmitirla al PC por puerto serie o bluetooth.

### - **Aplicaciones de escritorio:**

Esta etapa es en realidad un trabajo paralelo al de las otras etapas en la que se va a desarrollar las aplicaciones de PC en Python que van a hacer uso del periférico y que van a tener un doble objetivo, en primer lugar servirán como aplicaciones demo para poder mostrar las posibilidades del periférico, y por otra parte serán las que servirán de prueba en la fase de desarrollo para comprobar que el dispositivo funciona tal y como fue pensado que lo haría.

### - **Mejoras y ampliaciones:**

Esta fase queda abierta para posibles ampliaciones futuras si fuera viable:

- Comunicación inalámbrica entre nodos.
- Desarrollo de aplicaciones más complejas
- Migración de las aplicaciones a un entorno más profesional (como es el entorno de desarrollo 3D [21] (Blender.org, 2016) ).

# 5. Herramientas matemáticas y teoría.

## 5.1. Sistemas de representación matemática de las rotaciones.

El objetivo principal del sistema es proporcionar información de la orientación de las articulaciones de un cuerpo, de modo que necesitamos una manera para representar esas orientaciones o rotaciones. En la teoría al respecto existen varias maneras de representar las rotaciones, todas ellas con sus pros y sus contras, algunas son más compactas en lo que a cantidad de valores necesarios respecta y otras disponen de un algebra más eficiente en términos computacionales.

Existe muchísima bibliografía que trata el problema de representar matemáticamente las rotaciones, como [2, Fundamentos de Robótica] y a pesar de que todas son coherentes entre sí, abordan el problema desde diferentes puntos de vista.

Para representar un punto en el espacio, con su posición en coordenadas queda completamente definido, pero para representar un sólido es necesario definir cuál es su orientación respecto a un sistema de referencia, ya sea en forma de ángulos de Euler, Matriz de rotación, cuaterniones o cualquier otra representación equivalente.

Se han de utilizar sistemas de coordenadas. Para especificar la orientación de un sólido, lo haremos con un sistema de coordenadas de ejes ortogonales, normalmente móvil, que llamaremos **SISTEMA LOCAL**, respecto de otro sistema de referencia de ejes fijos, también ortogonales, que llamaremos **SISTEMA GLOBAL**.

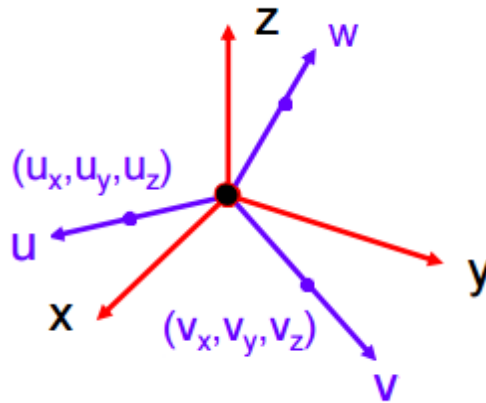


Figura 1: Sistemas de referencias local y global

De ahora en adelante definiremos al sistema de coordenadas Global por sus 3 vectores perpendiculares **OXYZ** vectores unitarios (modulo la unidad)  $X=(1,0,0)$   $Y=(0,1,0)$   $Z=(0,0,1)$  y al sistema de coordenadas Local como **OUVW**, que si no tenemos ninguna rotación  $U \rightarrow I$ ,  $V \rightarrow J$ ,  $W \rightarrow K$ .

En este documento no vamos a tratar las translaciones, es decir, los cambios de posición de los cuerpos, ya que no es necesario para el desarrollo del problema, de modo que los orígenes de coordenadas siempre serán los mismos para el sistema local y el sistema global.

#### 5.1.1. Ángulos Euler.

La representación por ángulos de Euler es la más sencilla e intuitiva de todas, consisten en un conjunto de tres coordenadas angulares, que definen los giros realizados sobre los ejes. Existen diferentes variaciones con diferentes denominaciones como ZXZ, UVU, UWU,... las cuales hacen referencia a los ejes que rotemos para llegar a definirla.

##### - Ángulos EULER (Z,X,Z):

Una de las más habituales, porque es la asociada a los movimientos básicos de un giróscopo, cuya representación puede verse en la Figura 2. Es importante destacar que cada nuevo giro se realiza sobre el sistema ya girado en el paso previo.



## 5. Herramientas matemáticas y teoría.

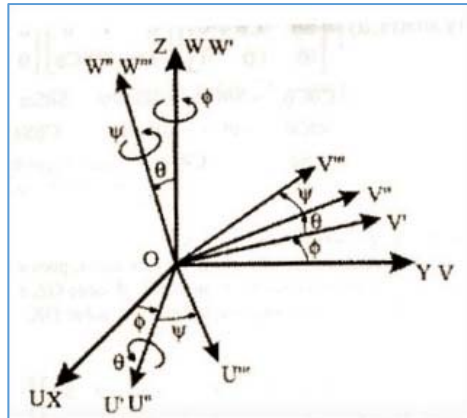


Figura 2: Representación gráfica de los ángulos de EULER

Partiendo de los sistemas OXYZ y OUVW coincidiendo, podemos situar el sistema OUVW en cualquier orientación siguiendo los siguientes pasos.

1. Girar el sistema OUVW un ángulo  $\Phi$  con respecto al eje OZ, convirtiéndose así en el OU'V'W'.
2. Girar el sistema OU'V'W' un ángulo  $\theta$  con respecto al eje OU', convirtiéndose así en el OU''V''W''.
3. Girar el sistema OU''V''W'' un ángulo  $\Psi$  con respecto al eje OW'' convirtiéndose finalmente en el OU'''V'''W'''.

### - ROLL, PITCH, YAW:

Esta representación es interesante debido a que es la utilizada en aeronáutica y muy intuitiva, se corresponde a giros sobre ejes globales. En la Figura 3 se observa cómo se representa en la aviación para representar la orientación de una aeronave.

De este modo (0, 0, 90) corresponde a un giro de la nave sobre el eje z de 90°

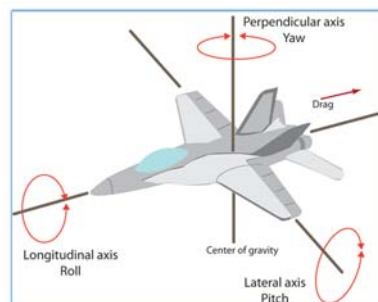


Figura 3: Representación gráfica EULER (roll,pitch,yaw)

- **VENTAJAS y DESVENTAJAS:**

Los ángulos EULER tienen la ventaja principal de que su representación en tamaño de datos es mínima, son 3 números nada más, por lo que la representación en espacio es mínima.

El mayor problema o desventaja es que los ángulos EULER sufren de una peculiaridad conocida como **GIMBALL LOCK**, que hace que ciertas rotaciones sean conflictivas.

### 5.1.2. Matriz de Rotación.

Las matrices de rotación son un método muy extendido para la descripción matemática de las rotaciones, entre otras ventajas, la principal es que como su representación es matricial, se beneficia de la comodidad que proporciona el álgebra matricial.

Una matriz de rotación en el espacio tridimensional es una matriz de 3x3. La matriz que rotación que corresponde al estado inicial, es decir a ninguna rotación sería la **matriz identidad**, y equivale a cuando los ejes OXYZ y OUVW son coincidentes:

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

- **Composición de Rotaciones:**

Y a partir de esta, podemos representar la rotación de un cuerpo como composición de rotaciones sobre cada uno de los ejes. La composición de rotaciones en el álgebra de las matrices de rotación se realiza con el producto de matrices.

Existen 3 matrices denominadas **matrices básicas de rotación**, que representan las rotaciones para cada uno de los 3 ejes.

*Matriz básica rotación eje X*

$$R(x, \alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\operatorname{sen} \alpha \\ 0 & \operatorname{sen} \alpha & \cos \alpha \end{bmatrix}$$

## 5. Herramientas matemáticas y teoría.

*Matriz básica rotación eje Y*

$$R(y, \alpha) = \begin{bmatrix} \cos \alpha & 0 & \sin \alpha \\ 0 & 1 & 0 \\ -\sin \alpha & 0 & \cos \alpha \end{bmatrix}$$

*Matriz básica rotación eje Z*

$$R(Z, \alpha) = \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Para formar la matriz de rotación resultante de las 3 rotaciones en los 3 ejes, se realiza multiplicando las matrices básicas en el orden  $R(Z, \alpha) * R(Y, \alpha) * R(X, \alpha)$ .

### - **Ventajas e inconvenientes:**

Las matrices de rotación son sin duda una de las representaciones más utilizadas, ya que la representación matricial es muy flexible, sin embargo, esta representación es muy redundante, ya que necesita 9 valores para representar la orientación de un cuerpo, por comparar, los ángulos Euler utilizan 3 veces menos valores (3 valores frente a 9), pero en cambio, no adolece del problema del Gimbal lock propio de los ángulos EULER.

### 5.1.3. Cuaterniones.

Otra representación muy interesante, a la vez que compleja y poco intuitiva, son los cuaterniones.

Fueron creados por William Rowan Hamilton en 1843, como una extensión de los números complejos, para las 3 dimensiones. Se expresan como un conjunto de un escalar 'u' y un vector 'w' de 3 componentes (u, w) o (u, w<sub>x</sub>, w<sub>y</sub>, w<sub>z</sub>)

- **Ventajas e inconvenientes:**

Su uso está muy extendido tanto en robótica como en diseño de videojuegos y entornos 3d. Su mayor ventaja es la eficiencia computacional del álgebra asociado, principalmente útil en el cálculo de trayectorias.

La desventaja que presenta esta representación es que es la menos intuitiva de todas, además que los cálculos y el álgebra asociados son poco habituales y por tanto requieren de conocimientos avanzados de matemáticas y cálculo.

## 5.2. Teoría de Sensores Inerciales.

Teniendo en cuenta las especificaciones del sistema debemos recurrir a 3 tipos de sensores, acelerómetros, giróscopos y magnetómetros. La elección no es trivial, ya que cada uno nos ofrece una ventaja pero cada uno tiene sus propias debilidades, como veremos a continuación.

### 5.2.1. Acelerómetros.

Los acelerómetros son dispositivos que miden la aceleración que se les aplica, ya sea la gravedad, las vibraciones o cualquier otro tipo de cambio en el movimiento.

La aceleración es **la tasa de cambio de la velocidad de un objeto**. La unidad de medida en el sistema internacional es metros por segundo al cuadrado ( $\text{m/s}^2$ ), aunque también está muy extendido la medida de esta magnitud en **fuerzas G**, que es una unidad más intuitiva ya que hace referencia a la atracción terrestre, siendo 1 unidad G equivalente a la fuerza de la gravedad terrestre en condiciones ideales sin atmosfera y a nivel del mar; la equivalencia es  $1\text{G} = 9,8 \text{ m/s}^2$  aproximadamente.

Su mayor **ventaja** para medir la orientación es que son sensores que se basan en una referencia externa, que es la gravedad y por tanto, no acumulan ruido, cada medida es independiente de la anterior.

Como **desventaja** principal está el problema de que al querer tomar como referencia la aceleración de gravedad, si el sistema está en movimiento, el acelerómetro mide todas las aceleraciones adicionales del sistema siendo incapaz de discernir por sí mismo que parte pertenece a la gravedad.

### - Características

#### - Número de ejes

Los acelerómetros actuales pueden medir la aceleración en uno, dos o tres ejes, también denominados grados de libertad (DOF) y denominados habitualmente X, Y, Z, por comparación con los ejes cartesianos en 3 dimensiones (ver Figura 4). Es evidente que necesitamos un acelerómetro de 3 ejes, ya que nuestra aplicación opera en las 3 dimensiones.

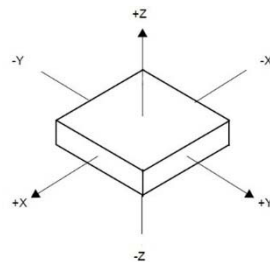


Figura 4: Ejes de un sensor

#### - Tecnología MEMS

Los acelerómetros actuales más utilizados son dispositivos electromecánicos de escala (habitualmente) micrométrica o más pequeñas. También denominados 'Micro Máquinas' en Japón o tecnología de Micro Sistemas MST (en Europa).

Los sensores MEMS (ver Figura 5) usan una estructura de placas de silicio suspendidas, algunas fijas y otras móviles que varían su posición según la aceleración que se le aplique. Esta posición se monitoriza en relación con la posición original debido a que las placas son capacitivas, este cambio de posición se convierte a una señal digital proporcional a la aceleración que sufre el sensor.

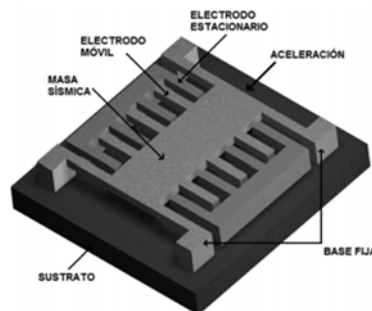


Figura 5: Sistema MEMS para la aceleración en 1 eje.

## 5. Herramientas matemáticas y teoría.

### - Rango y precisión

La mayoría de los acelerómetros tienen un **rango** de aceleración que pueden medir, es decir, el valor mínimo y máximo de aceleración que pueden medir, para valores superiores, el acelerómetro devolvería su máximo y para inferiores su mínimo. Lo habitual es que este rango sea seleccionable, los más típicos son  $\pm 1g$ ,  $\pm 2g$ ,  $\pm 4g$ ,  $\pm 8g$ ,  $\pm 16g$ , pero depende de cada sensor.

La precisión de un acelerómetro es el número de bits de su conversor ADC, habitualmente 16bits, este parámetro es va ligado al del rango ya que según el rango seleccionado, la precisión efectiva aumenta o disminuye.

Es importante destacar que un mayor rango no es intrínsecamente mejor, ya que la precisión disminuye, para ejemplificar este concepto, supongamos un acelerómetro que puede operar en un rango  $\pm 1g$  y  $\pm 2g$ , cada medida que hace el sensor la recibimos en un registro de 16bits, de modo que un valor 0 equivaldría a un valor de  $-2g$  o menor, y un valor de  $65.535 (2^{16})$  a  $+2g$ , esto nos dice que cada bit que incrementa equivale a un incremento de  $2/65.535 = 0.000030g$ , este es el intervalo más pequeño que podría medir el sensor con este rango seleccionado. Si seleccionamos el rango a  $\pm 1g$  el intervalo más pequeño que puede medir es  $1/65.535 = 0.000015g$ .

Una buena analogía es como si tuviéramos varias reglas para medir distancias de varios tamaños, de 1, 2, 4, 8... metros, pero todas tienen solo 16 líneas divisorias, y por tanto tenemos que elegir la regla en función de lo que queramos medir.

Para medir pequeñas vibraciones sobre una mesa, utilizando un acelerómetro de gama pequeña  $\pm 1g$  proporcionará datos más detallados que el uso de uno de  $250g$  (que es más adecuado para cohetes).

### - Interfaz

La interfaz de un acelerómetro es la manera que tiene el dispositivo de entregarnos los datos, existen 2 tipos de interfaz: Analógica, Digital (I2C o SPI) o por Modulación de Ancho de Pulso (PWM). Las más habituales son las 2 primeras.

Interfaz analógica: producen una tensión de salida proporcional a la aceleración detectada. En 0g la salida será el valor intermedio entre 0v y el voltaje de referencia (habitualmente 3,3v o 5v), es la interfaz más sencilla, siendo únicamente necesario un conversor analógico-digital que la mayoría de microcontroladores ya disponen integrado.

Interfaz de PWM: producen una onda cuadrada con una frecuencia fija, pero el ciclo de trabajo (pulso arriba) variará con la aceleración detectada (ver Figura 6). Estos modelos son bastante raros.

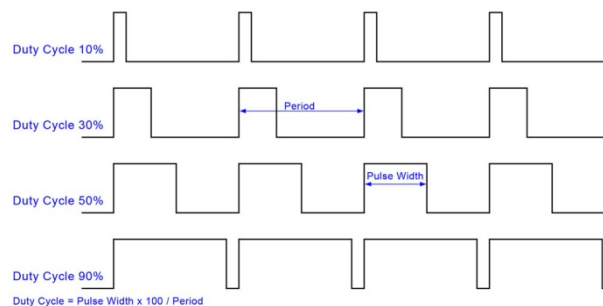


Figura 6: Señales PWM

Acelerómetros digitales: son los más avanzados, generalmente poseen una interfaz de comunicación serial tipo SPI o I<sup>2</sup>C. Pueden considerarse los más complejos de implementar, pero realmente son los más populares ya que tienen más características y además el tratamiento de señales analógicas siempre es más susceptible a la aparición de ruido.

### - Calibración de acelerómetros:

Los acelerómetros como todos los sensores, son propensos al ruido y tienen desviaciones y particularidades que hacen necesario un proceso de calibración para su correcto funcionamiento. Los acelerómetros, adolecen de dos tipos de desviaciones: Offset y Factor de escala (ganancia).

#### - Offset.

Los acelerómetros internamente toman medidas analógicas y las convierten a digital con un conversor ADC de una precisión de x bits, de modo que si disponemos de un acelerómetro con una precisión de 16 bits, las medidas que nos proporciona estarán en el rango de valores  $0 - 2^{16}$  [0-65535].



## 5. Herramientas matemáticas y teoría.

Teniendo esto en cuenta, el sensor en estado de reposo debería devolver un valor idealmente centrado entre esos valores, es decir 32.768, El offset es la desviación que tiene un sensor respecto de esa medida ideal. La siguiente gráfica, en la Figura 7, muestra la salida real y la esperada para un acelerómetro de rango  $\pm 1g$

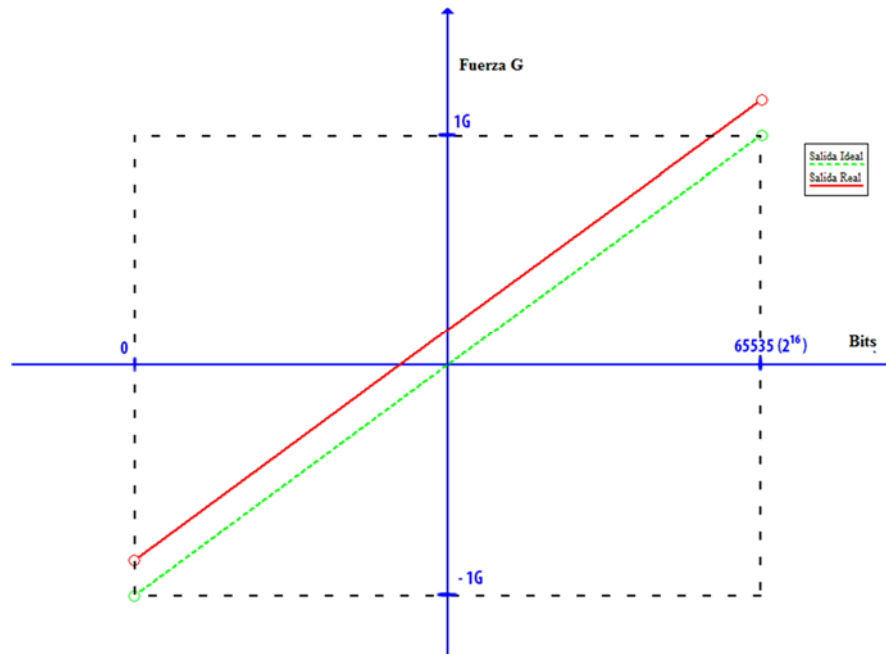


Figura 7: Error de ganancia de un acelerómetro

La corrección de este tipo de desviación es sencilla, lo primero que debemos hacer es **calcular el offset**, el algoritmo es el siguiente:

- Tomar muchas medidas del sensor en estado de reposo.
- Calcular la media
- Restar a la media, el valor ideal. Este valor es el offset.

*[Este método es válido para el eje X y el eje Y cuando el sensor está completamente horizontal, de modo que la aceleración de la gravedad actúa únicamente sobre el eje Z. Para calcular el offset del eje Z, debemos dejar el sensor en reposo, tomar las medidas y sumarle el valor en bits correspondiente a 1G que es lo que debería estar midiendo en esa posición]*

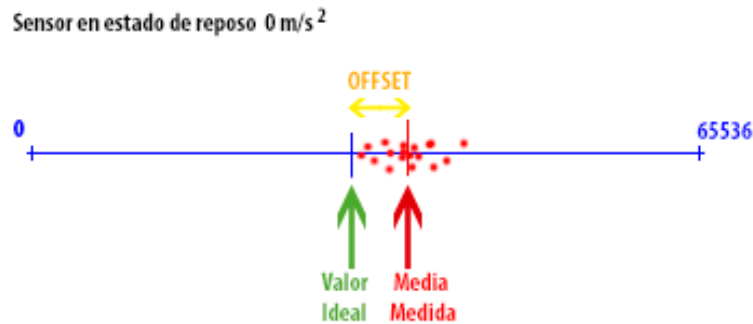


Figura 8: Representación gráfica del offset de un acelerómetro

Una vez calculado el Offset, lo siguiente es **rectificar la salida del sensor**, para ello analizamos la gráfica, la salida del sensor se puede aproximar a la ecuación de una recta ( $y = mx + b$ ) donde 'y' es la salida del sensor en fuerza G, 'm' sería el factor de escala (*ver más adelante*), 'x' sería los bits que nos devuelve el conversor ADC, y 'b' sería la desviación Offset.

- $Salida\_ideal = escala \cdot \#bits$
- $Salida\_Real = escala \cdot \#bits + offset$
- $Salida\_corregida = Salida\_Real - offset$

### - Factor de escala (Ganancia)

Como se indica al principio de este punto, existe otro tipo de desviación en los acelerómetros que es la desviación de la ganancia. La ganancia es un valor que nos proporciona el fabricante del dispositivo y es una constante que nos sirve para convertir el número en bits que devuelve el sensor, a fuerzas g o metros por segundo, la corrección de este parámetro es más compleja que la del anterior. En principio si el sensor funciona correctamente la desviación de este tipo no debería ser muy apreciable.

Para comprobar si la ganancia que nos proporciona el fabricante es correcta, tomamos varias medidas del acelerómetro en reposo sobre una mesa bien alineada. Después de corregir el offset del sensor, la medida en fuerzas g que nos debe devolver el sensor es exactamente 1 (la de la gravedad actuando completamente alineada con el eje z del acelerómetro).

## 5. Herramientas matemáticas y teoría.

El proceso para calcular la ganancia de un acelerómetro se basa en la gravedad, que es una aceleración ya conocida en magnitud y dirección, de modo que si alineamos un eje del acelerómetro con el de la gravedad (centro de la tierra), el valor que esperamos devuelva el acelerómetro es igual a 1g o  $9.8\text{m/s}^2$ , y si invertimos el dispositivo -1g. Por tanto el algoritmo para calcular la ganancia de cada eje:

- Tomar muchas medidas del sensor alineando el eje con la gravedad
- Calcular la media de estas medidas (Media\_positiva)
- Rotar 180º el sensor para alinearlo con la dirección inversa a la gravedad
- Calcular la media de estas medidas (Media\_negativa)

$$\text{GananciaEjeX} = (\text{Media\_positiva} - \text{Media\_negativa}) / 2$$

Por último, después del cálculo de la ganancia y del offset, la formula efectiva para obtener la aceleración en fuerzas g de un acelerómetro es la siguiente.

$$\text{FuerzaG} = (\text{GANANCIA} * \text{bits}) - \text{OFFSET}$$

### 5.2.2. Giróscopo.

Los giróscopos son dispositivos que miden la **velocidad angular** que sufre un cuerpo, esto se mide en el sistema internacional en radianes por segundo (rad/s) o en otras ocasiones también es habitual verlo en º/s.

Su mayor **ventaja** es que son unos sensores muy precisos, con muy bajo ruido y se ven muy poco afectados por las vibraciones.

Como **desventaja**, hay un error que se introduce al tratar de calcular la posición a través de un giróscopo ya que para calcular la posición de un cuerpo a través de la velocidad angular, es necesario conocer su posición anterior, y el tiempo transcurrido a esa velocidad, lo cual casi siempre introduce pequeños errores que se van acumulando. Estos errores se denominan en inglés '**drift**' o desviación, y se pueden minimizar si calculamos sobre incrementos muy pequeños de tiempo, pero siempre se introduce un pequeño error, y este error es acumulativo.

- **Características**

Los giróscopos básicamente se pueden clasificar en las mismas propiedades que los acelerómetros:

- Número de Ejes
- Tecnología MEMs
- Interfaz
- Rango y precisión

Estas propiedades tienen un significado equivalente al que se describió en el apartado de los acelerómetros, salvo pequeñas peculiaridades que no son relevantes para el proyecto.

- **Calibración de Giroscopos.**

Los giróscopos adolecen de los mismos problemas que los acelerómetros:

- Offset

En el caso del offset, la estrategia es la misma. Primero debemos calcular el valor de la desviación, y luego se ha de utilizar para realizar las medidas

Para calcular el offset, debemos de situar el sensor en estado de reposo, y realizar múltiples medidas, y calcular su media, tal y como se ve en la imagen. El valor de offset corresponde a la diferencia entre la media calculada y el valor ideal que debería tener el sensor en estado de reposo, es decir, la mitad del rango de valores (en la imagen 16 bits).



Figura 9: Representación gráfica del offset

## 5. Herramientas matemáticas y teoría.

Una vez calculado el offset, para corregir las desviaciones de las medidas que tomemos con el sistema, la fórmula es la siguiente:

$$\text{Medida corregida} = \text{medida sensor} - \text{offset}$$

### - Ganancia

Para corregir la ganancia de un giróscopo es necesario herramientas de mucha precisión, y queda fuera del alcance de este documento su elaboración, ya que para comprobar si un giróscopo nos devuelve un valor de velocidad angular adecuado, debemos girarlo a esa velocidad con una herramienta de precisión que nos asegure que está girándolo a esa velocidad concreta y comparar con el valor que el giróscopo devuelve en ese momento, (equivalente al proceso del acelerómetro), lo cual no se puede hacer de modo casero” sin instrumentación precisa y cara.

### 5.2.3. MPU-6050: Acelerómetro + Giróscopo.

El MPU6050 [23, Invensense.com] (ver Figura 10) es un chip compuesto de un acelerómetro, un giroscopio y un procesador interno para cálculo 3D de orientaciones. Es un chip económico, (apenas 1.5€ en aliexpress.com), de unas prestaciones muy superiores a sus competidores en lo que a precisión se refiere. Utiliza el protocolo de comunicación I2C, con la dirección 0x68, y dispone de un bus independiente para conectar un magnetómetro al cual se puede acceder desde los registros del bus principal.

Figura 10: Chip MPU6050



- **Características Acelerómetro:**

- 3 Ejes
- Precisión: 16Bits
- Rango programable:  $\pm 2g$ ,  $\pm 4g$ ,  $\pm 8g$ ,  $\pm 16g$
- Filtro paso bajo programable.
- Consumo:  $500\mu A$  (en funcionamiento)
- Interrupciones programables
- Interfaz: I2C

- **Características Giroscopio.**

- 3 Ejes.
- Precisión: 16Bits
- Escala programable:  $\pm 250$ ,  $\pm 500$ ,  $\pm 1000$  y  $\pm 2000$  °/sec
- Filtro paso bajo programable.
- Intensidad  $3.6mA$  (en funcionamiento)  $5\mu A$  (standby)
- Calibrado de fábrica

- **Características adicionales**

- Bus I2C auxiliar para leer datos de otros sensores.
- Voltaje de funcionamiento  $3.3v$ .
- Velocidad de transmisión  $400kHz$  (Fast Mode I2C).
- Certificado RoHS.

- **GY-521 BreadBoard**

El chip MPU6050 (ver Figura 11) es un componente integrado minúsculo, lo cual lo hace muy atractivo para aplicaciones industriales, pero para el prototipado a nivel universitario es complicado su uso y manipulación, es por ello que habitualmente se venden ya montados en placas que hacen más accesibles los pins, además de añadir las resistencias pull-up necesarias en los buses I2C.

## 5. Herramientas matemáticas y teoría.



Figura 11: GY-521

La placa de prototipo GY-521 es la más extendida para el MPU6050, además de las resistencias pullUp añade un regulador de voltaje que nos permite alimentar el chip a 5v, algo muy útil si utilizamos microcontroladores con esa lógica.

Los Pins que nos proporciona la placa son los imprescindibles para el uso del chip:

- **Vcc:** +5v
- **GND:** tierra (0v)
- **SCL:** I2C clock
- **SDA:** I2C data
- **XDA y XCL:** bus I2C auxiliar
- **AD0:** para cambiar la dirección I2C del chip de 0x68 a 0x67
- **INT:** pin de interrupción

### - Configuración y uso:

El sensor, es realmente complejo y su descripción completa supera el objeto de esta sección, no obstante a continuación se describen los pasos necesarios para inicializar, configurar y recuperar medidas del mismo.

#### - Test de conexión

En la hoja de especificaciones del Chip indica como buena práctica, que en el proceso de inicialización se incluya una lectura al registro MPU6050\_WHO\_AM\_I (0x65) este registro siempre devuelve 0x68, es decir la dirección i2c del chip (independientemente de AD0), de modo que si existe algún error de conexión, este es un buen método para detectarlo en la inicialización.

### - Inicialización

Al recibir la alimentación el chip se encuentra en modo SLEEP, y para “despertarlo” debemos escribir en el registro MPU6050\_PWR\_MGMT\_1 (0x6b) un 0.

Una vez escrito el chip ya está listo para enviar medidas, por defecto el chip al inicializar:

- Rango del giroscopio +- 250°/s
- Rango del acelerómetro +2g
- Reloj a 8Mhz

### - Configuración de Rangos

Si consultamos la hoja de especificaciones del chip, se observa que la selección de rangos se hace en los registros 27 (giroscopio) y 28 (acelerómetro) (ver Tabla 2).

**Para el giróscopo, el registro 27 (0x1B):**

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1B	27	XG_ST	YG_ST	ZG_ST	FS_SEL[1:0]		-	-	-

*Tabla 2: Registro 27: configuración del giroscopio*

La configuración del rango se establece con los bits 3 y 4, con 2 bits tenemos 4 posibles configuraciones, que son las siguientes:

FS_SEL	Full Scale Range	LSB Sensitivity
0	± 250 °/s	131 LSB/°/s
1	± 500 °/s	65.5 LSB/°/s
2	± 1000 °/s	32.8 LSB/°/s
3	± 2000 °/s	16.4 LSB/°/s

*Tabla 3: Selección de Rango del giróscopo*

Es importante destacar la **columna LSB Sensitivity** (*Least Significant Bit*) nos indica, cuantos bits equivalen a 1 grado por segundo, por ejemplo para el rango +- 250°/s si el giróscopo nos devuelve un valor de 262 en cierto eje, y queremos saber a cuantos grados por segundo equivale, la manera de calcular es dividir el valor del acelerómetro entre la columna LSB sensitivity, generalizando, la fórmula para la conversión sería:

$$\text{Valor\_en\_grados\_por\_segundo} = \text{medida\_giroscopo} / \text{LSB\_sensitivity}$$



## 5. Herramientas matemáticas y teoría.

Es decir, que cada bit que incrementamos equivale a un incremento en  $^{\circ}/s$  de lo que indica la columna.

### Para el Acelerómetro, el registro 28 (0x1C):

Igual que en el caso del giroscopio la selección de rango esta en los bits 3 y 4:

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1C	28	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]				-

Tabla 4 : Registro 28: configuración para el acelerómetro.

Los 4 posibles rangos seleccionables y su ganancia correspondiente se pueden ver en la siguiente tabla 5:

AFS_SEL	Full Scale Range	LSB Sensitivity
0	$\pm 2g$	16384 LSB/g
1	$\pm 4g$	8192 LSB/g
2	$\pm 8g$	4096 LSB/g
3	$\pm 16g$	2048 LSB/g

Tabla 5 : Selección del rango del acelerómetro.

### - Adquisición de datos.

Cuando inicializamos el chip, este se encarga internamente de realizar las medidas y ponerlas a disposición en sus registros I2C, y la prioridad que tienen las peticiones I2C es la más alta, de modo que cada vez que solicitamos los datos, el chip interrumpe cualquier operación interna que esté realizando para proporcionarnos la medida.

Los registros que se manejan en el protocolo I2C (ver Figura 12) siempre son de 8 bits, de modo que al ser la precisión del sensor de 16 bits, las medidas se alojan en 2 registros. Como tenemos 3 ejes por 2 sensores tenemos un total de 12 registros que medir para recuperar las medidas del acelerómetro y del giróscopo. Los registros son los siguientes:

Nombre del Registro	Dirección	Significado
MPU6050_ACCEL_XOUT_H	3B	Acelerometro-EjeX-HighByte
MPU6050_ACCEL_XOUT_L	3C	Acelerometro-EjeX-LowByte
MPU6050_ACCEL_YOUT_H	3D	Acelerometro-EjeY-HighByte
MPU6050_ACCEL_YOUT_L	3E	Acelerometro-EjeY-LowByte
MPU6050_ACCEL_ZOUT_H	3F	Acelerometro-EjeZ-HighByte

MPU6050_ACCEL_ZOUT_L	40	Acelerometro-EjeZ-LowByte
MPU6050_GYRO_XOUT_H	43	Giroscopo-EjeX-HighByte
MPU6050_GYRO_XOUT_L	44	Giroscopo -EjeX-LowByte
MPU6050_GYRO_YOUT_H	45	Giroscopo -EjeY-HighByte
MPU6050_GYRO_YOUT_L	46	Giroscopo -EjeY-LowByte
MPU6050_GYRO_XOUT_H	47	Giroscopo -EjeZ-HighByte
MPU6050_GYRO_XOUT_L	48	Giroscopo -EjeZ-LowByte

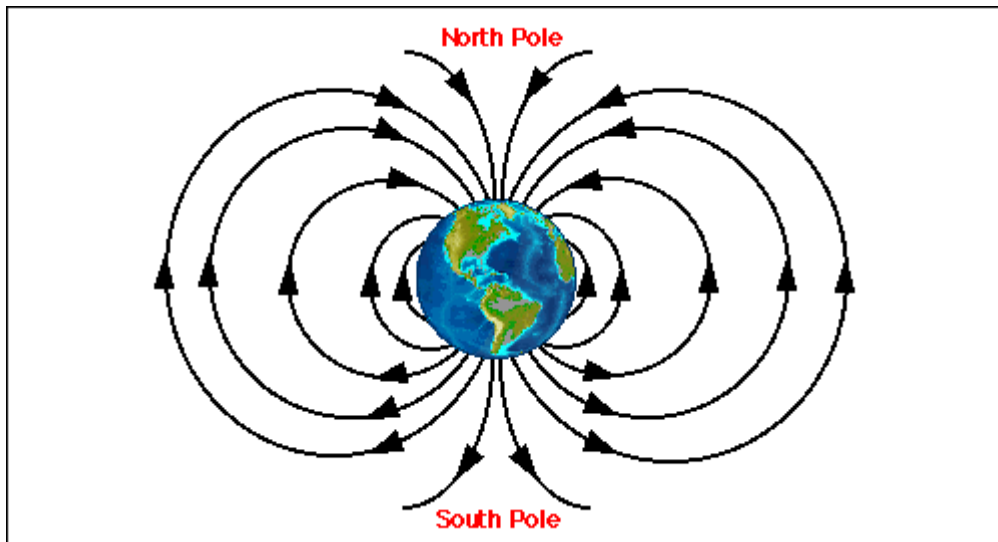
Figura 12: Registros de datos del MPU6050

#### 5.2.4. Magnetómetro.

Los magnetómetros son sensores que detectan campos magnéticos. Un campo magnético es un concepto que sirve para describir la influencia magnética que producen las corrientes eléctricas o los imanes permanentes (por ejemplo los de hierro). Se mide como un vector, y por tanto tiene dirección y magnitud. Su unidad en el sistema internacional es el Gauss.

- **Campo Magnético terrestre.**

## 5. Herramientas matemáticas y teoría.



*Figura 13: Líneas de Campo magnético de la tierra*

Ahora bien, ¿qué utilidad puede tener un dispositivo que mide la influencia magnética que producen cargas eléctricas en movimiento?, pues resulta que en la tierra crea su propio campo magnético debido a las corrientes eléctricas que se producen en su núcleo de hierro y níquel líquido. Como podemos ver en la Figura 13 el campo magnético de la tierra tiene unas líneas de campo con una trayectoria definida y conocida, salen del polo sur y se dirigen al polo norte.

Es importante destacar que en la superficie terrestre, si medimos el campo magnético, en el ecuador es paralelo a la superficie, en el polo norte su dirección es hacia abajo y en el polo sur hacia arriba.

La magnitud del campo terrestre en la superficie, varía según la zona, pero es alrededor de 0.5 Gauss.

### - Norte Magnético y Norte Geográfico.

Existe una diferencia entre lo que conocemos como norte (norte geográfico) y el norte magnético de la tierra, como se ve en la imagen, hay cierta desviación.

**Norte Geográfico:** También es conocido como norte verdadero, y coincide con el eje de rotación terrestre. Su posición no coincide exactamente con el norte geográfico y se mueve de año en año tanto que durante el siglo XX su posición varió 1100

kilómetros de distancia. Actualmente el norte magnético varía su posición unos 60 kilómetros por año.

**Norte Magnético:** El norte magnético está definido por el campo magnético de la tierra y es a donde apuntan las brújulas o compases náuticos.

**Declinación Magnética:** Esta diferencia/desviación entre el norte geográfico y el norte magnético se conoce como declinación magnética y se puede calcular con precisión (ver Figura 14). Es un concepto importante porque depende de qué aplicación estemos desarrollando con los magnetómetros nos será desde innecesaria hasta imprescindible.

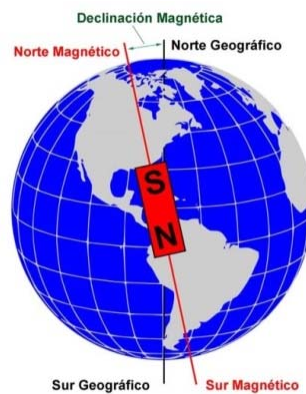


Figura 14: Declinación Magnética

- **Campo magnético como brújula.**

A lo largo de la historia del hombre se ha utilizado mucho el campo magnético terrestre para la orientación y la navegación, las brújulas tradicionales, hacen uso de este para indicar el norte.

En lo que respecta a los magnetómetros, el principio es el mismo, tenemos un dispositivo que es capaz de indicarnos un vector que apunta al norte. De modo que si alineamos el dispositivo con el suelo, seremos capaces de conocer dónde está el norte.

## 5. Herramientas matemáticas y teoría.

### - Magnetómetros para calcular rotaciones.

Al igual que los acelerómetros, que nos indicaban una referencia externa de donde está el “suelo” o centro de la tierra, los magnetómetros, nos ofrecen una referencia de donde está el norte, que podríamos llamar el eje X.

También adolecen del mismo problema que el acelerómetro pero para ejes diferentes, como la información que nos da es un vector de dirección en el eje X, no podemos conocer la rotación sobre ese eje.

A modo de ejemplo imagine que se está en el espacio sin ninguna referencia externa, y solo conoce que hacia donde usted está mirando es hacia el norte, usted, no puede saber si en el eje de coordenadas que haya establecido está de pie o boca abajo como indica la imagen, ya que solo conoce la dirección en la que mira. Esta analogía nos da a entender el alcance de la información que estamos obteniendo de un magnetómetro.



Esto es algo complicado de entender en primera instancia, pero se profundizara en el tema en la etapa de desarrollo del algoritmo.

### - Características de los magnetómetros

Las principales características de los magnetómetros son muy similares a las ya vistas para los acelerómetros y los giróscopos:

#### - Numero de ejes

Los magnetómetros más habituales son los de 3 ejes como en el caso de los giróscopos y los acelerómetros (ver Figura 15). Son los que se utilizan en este proyecto.

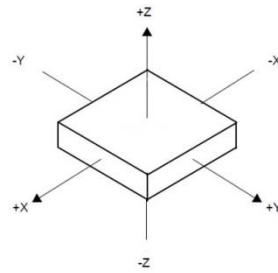


Figura 15: ejes de un magnetómetro

#### - Tecnología MEMS

Los magnetómetros más habituales, y los que se utilizan para el desarrollo de este proyecto se basan en la tecnología MEMs, por tanto son dispositivos económicos, de tamaño reducido y significativamente precisos respecto a otras soluciones con tecnologías más antiguas.

#### - Rango y precisión

En el caso de los magnetómetros es menos común encontrar modelos cuyo rango sea seleccionable, esto es debido a que principalmente los magnetómetros se utilizan para medir el campo terrestre y este en la superficie terrestre tiene siempre un valor de 0.5 Gauss y por tanto, los magnetómetros convencionales se diseñan para manejar un rango de valores alrededor de ese valor, con un margen suficiente para no saturar, en caso de encontrarse con influencias externas (grandes masas de hierro o similar)

#### - Interfaz

Las interfaces de los magnetómetros más habituales al igual que en los modelos de acelerómetro y giroscopio son: Analógica, Digital (I2C o SPI) o por Modulación de Ancho de Pulso (PWM).

### 5.2.5. Magnetómetro HMC5883L

El magnetómetro HMC5883L [22 [Honeywell.com](http://Honeywell.com)] (ver Figura 16) es un integrado diseñado para detectar pequeños campos magnéticos con una interfaz digital que utiliza el protocolo de comunicación I2C para aplicaciones del tipo brújula digital o magnetometría.

## 5. Herramientas matemáticas y teoría.



Figura 16: Chip HMC5883L

Se encuentra dentro de lo que se denomina sensores de alta resolución debido a su precisión de 2 mili Gauss. Incluye entre otras características adicionales auto cancelación del offset, su convertidor ADC es de 12 Bits por lo que se mueve en un rango de valores de 0 a 4096 bits entre valores equivalentes de  $\pm 8$  Gauss, esto equivale a una precisión entre 1 y 2 grados en aplicaciones del tipo brújula. Tiene un tamaño muy reducido de 3x3x0.9 mm dispone de 16 pines como se ve en la imagen.

### - **Características:**

- 3 Ejes
- Precisión: 12bits
- Rango  $\pm 8$  Gauss
- Consumo: 100 $\mu$ A
- Interfaz I2C 400KHz
- Velocidad de actualización 160 Hz
- Voltaje de funcionamiento 3.3v.
- Certificado RoHS.

### - **GY-271 BreadBoard**

El chip HMC5883L es un componente integrado minúsculo, lo cual lo hace muy atractivo para aplicaciones industriales, pero para el prototipo a nivel universitario y de investigación es complicado su uso y manipulación, es por ello que habitualmente se venden ya montados en placas que hacen más accesibles los pins, además de añadir las resistencias pull-up necesarias en los buses I2C.

La placa de prototipo GY-271 (ver Figura 17) es la más común para este chip, además de las resistencias pull-Up añade un regulador de voltaje que nos

permite alimentar el chip a 5v, algo muy útil si utilizamos microcontroladores con ese voltaje de alimentación.

Los pines principales que proporciona la placa son los de alimentación (2) GND y +5Vcc, los de comunicación I2C, (2) SCL y SDA, y uno adicional para interrupciones.

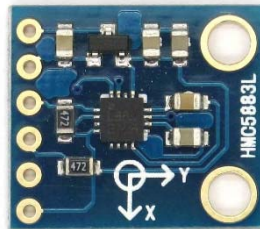


Figura 17 : GY-271

#### - **Configuración y uso.**

El chip es algo más sencillo que el MPU6050 que vimos anteriormente, tiene muchos menos registros, en el presente documento se describen los pasos necesarios para inicializar, configurar y recuperar medidas del mismo.

La dirección I2C del chip es **0x3C**, y a través de ella y de los números de registro podemos acceder a toda su funcionalidad

#### - **Modo de operación.**

El HMC5883L tiene 3 modos de operación, medidas continuas, única medida y modo idle, el propósito principal de distinguirlos es el consumo eléctrico:

**Modo medidas continuas:** el sensor toma medidas continuamente a una frecuencia seleccionable y los pone a disposición del usuario en los registros correspondientes, si el controlador no recoge la medida en el momento, la siguiente medida sobrescribirá a la anterior, de modo que el controlador siempre recibirá última medida que el sensor ha realizado. Es importante recalcar que el sensor no ocupa el bus en ningún caso si el controlador no lee los registros correspondientes.

**Modo única medida:** en esta modalidad el sensor realiza una única medida y pone los datos a disposición en los registros correspondientes, y después de esto, el sensor pasa al modo idle (ocioso) hasta que el controlador solicite otra medida. Este modo es



## 5. Herramientas matemáticas y teoría.

más eficiente energéticamente ya que solo gasta energía para las medidas que va a utilizar, aunque también es más lento.

**Modo Idle:** es un modo de ahorro de energía en el que la mayor parte del chip esta deshabilitado con este fin. Los registros de datos están accesibles.

### 5.3. Microcontrolador.

Para procesar la información que proviene de los sensores va a ser necesario una unidad de procesamiento, lo suficientemente potente como para ofrecernos una **velocidad de cómputo** que soporte las especificaciones de tiempo real para la aplicación, y a la vez sea económico, ya que este proyecto es sustentado íntegramente por el estudiante universitario que lo escribe, y por último, que tenga un tamaño y peso muy reducidos para que no resulte incómodo.

#### 5.3.1. Características.

Para cubrir las necesidades del proyecto necesitamos elegir un microcontrolador con unas características muy concretas:

**Tamaño y peso:** El dispositivo debe ser lo más pequeño posible ya que van a tener que ir incrustados en módulos que llevarán “puestos” los usuarios en cada articulación.

**Comunicación:** El microcontrolador debe tener mínimo integrado protocolo I2C y puerto serie, esto se puede implementar con otros chips externos, pero la necesidad de construir un dispositivo lo más pequeño posible nos obliga a buscar una arquitectura que ya lo integre.

**Velocidad:** Al ser una aplicación en tiempo real tenemos que encontrar la plataforma más rápida que podamos encontrar. Las velocidades más habituales para micro controladores embebidos son 4Mhz 8Mhz 16Mhz 48Mhz 70Mhz y 80Mhz.

**Arquitectura:** La frecuencia de reloj del microcontrolador es muy importante en la velocidad que ejecuta operaciones, pero la arquitectura lo es como mínimo igual de importante, la mayoría de micros actuales son de 8bits, pero existen algunos muy interesantes de 32Bits.

**Precio:** El dispositivo no puede superar los 100€ de presupuesto y dado se necesitan un mínimo de 9 micro controladores y otros tantos sensores, hay que tener muy en cuenta el precio unitario de cada microcontrolador.

### 5.3.2. Modelos Considerados

Siguiendo los anteriores criterios las mejores opciones que existen actualmente en el mercado son las siguientes:

- **Arduino Pro Mini (Atmel 328p 16Mhz):**

[24, Arduino.cc] (ver Figura 18) Es un microcontrolador a 16Mhz con una arquitectura de 8Bit y un precio unitario de 1.5€ en china, con una extensa comunidad, es una plataforma Open Source que se basa en Chips Atmel, realmente Arduino da el nombre a lo que se conoce como bootloader que es un firmware que viene cargado en el chip de Atmel que hace de capa intermedia que nos abstrae de la arquitectura del chip, y hace que podamos programarlo en C++.

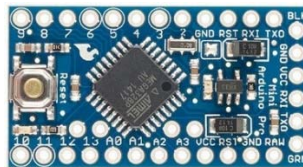


Figura 18 : Arduino Pro Mini 5v  
16Mhz

- **ESP8266:**

[25, ESP8266.com] Es la nueva revolución en micro controladores, debido a su bajo coste 2€ en china, el ESP8266 (ver Figura 19) funciona a 80MHz con una arquitectura de 32bits, es sin lugar a duda el chip más puntero del mercado en relación a su precio, como puntos flacos tiene, que se programa en Lua, un lenguaje no muy común, además de que al ser bastante novedoso, existe muy poca comunidad en comparación con Arduino.



## 5. Herramientas matemáticas y teoría.

*Figura 19 : ESP8266*

### Teensy 3.1: (ver Figura 20)

[26, pjrc.com] Es una muy buena plataforma basada en Arduino y funciona a 70 MHz con una arquitectura de 32 bits, su mayor problema es el precio llegando a costar 20€ por unidad.



*Figura 20 : Teensy 3.1*

El criterio que más ha pesado en la elección del microcontrolador ha sido el precio, ya que para un sistema completo necesitaremos un mínimo de 10 microcontroladores con sus correspondientes sensores. Por lo tanto se ha escogido el **Arduino pro mini 16Mhz** como la plataforma para el proyecto.

## 5.4. Protocolos de comunicación.

El dispositivo, es en sí mismo una red de nodos y sensores jerárquica en varios niveles, y con varios protocolos de comunicación implementados en sus diferentes niveles, es por tanto necesario comprender como funciona cada uno y como hacer uso de ellos.

### 5.4.1. Puerto Serie RS-232.

Es un protocolo de comunicación muy común entre dispositivos, los PC actuales ya no los traen de serie desde la aparición del puerto USB, pero existen multitud de conversores Puerto-serie->USB que hacen que este protocolo de comunicación este a la orden del día. El conector más utilizado para conectar los dispositivos es el DB9 (ver Figura 21) aunque existen otros.



Figura 21: Conector DB9, para puerto serie

Como su propio nombre indica la comunicación es serial, es decir un bit detrás de otro y al utilizar 2 cables para comunicación (tx y rx) la comunicación es **full-duplex**. Existen más líneas con diferentes propósitos, la línea de tierra (GND o referencia), y aunque no es necesario también dispone de líneas para el control de flujo.

Los parámetros más importantes de la comunicación serie 232 son:

- **BaudRate:** que equivale a la velocidad de transmisión, Indica el número de bits por segundo que se transfieren, y se mide en baudios (*bauds*), el protocolo acepta multitud de velocidades siendo las más habituales 9600, 115200 y 250000 baudios.
- **Bits de datos:** Se refiere a la unidad de transmisión, es decir a la cantidad de bits que se mandan en cada transmisión, por ejemplo, si transmitimos caracteres ASCII, (128 posibilidades) se transmiten de 7 en 7.
- **Paridad:** es un método de detección de errores en la transmisión, los valores habituales son par, impar, marcada, espaciada.
- **Bits de parada:** indica el fin de la comunicación de un solo paquete. Suelen ser 1, 1.5 o 2 bits, también son útiles para la sincronización.

- **Puerto Serie Arduino.**

Algunas tarjetas Arduino traen un conector USB para la conexión directa con el ordenador, pero en el caso del Arduino pro mini 16MHz debido a su reducido tamaño, únicamente tiene disponibles los pines para su conexión con un conversor puerto serie

## 5. Herramientas matemáticas y teoría.

(niveles TTL) a USB. En este caso se ha utilizado el cable llamado TTL-232R-5V (ver Figura 22) de la marca FTDI.



*Figura 22 : Cable FTDI TTL-232R-5V*

### 5.4.2. Bluetooth

El bluetooth es un protocolo de comunicación que define una red inalámbrica del tipo WPAN que trabaja por radiofrecuencia en la banda ISM de los 2,4 GHz.

Se ha elegido esta tecnología para implementar la comunicación inalámbrica con el PC debido a que en la comunidad Arduino existe mucha documentación y a su bajo coste económico. Además casi cualquier Pc actual dispone de un módulo bluetooth integrado y en el caso de no tenerlo, este se puede encontrar por menos de 2 € en cualquier tienda.

#### - **Hc-05 Modulo Bluetooth**

[14c, Prometec.com] Para el proyecto se ha elegido este módulo ya que es el más utilizado para prototipos y en la comunidad de Arduino existen muchos tutoriales y guías para su configuración y uso. Se puede ver este módulo en la Figura 23.



*Figura 23 : Hc-05 modulo Bluetooth*

El modulo se configura mediante puerto serie con el host por comandos AT y una vez configurado establece una conexión puente con el host que se hace transparente al microcontrolador, haciendo que a todos los efectos funcione como una conexión a puerto serie cableada.

### - Pinout

El modulo dispone únicamente de 6 pines:

- **Alimentación 5v:** habitualmente se distribuyen con reguladores integrados en placa para alimentarlos a 5v porque el integrado principal se alimenta a 3v.
- **GND:** tierra.
- **Tx:** pin de transmisión
- **Rx:** pin de recepción.
- **WakeUp:** Pin para cambiar de modo comandos AT o conexión Host
- **State:** indicador de conexión establecida con el host

### - Funcionamiento.

Una vez alimentado con los 5v, el modulo comprueba el pin WakeUp, si está activo, el modulo inicia en modo comandos AT, en este modo podemos definir los parámetros de conexión del módulo tales como:

- Velocidad de transmisión (Baudrate)
- StopBits
- Paridad
- Seguridad: contraseña (passPhrase), módulos aceptados ...
- Método de conexión
- Visibilidad
- Nombre modulo
- ....

En la Figura 24 se puede observar un ejemplo de cómo se configura el modulo mediante un terminal conectando el modulo al pc por puerto serie:

## 5. Herramientas matemáticas y teoría.

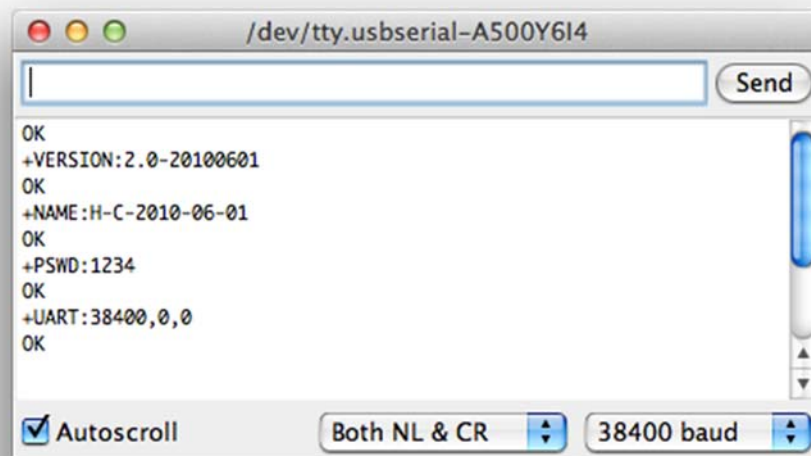


Figura 24: comandos AT al módulo Bluetooth Hc-05

### 5.4.3. Protocolo I<sup>2</sup>C.

Es un bus de datos que utiliza nada más que 3 líneas y comunicación serial, con lo que nos proporciona una manera de interconectar varios dispositivos sin demasiados cables, pero teniendo en cuenta que al ser un bus, solo pueden comunicarse 2 dispositivos a la vez.

Su velocidad estándar es de 100 kbits/s pero la mayoría de dispositivos actuales implementan una versión llamada Fast I2C que alcanza los 400 kbits/s.

Su funcionamiento es sencillo, tenemos 1 línea de datos 1 línea de reloj y la tierra o GND como referencia para los niveles bajos, la línea de reloj SCL nos sirve para ajustar la velocidad de transmisión y que todos los dispositivos estén sincronizados, y la línea de datos (SDA) se encarga del direccionamiento y de los datos. El direccionamiento, funciona de una manera muy sencilla con 8-Bits, 7 de ellos para dirección y uno para seleccionar modo escritura o lectura, cada dispositivo conectado al bus debe tener su dirección i2c única, de modo que el nodo maestro cuando solicita datos a esa dirección, bloquea el bus para la comunicación entre ellos.

La comunicación siempre se produce bajo el paradigma Maestro-esclavo, existiendo únicamente un dispositivo maestro y tantos esclavos como direcciones haya es decir para 7 bits de direccionamiento existen 128 posibles direcciones pero solo son útiles 112 ya que 16 de ellas están reservadas.

A nivel de programación la comunicación es muy sencilla ya que lo más habitual es que los dispositivos, pongan a disposición del maestro la información en diferentes registros ya conocidos por ambos de modo que se accede a ellos según la dirección del registro que se desea consultar, el acceso a la información es muy similar a como si se tratara de memorias EEPROM, aunque exista un dispositivo externo que actualiza esa información, para el maestro no es más que un acceso de lectura o escritura a un registro (1 byte).



# 6. Montaje del prototipo y adquisición de datos.

En esta etapa lo primero que buscamos es conectar el microcontrolador a los sensores y al ordenador para comprobar la comunicación entre los componentes, la comunicación y su correcta conexión.

Para esta primera iteración del proyecto vamos a utilizar los siguientes componentes Hardware (ver Figura 25):

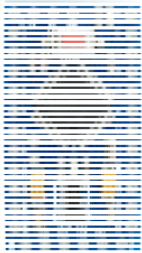

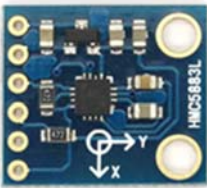
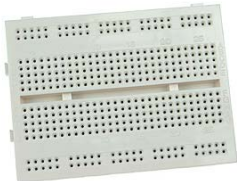
			
Arduino Pro Mini 5v 16Mhz	MPU6050 GY-521	HMC5883L GY-271	Placa de prototipos ProtoBoard

Figura 25 : Componentes necesarios para montaje del nodo prototipo

Siguiendo el las hojas de especificaciones vamos a conectar los componentes según el siguiente esquema realizado con fritzing, una herramienta Open Source para diseño de circuitos muy útil. El diagrama resultante se puede ver en la Figura 26.

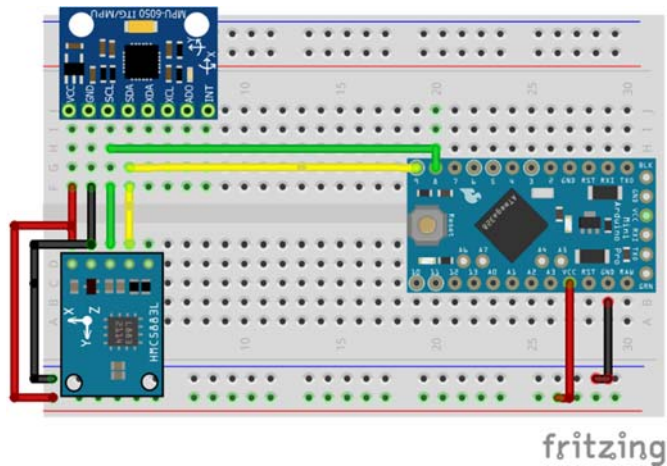


Figura 26: Diseño del prototipo de nodo en fritzing

## 6.1. Hardware.

En este punto del desarrollo es conveniente hacer la conexión de los componentes en una protoboard, que son placas de plástico perforadas para el desarrollo de prototipos y que puedan ser reutilizados, como puede verse en la Figura 27.

Cuando el nodo este correctamente funcionando se diseñara un circuito impreso lo más pequeño posible para su presentación final.

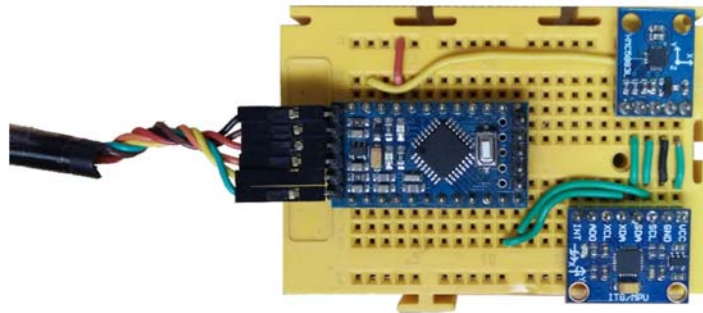


Figura 27: Prototipo en protoboard de nodo esclavo

El conexionado de los componentes es muy sencillo, es el que se muestra en la Tabla 6:

Arduino Pro Mini	MPU6050	HMC5883L
VCC 5v	VCC 5v	VCC 5v
GND	GND	GND
Digital Pin 9	SDA	SDA
Digital Pin 8	SCL	SCL

Tabla 6: Conexionado de los pines de los componentes

## 6.2. Módulos Software para Nodo Esclavo:

A continuación vamos a diseñar los módulos software necesarios para recuperar la información de los sensores y mandarla por puerto serie al ordenador.

La plataforma Arduino nos permite la programación en el lenguaje C++, el cual es orientado a objetos, de modo, que utilizaremos diagramas UML para modelar los componentes necesarios.

## 6. Montaje del prototipo y adquisición de datos.

La **clase MPU6050**, representa este dispositivo y en ella vamos a implementar los métodos necesarios para recuperar los datos en bruto, un protocolo de calibración, y la conversión de los datos de Bits a fuerza G y a grados por segundo.

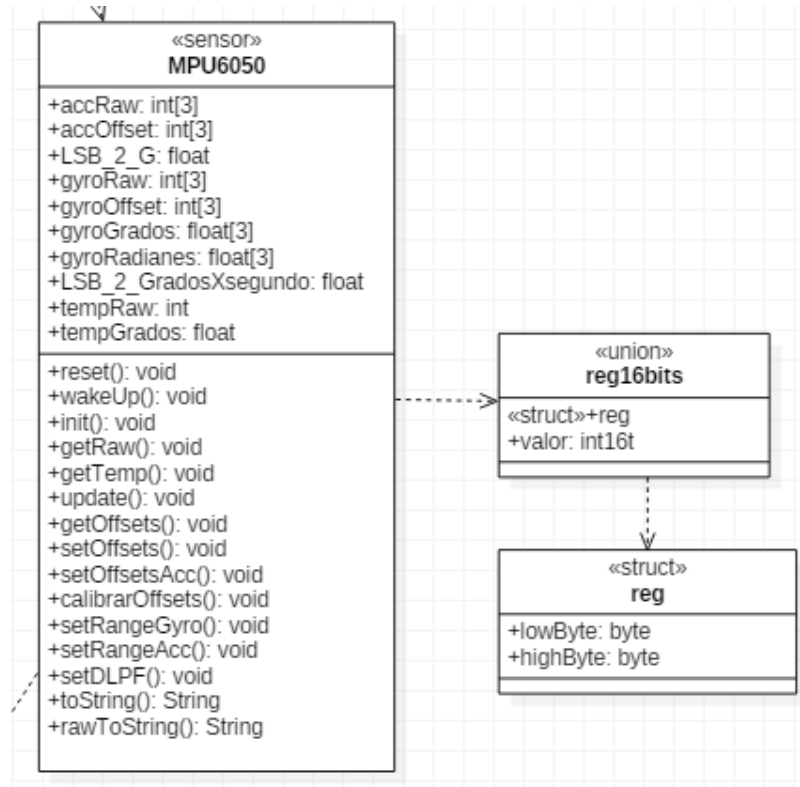


Figura 28 : Diagrama UML de clase para el módulo MPU6050

En la figura 28 podemos ver el diagrama UML de la clase que gestiona todo lo relacionado con el sensor, desde la comunicación con este por el bus I2C hasta el procesamiento de los datos en brutos para pasarlos a magnitudes físicas, el proceso se describió en la sección “3.2.3. MPU-6050: Acelerómetro + Giróscopo.”, a continuación se muestra una breve descripción de los métodos más relevantes.

- **Reset (), wakeUp(), Test() e init()** : son los métodos necesarios para el proceso de inicialización del módulo.
- **getRaw()** : es el encargado de comunicarse con el sensor y recuperar las medidas del conversor ADC, guarda el resultado en los atributos accRaw y gyroRaw, que son arrays de tamaño 3, uno para cada eje del acelerómetro y del giróscopo.

- **Update():** este método, se encarga de la conversión de los datos en bruto (accRaw[3] y gyroRaw[3]) a su equivalente en magnitudes físicas.
- **CalibrarOffsets():** inicia el proceso de calibración explicado en la sección 3.2.3.

La clase **HMC5883L**, cuyo diagrama UML podemos ver en la Figura 29, representa al magnetómetro con el mismo nombre y al igual que la clase anterior para el MPU, esta implementa los métodos necesarios para:

- Adquisición de datos del sensor.
- Procesado y conversión a magnitudes físicas (mili Gauss)
- Proceso de calibración.

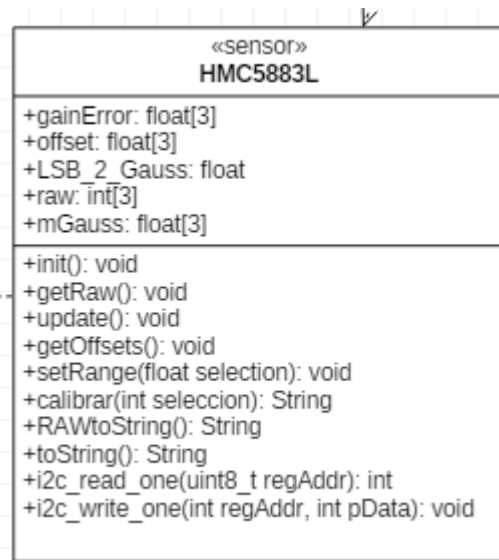


Figura 29 : Diagrama UML de clase para el modulo del magnetómetro

Como se puede observar en el diagrama UML, también se incluyen otros métodos para devolver los datos procesados como strings y los correspondientes a la comunicación I2C, a continuación se muestra una breve descripción de los métodos más relevantes.

- **init()** : es el método que gestiona la inicialización del modulo

## 6. Montaje del prototipo y adquisición de datos.

- **getRaw()** : es el encargado de comunicarse con el sensor y recuperar las medidas del conversor ADC, guarda el resultado en los atributos raw[3], que es un array de tamaño 3, uno para cada eje.
- **Update()**: este método, se encarga de la conversión de los datos en bruto (raw[3]) a su equivalente en magnitudes físicas mili Gauss en la variable mGauss[3].
- **CalibrarOffsets()**: inicia el proceso de calibración.

### 6.3. Aplicación de pruebas: sensoresEnBruto.py

Con el fin de poder probar la funcionalidad del módulo software se ha programado un pequeño sketch de Arduino que recibe comandos por puerto serie y devuelve los valores recibidos por los sensores, así se puede comprobar por pantalla si el código se comunica correctamente con los sensores y recupera la información, podemos ver el resultado por pantalla en un terminal de puerto serie como muestran las Figuras 30 y 31, cada vez que se manda un carácter 'c' al módulo devuelve una línea con los 9 valores separados por comas que representan en este orden (accX,accY,accZ,gyroX,gyroY,gyroZ,magX,magY,magZ) los valores en bruto de los sensores, es decir los que vienen directamente del conversor ADC del sensor, y si mandamos el comando 'b' (Figura 31) recibiremos los datos procesados y convertidos a grados por segundo para el giróscopo, fuerzas g para el acelerómetro y mili Gauss para el magnetómetro, ya calibradas y corregidas las desviaciones.

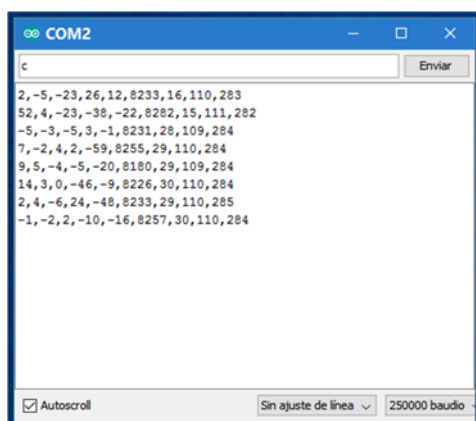


Figura 31: Terminal Valores en bruto (CMD 'c')

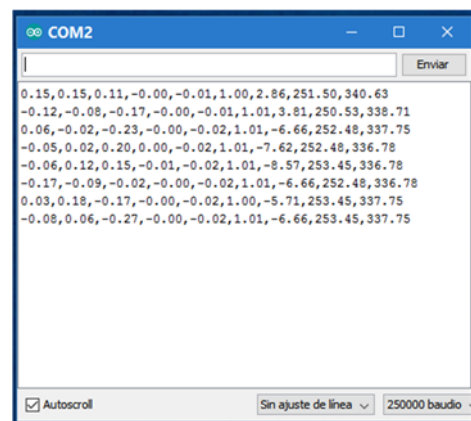


Figura 30: Terminal Valores Físicos (CMD 'b')

No obstante como la visualización de los datos son vectores de 3 coordenadas se ha desarrollado una aplicación de escritorio para probar esta funcionalidad, que pinta por pantalla en un eje de coordenadas los vectores en forma de flecha, como se puede observar en la imagen de abajo, la aplicación está escrita en Python y se puede encontrar en la carpeta vPython/sensoresEnBruto. Es una aplicación sencilla que va mandando el comando 'a' por el puerto serie al nodo para recibir los 9 valores de los 3 sensores y pintarlos como vectores en un sistema de coordenadas.

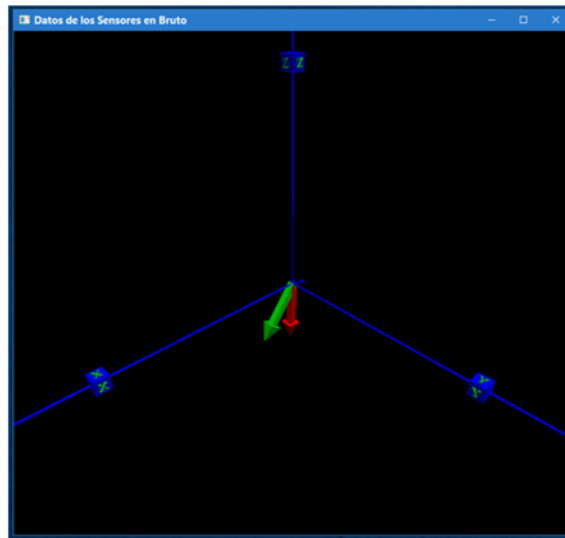


Figura 7: App de demostración "Sensores en bruto"

En la Figura 7 se puede observar una flecha roja, que corresponde al acelerómetro, podemos comprobar que los valores son correctos ya que estando el prototipo encima de la mesa estático, la directamente hacia el suelo. La representación de los valores del magnetómetro son la flecha verde, y podemos observar que apunta en línea al eje X y tiene cierta inclinación hacia abajo, esto es coherente con lo que se explica en el apartado 4.2.4, en la prueba se ha utilizado una brújula convencional para orientar el eje x del magnetómetro con el norte, de modo que lo que esperamos del magnetómetro es un vector que apunte hacia el eje X, la inclinación hacia abajo se debe como se explicó también en la sección dedicada a los magnetómetros, que las líneas de campo en el hemisferio norte (España), apuntan hacia abajo y al norte. Y por último el giróscopo no muestra ningún valor debido a que el dispositivo tiene velocidad angular 0 y por tanto el vector correspondiente esperado es (0,0,0).

## 6. Montaje del prototipo y adquisición de datos.

La implementación en código se encuentra en el **código fuente** adjunto a este documento.





# 7. Técnicas de filtrado y fusión.

Esta etapa es, quizás, la parte más importante y compleja del proyecto, ya que los sensores de por sí no representan orientación ninguna, solo nos ofrecen información de ciertas magnitudes físicas, aceleración, velocidad angular y campo magnético, y es en esta etapa en la que hacemos uso de estas referencias para **inferir la posición en la que se encuentra el dispositivo**.

Como veremos en este apartado, cada sensor además es propenso a ciertas interferencias e introducen ruido al sistema y es nuestra responsabilidad el saber lidiar con ellas e intentar sacarlas del sistema para que la señal de salida sea fiable y útil. A esta tarea se la llama **filtrado de señales** y no es en absoluto una tarea trivial, existe toda una rama de la computación llamada teoría de señales que se encarga de tratar estos temas, en este proyecto se hará uso de los filtros paso bajo, paso alto, y una combinación de ambos llamada filtro complementario.

La implementación en código c++ para Arduino y el montaje hardware, es original del autor de este documento, pero la teoría y la técnica de filtro complementario han sido extraídas del artículo [27, Sergiu Baluta].

## 7.1.1. Estimación de la orientación con acelerómetro y magnetómetro.

En primer lugar vamos a definir los ejes de coordenadas, para sistema fijo utilizaremos las letras mayúsculas IJK y para el sistema local móvil, asociado al dispositivo, utilizaremos las minúsculas ijk.

Como vemos en la imagen IJK equivaldría a (norte, Oeste, cielo)

## Representación de la orientación de un sistema articulado

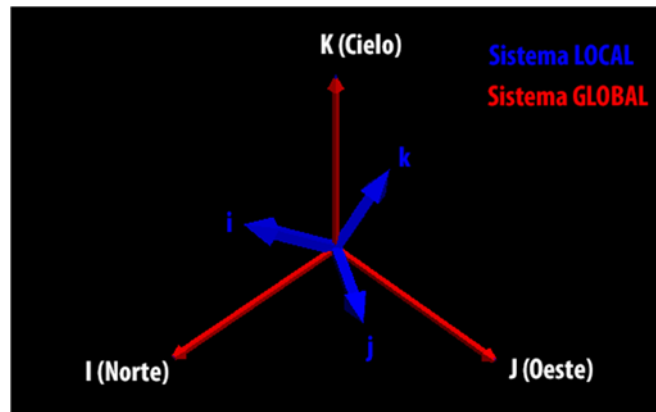


Figura 32 : Sistema Local y sistema Global

En la Figura 32 podemos ver el eje de coordenadas global en rojo y el local en azul, en este ejemplo, el sistema Local esta rotado  $60^\circ$  sobre el eje I y  $60^\circ$  sobre el eje J.

Como sabemos la gravedad siempre apunta al centro de la tierra, en nuestro sistema de coordenadas Global, equivaldría al vector de dirección  $(0, 0, -1)$ , en la siguiente imagen lo vemos representado con una flecha amarilla.

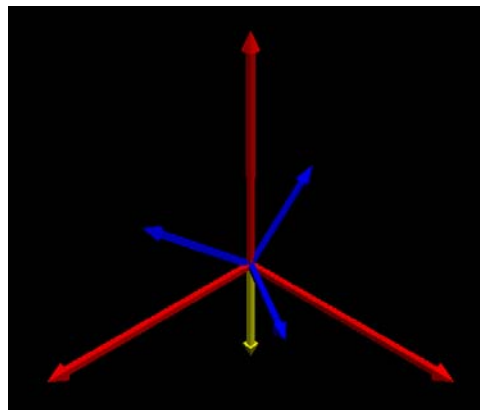


Figura 33: Gravedad  $(0, 0, -1)$  medida en Sistema Global

Pero el acelerómetro nos devuelve el vector gravedad medido desde el sistema de coordenadas LOCAL, como podemos ver en la siguiente imagen, en la que aparece el sistema LOCAL centrado, y la gravedad en este sistema tiene otras componentes diferentes.

## 7. Técnicas de filtrado y fusión.

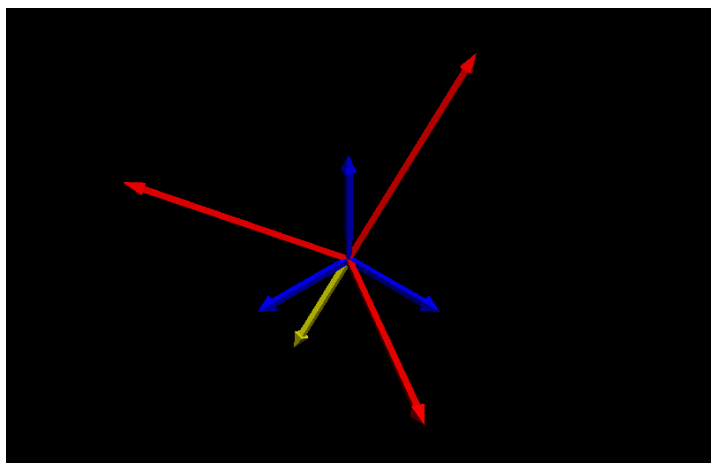


Figura 34: Gravedad medida en sistema Local (acelerómetro)

En este caso vemos que la gravedad desde este sistema de referencia no es  $(0, 0, -1)$ , sino que tiene una inclinación, y esto es lo que el acelerómetro nos proporciona, lo interesante es que el valor de la gravedad sigue estando alineado con el vector K del eje Global, esto es evidente, pero es interesante destacarlo, porque teniendo el valor de la gravedad (proporcionado por el acelerómetro) tenemos una referencia con el sistema de coordenadas Global. El **vector K medido en sistema Local** es igual al valor que nos proporciona el **acelerómetro multiplicado por  $(-1)$** .

A continuación vamos con las medidas del magnetómetro, el magnetómetro tiene un problema, que no está alineado exactamente con el eje I, ya que como pudimos comprobar en el apartado “6.3 Aplicaciones de pruebas: Sensores en bruto” apunta al norte hacia el suelo debido a que se alinea con las líneas de campo y en el hemisferio norte apuntan hacia abajo, existe un buen “truco” que nos proporciona el cálculo vectorial para solucionar esto.

En la siguiente imagen podemos ver 2 flechas, la roja para el acelerómetro y la azul para el magnetómetro, si calculáramos el producto vectorial de las 2, nos proporcionaría un vector alineado perfectamente con el eje J es decir, el oeste. Es muy importante el orden del producto (Magnetómetro \* Acelerómetro) ya que de otro modo no sería correcto el cálculo.

Flecha	Descripción
Roja	Gravedad, acelerómetro
Azul	Norte Magnético, Magnetómetro
Verde	OESTE: Producto Vectorial (magnetómetro , acelerómetro)

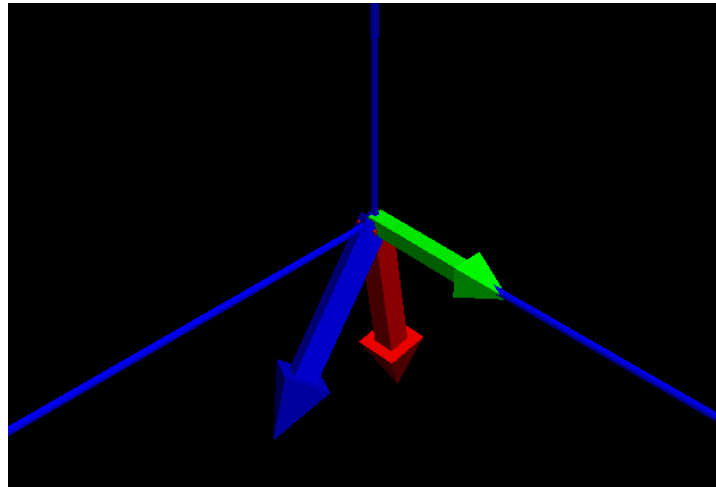


Figura 35: Producto Vectorial magnetómetro y acelerómetro (OESTE)

En este punto **ya tenemos las coordenadas de eje K (Cielo) y del eje J (Oeste) medidas en sistema Local**, pero el eje I, seguimos teniendo el problema de que el magnetómetro apunta hacia abajo, bien, pues si ahora volvemos a multiplicar vectorialmente el vector gravedad del acelerómetro, por el vector Oeste que acabamos de calcular, voila, ahora tenemos el vector I, siempre medidos desde el sistema de referencia Local. Podemos observarlo gráficamente en la imagen de abajo, y una vez más es crucial que el orden de la multiplicación vectorial sea el correcto (gravedad X oeste).

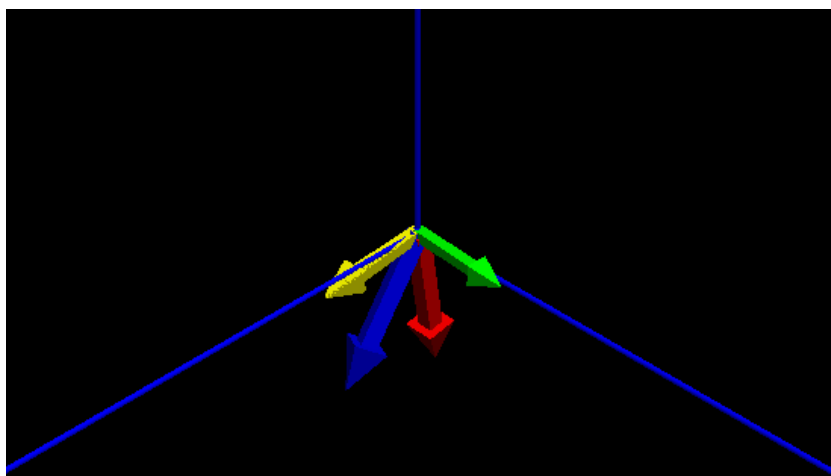


Figura 36: Producto vectorial Acelerómetro y Oeste (amarillo)

## 7. Técnicas de filtrado y fusión.

Por último, estamos utilizando todo el rato la gravedad como referencia, pero en nuestro eje de coordenadas, el vector K apunta en dirección contraria, por tanto, para calcular K tenemos que multiplicar el vector gravedad por -1 y ya tenemos el vector K.

Recapitulando, ya tenemos una manera de calcular todos los vectores del sistema de referencia IJK (Global) representados desde el sistema Local ijk.

Vector	Método para calcularlo
K	Vector gravedad del acelerómetro * (-1)
J	Acelerómetro X Magnetómetro
I	Acelerómetro X vector_J

Tabla 7: Calculo de sistema Global a partir de datos de los sensores

Con estos datos podemos **construir la matriz de rotación** (sistema local) de una manera muy sencilla:

$$DCM_{LOCAL} \begin{bmatrix} I_x & J_x & K_x \\ I_y & J_y & K_y \\ I_z & J_z & K_z \end{bmatrix}$$

Gracias a las propiedades de las matrices de rotación, el cambio de sistema de coordenadas que queremos hacer, se realiza simplemente calculando la matriz traspuesta a la DCM local. Es decir, la Matriz de rotación ya calculada serial:

$$DCM_{GLOBAL} = \begin{bmatrix} I_x & I_y & I_z \\ J_x & J_y & J_z \\ K_x & K_y & K_z \end{bmatrix}$$

Con esta información, ya podemos representar la rotación del dispositivo, simplemente obteniendo los valores del acelerómetro y del magnetómetro, el algoritmo sería:

- Obtener datos del magnetómetro (mx,my,mz) y del acelerómetro(ax,ay,az).
- Vector  $K_{Local} = acc * -1$
- Vector  $J_{Local} = magnetómetro \times acelerómetro$  (prod.vectorial)
- Vector  $I_{Local} = acc \times J_{Local}$  (prod.vectorial)
- Construir la Matriz de rotación según la ecuación anterior.

Solo utilizando estos cálculos, ya podemos representar la orientación del dispositivo, pero esta técnica, no hace uso del giroscopo, que es el sensor más preciso que tenemos y no realiza ningún filtrado de la señal de ningún tipo y por lo tanto tiene una parte importante de ruido que hace que las medidas que obtenemos no sean demasiado precisas.

### 7.1.2. Matriz de Rotación con giroscopio.

El giroscopo como hemos visto anteriormente nos proporciona un vector de velocidad angular, cada componente representa el giro en cada eje. Al trabajar con velocidades, tenemos que tener en cuenta el tiempo, ya que la velocidad angular no nos dice que orientación tiene un cuerpo sino, cuanto ha girado desde el momento anterior. Vamos a construir una DCM en función del tiempo, de modo que denotaremos a la DCM en el momento  $t$  como  $DCM(t)$  y la después de un incremento de tiempo  $dt$  será  $DCM(t+dt)$ :

Es importante destacar que continuamos calculando en sistema local, ya que el acelerómetro nos proporciona las medidas en ese sistema de referencia.

Para calcular la nueva DCM vamos a ir calculando la nueva posición de cada vector aplicando las ecuaciones de la velocidad angular.

En la imagen de abajo podemos ver como se resolvería para el vector  $J$ , los otros 2 vectores sería exactamente igual.

## 7. Técnicas de filtrado y fusión.

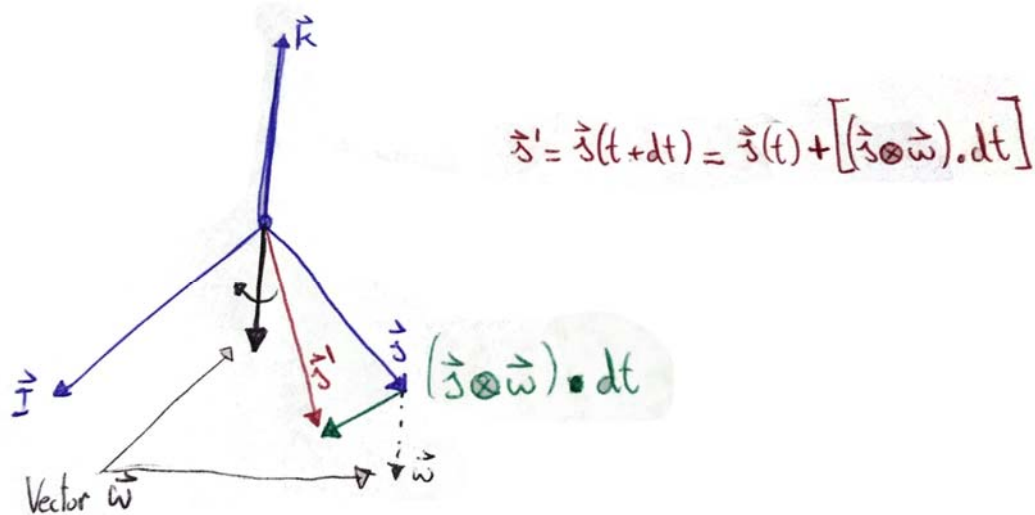


Figura 37: cálculo de  $DCM(t+dt)$  a partir de la velocidad angular  $w$

### 7.1.3. Velocidad Angular a partir del acelerómetro y magnetómetro.

Para **poder fusionar** los datos que nos proporcionan el giróscopo, el acelerómetro y el magnetómetro, tenemos que adaptar nuestros cálculos, para obtener de los 3 sensores la misma magnitud, y esa magnitud va a ser la velocidad angular.

La velocidad angular es la magnitud que nos proporciona directamente el **giróscopo**, por tanto no es necesario realizar ningún cálculo adicional.

El acelerómetro como vimos en el primer punto de esta sección está directamente relacionado con el eje K ( $Acc*(-1)$ ), y es por ello que vamos a estimar la velocidad angular a partir de este.

Si tenemos la posición del vector K en el momento t, que denominaremos  $K(t)$ , como una posición conocida calculada con anterioridad, y  $K(t+dt)$  que es la posición actual del vector medida del valor del acelerómetro (multiplicado por -1), calcular la velocidad angular es justo la operación contraria a la que realizamos con el giróscopo, se puede ver gráficamente en la siguiente imagen:

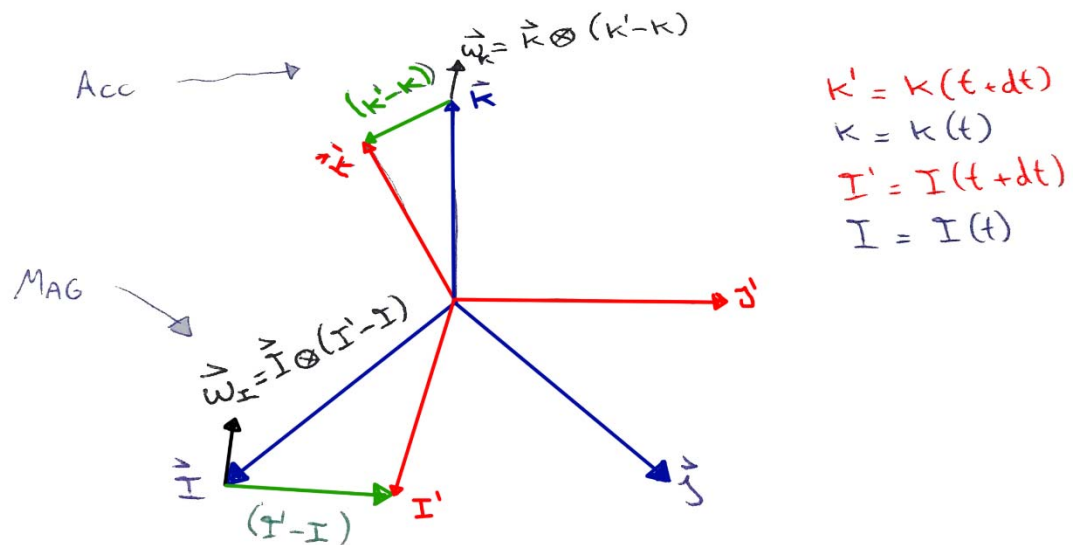


Figura 38: Velocidad angular a partir de los dos vectores desplazados

En la imagen se puede también apreciar cómo se realiza el cálculo correspondiente al vector  $\vec{i}$  de la misma manera.

Hay que tener en cuenta que aunque el cálculo se realiza para los vectores  $\vec{i}$  y  $\vec{k}$  según magnetómetro y acelerómetro respectivamente, como la magnitud a la que llegamos es velocidad angular es general para todo el sistema.

#### 7.1.4. Filtro complementario.

Ahora que ya tenemos 3 estimaciones basadas en 3 sensores diferentes (acelerómetro, magnetómetro y giroscopo) y las 3 están expresadas en la misma magnitud, la velocidad angular, tenemos que juntarlas, para obtener una estimación lo más fiable posible. La primera impresión podría ser sumar las componentes de los 3 sensores y dividir entre 3, es decir hallar la media de las 3 medidas, y esto sería correcto, pero no estaríamos explotando correctamente el conocimiento que tenemos de los sensores.

El giroscopo es un sensor muy preciso y rápido, es de los 3 el que realiza medidas más fiables, pero tiene el problema de que al no ser medidas absolutas, cada medida depende de la anterior, y por tanto cada mínima desviación de la medida, se acumula,



## 7. Técnicas de filtrado y fusión.

haciendo que al tiempo ese cumulo de error (llamado drift en la bibliografía) sea inaceptable

Por otra parte tenemos el acelerómetro, cuyas medidas son muy propensas al ruido, debido a que cuando el cuerpo cambia de posición, generalmente sufre una aceleración, y esta es medida por el acelerómetro, haciendo que se mezcle con nuestra referencia que es la gravedad, pero proporciona una referencia absoluta que no acumula error, cada medida es independiente de la anterior.

Por último el magnetómetro es un sensor con menos ruido que el acelerómetro, pero menos preciso, y también proporciona una medida absoluta, con lo que una medida es independiente de la anterior. La mayor fuente de interferencias para un magnetómetro son los dispositivos electrónicos y las grandes concentraciones de hierro, que alteran el campo magnético.

Es por esto que se va a utilizar un filtro complementario, o media ponderada, que es una variante de calcular la media de los 3 sensores, se trata de asignar un porcentaje de peso a cada sensor, de modo, que por ejemplo la medida final es el 10% proporcionada por el acelerómetro, 15% por el magnetómetro, y 75% por el giróscopo, de esta manera estamos fiándonos en gran medida del giróscopo, y el acelerómetro y el magnetómetro no nos introducen su ruido en tan alta medida, pero nos proporcionan esa referencia absoluta que corrige al giróscopo en espacios largos de tiempo y sobre todo cuando el dispositivo se encuentra parado.

La fórmula para el filtro complementario generalizada sería:

$$\frac{(\text{Medida}_1 * \text{peso1}) + (\text{Medida}_2 * \text{peso2}) + (\text{Medida}_3 * \text{peso}_3)}{\text{peso1} + \text{peso2} + \text{peso3}}$$

Para nuestro caso, las medidas, serían las velocidades angulares medidas según cada sensor y el peso que le queramos asignar a cada sensor.

### 7.1.5. Corrección de ortogonalidad de Matriz de rotación.

Los cálculos en coma flotante y los errores introducidos por los sensores hacen que la matriz de rotación obtenida puede que no sea válida a todos los efectos, una matriz de rotación debe tener todos vectores ortogonales y unitarios (modulo 1). Por tanto para aplicaciones prácticas de estos cálculos, como la implantación que se ha realizado en c++ para Arduino, es necesario realizar una corrección para garantizar la ortogonalidad.

El método es en esencia sencillo ya que si tenemos 2 vectores casi ortogonales como podemos ver en la imagen, A y B podemos obtener un vector B' ortogonal a A con dos productos vectoriales, el método es similar al que utilizamos para corregir el magnetómetro.

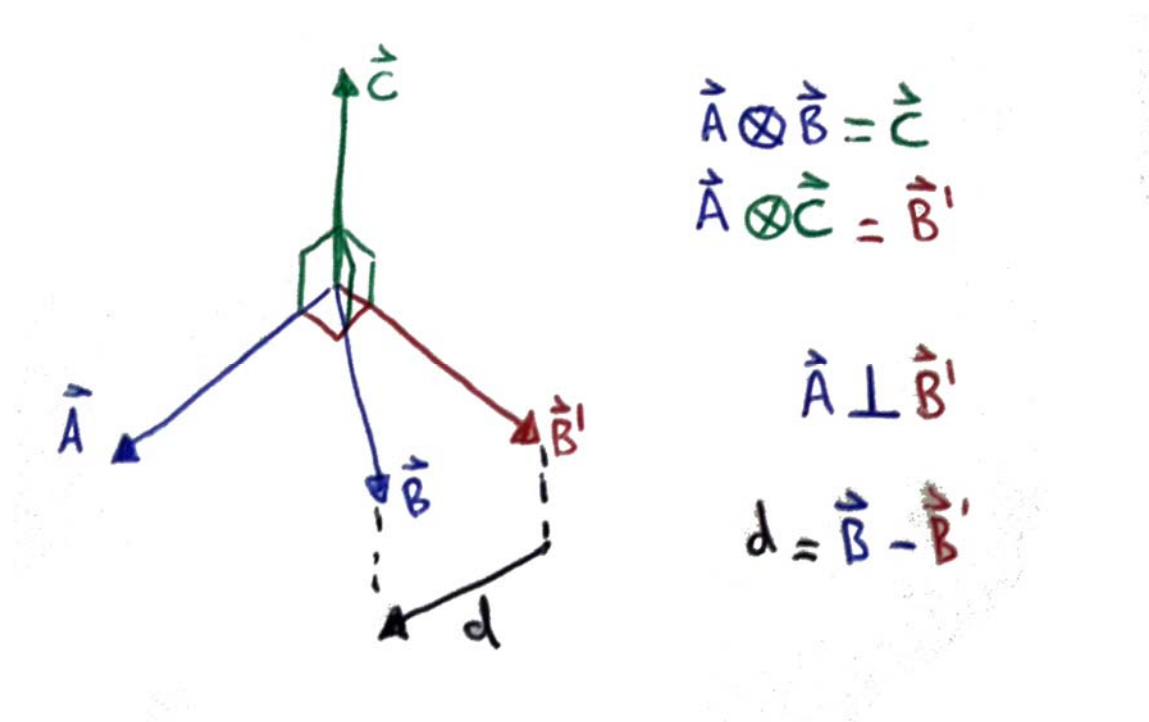


Figura 39: corrección ortogonalidad de dos vectores

En el cálculo vectorial existe una regla llamada del triple producto, que nos va a ayudar a simplificar esta operación, suponiendo que A y B son vectores unitarios (podemos normalizarlos antes para asegurarnos) la operación simplificada nos queda de la siguiente manera:

## 7. Técnicas de filtrado y fusión.

$$\mathbf{b}' = \mathbf{c} \times \mathbf{a} = (\mathbf{a} \times \mathbf{b}) \times \mathbf{a} = -\mathbf{a} (\mathbf{a} \cdot \mathbf{b}) + \mathbf{b} (\mathbf{a} \cdot \mathbf{a}) = \mathbf{b} - \mathbf{a} (\mathbf{a} \cdot \mathbf{b}) = \mathbf{b} + \mathbf{d}, \text{ donde } \mathbf{d} = -\mathbf{a} (\mathbf{a} \cdot \mathbf{b})$$

Para que la corrección sea completa es necesario hacer el cálculo a la inversa para corregir  $\mathbf{a}$ , y a la desviación equivalente la llamaremos  $\mathbf{e}$ . Ahora, tenemos la desviación de cada vector respecto del otro y la manera más correcta es aplicar una corrección de la mitad de  $\mathbf{e}$  y la mitad de  $\mathbf{d}$  a cada uno como se ve en la imagen:

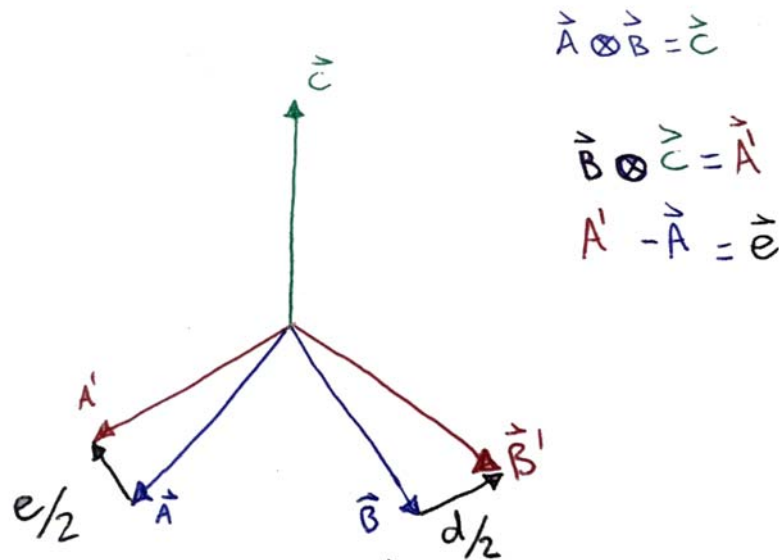


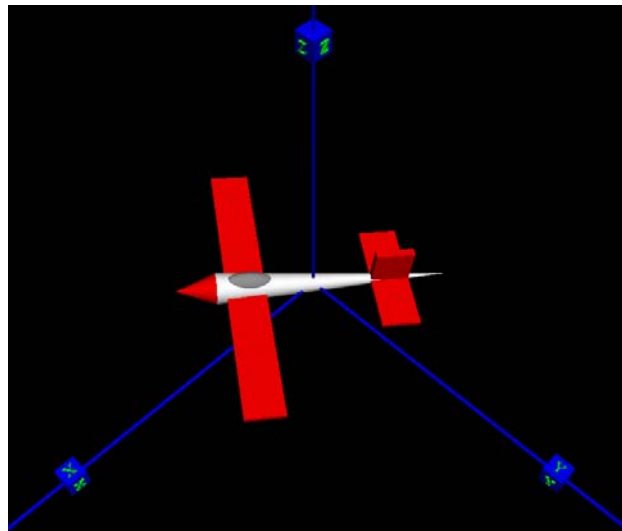
Figura 40: corrección de la ortogonalidad repartiendo el error

De igual que hicimos en el apartado anterior se puede simplificar la operación basándonos en que los vectores son unitarios y la regla del triple producto. De modo que el algoritmo resultante para la corrección de la ortogonalidad de la matriz de rotación es:

$$\begin{aligned} \text{error} &= (\mathbf{I} \cdot \mathbf{K}) / 2 \\ \mathbf{I1} &= \mathbf{I} - (\mathbf{K} \cdot \text{error}) \\ \mathbf{K1} &= \mathbf{K} - (\mathbf{I} \cdot \text{error}) \\ \mathbf{J1} &= \mathbf{K1} \times \mathbf{I1} \end{aligned}$$

#### 7.1.6. Aplicación de prueba: Avion.py

Esta aplicación ha sido desarrollada para probar si la implementación de las técnicas expuestas en este capítulo es correcta. La aplicación se conecta al puerto serie del dispositivo, y manda repetidamente el carácter 'a' que es el comando por el cual el dispositivo debe devolver la matriz de rotación en nueve valores separados por comas. Una vez recibidos pinta un avión sobre un sistema de coordenadas para comprobar si el avión se alinea perfectamente con el dispositivo.



*Figura 41: Aplicación Avion.py*

En la documentación adjunta a este documento, se puede encontrar el código fuente que implementa el IMU, la aplicación Avion.py y un video demostrativo de la aplicación en funcionamiento.

# 8. Redes de comunicación.

El siguiente grafico muestra una visión global de los distintos tipos de redes de comunicación que utiliza el sistema, en los siguientes apartados, se desglosa y explica cada una de ellas.

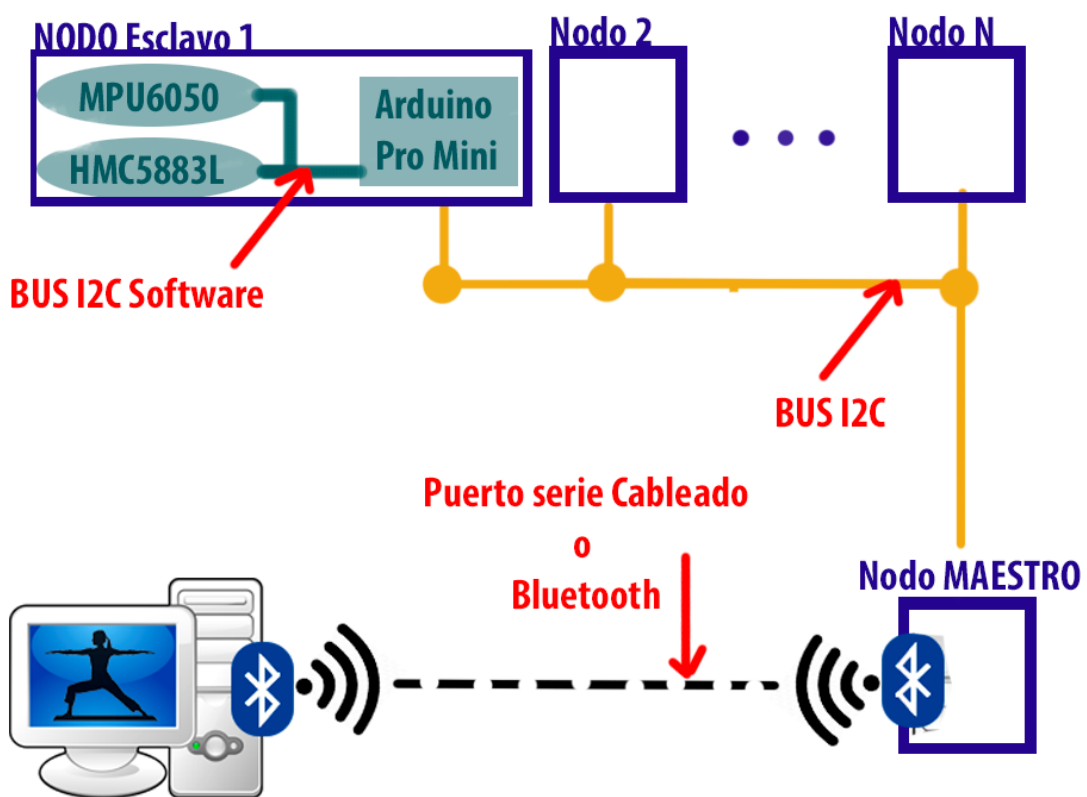


Figura 42: Redes de comunicación del dispositivo

## 8.1. Comunicación Sensores->Arduino. (I<sup>2</sup>C Software)

Los sensores, tanto el MPU6050 (acelerómetro + Giroscopio) como el HMC5883L (magnetómetro) poseen una interfaz I<sup>2</sup>C como se describió en los apartados 3.2.3 y 3.2.5 respectivamente y la conexión con el Arduino es a través de las pistas del circuito impreso diseñado para tal fin y descrito en la sección 9.1.1. Las placas gy-

271(HMC5883L) y gy-521(MPU6050) poseen resistencias pull up, que como se comentó en la sección correspondiente al protocolo I2C son necesarias en este bus.

Por otra parte, como se indica en el título, es necesario destacar que el microcontrolador Arduino Pro Mini, solo dispone de un bus I2C hardware y en este proyecto se utilizan 2 buses, el que conecta el Arduino con los sensores, y el que conecta la red de Arduino con el nodo central, y es por ello que se ha utilizado una librería que utiliza 2 salidas digitales del Arduino para simular la comunicación I2C mediante software, esta librería se distribuye mediante licencia GNU (General Public License), se basa en otra creada por Peter Fleury's, y esta traducida a ensamblador y por tanto es muy ligera y muy rápida, por tanto el gasto computacional que supone es mínimo. Haciéndola ideal para aplicaciones en tiempo real como esta.

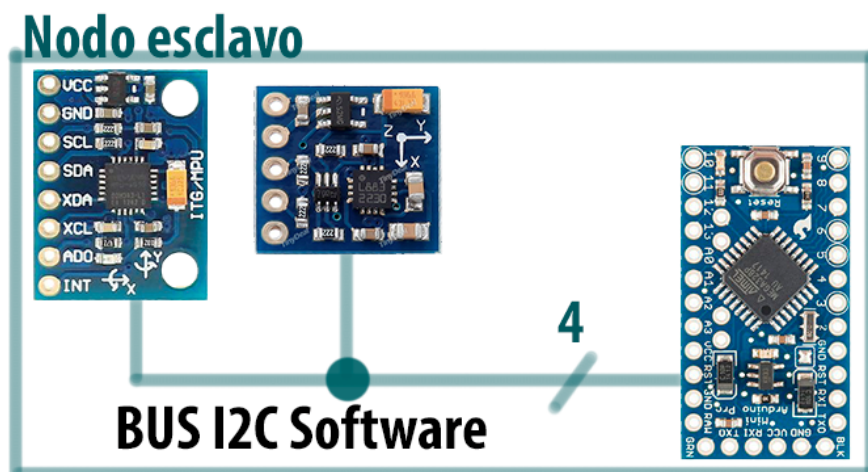


Figura 43: esquema bus i2c sensores-Arduino

## 8.2. Comunicación Nodos-Eslavos->Nodo-Maestro

La comunicación de los nodos esclavos con el maestro se realiza a través del bus i2c hardware del que dispone el Arduino de serie, de modo que el maestro realiza una pasada por todos los nodos solicitándoles su matriz de rotación, mediante peticiones i2c al bus. El nodo maestro es el que integra unas resistencias PullUp para el correcto funcionamiento del bus, ya que el Arduino no dispone internamente de ella.

### 8.3. Comunicación Maestro-PC

La comunicación con el nodo maestro es a través de puerto serie, pero en esta ocasión existen 2 versiones, la cableada y la inalámbrica.

La versión cableada utiliza el cable descrito anteriormente de la marca FTDI TTL-232R-5v, y los parámetros de conexión son velocidad de 250000 baudios, 8-N-1.

Para la versión Inalámbrica se dispone del módulo bluetooth Hc-05 que requiere una pequeña configuración en el PC que se va a conectar una única vez para emparejar los dispositivos, es una configuración sencilla a través del asistente de Windows. Una vez emparejados, el PC crea un puerto serie virtual que es a todos los efectos igual que uno físico, y el PC se puede conectar al dispositivo igual que si fuera una conexión cableada.

### 8.4. Aplicación de prueba: Esqueleto.py.

Y por último llegamos a la aplicación que nos sirve como demostración del proyecto completo, y que nos sirve para probar la comunicación en la red de nodos y representar la orientación de las articulaciones del cuerpo humano.

Como se puede ver en las siguientes imágenes la representación 3d se ajusta perfectamente a la modelo que lleva puestos los sensores por todo el cuerpo:

## Representación de la orientación de un sistema articulado





## 8. Redes de comunicación.



En la documentación adjunta a este proyecto se puede encontrar el código fuente de la aplicación esqueleto.py

# 9. Elementos Adicionales.

## 9.1. Hardware:

### 9.1.1. Circuito Impreso (PCB).

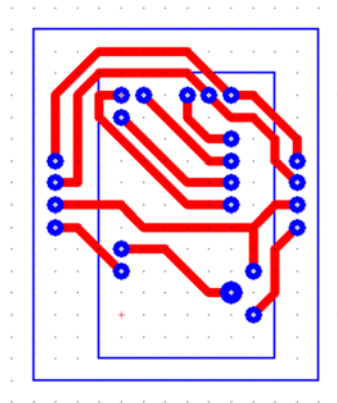
#### - Diseño del circuito Impreso

Para la realización de los nodos se ha tenido que diseñar un circuito impreso para conectar todos los componentes:

- MPU6050
- HMC5883L
- Arduino Pro Mini 5v 16MHz
- Puertos para BUS I2C de comunicación entre nodos

En la fase de diseño se realizó un gran esfuerzo para poder recoger todos los componentes en el menor volumen posible, el resultado es bastante satisfactorio, pero al ser componentes para prototipos el margen de mejora con montaje industrial y componentes SMD es enorme, es muy posible que con herramientas industriales el volumen del nodo se redujera hasta un 60% del volumen original, incluso más.

El resultado es el siguiente:



- **Realización de las placas por insolación**

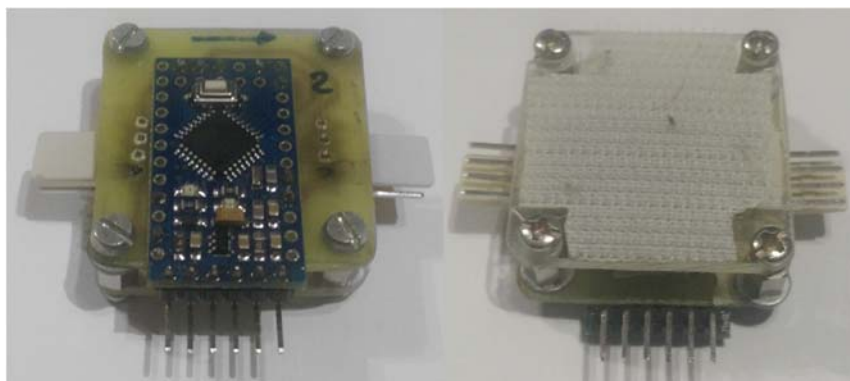
El interés por reducir el tamaño de la placa, hicieron que se descuidara la facilidad de soldar los componentes, haciendo que la tarea fuera muy frustrante y complicada ya que al ser un circuito que a pesar de disponer de una única capa, se sueldan componentes solapados por los dos lados, haciendo la soldadura un trabajo muy complicado y más teniendo en cuenta que el soldador del que se disponía a pesar de ser de buena calidad (JVC) la punta no era la más apropiada para ese tamaño.

El sistema para atacar a las placas primero fue el clásico y casero, pero efectivo, rotulador indeleble Eding y un atacado directo por una mezcla de ácido clorhídrico (HCl) con Peróxido de hidrógeno (agua oxigenada  $H_2O_2$ ) al 50%

Con este método el resultado era complicado de conseguir pistas finas de buena calidad y por tanto se decidió construir una pequeña insoladora Led, para la realización de las siguientes placas. La insoladora se realizó con diodos ultravioleta led, algunas resistencias, una caja para pinturas y un transformador.



El resultado fue unos circuitos impresos de más calidad:



## 9. Elementos Adicionales.

### 9.1.2. Cableado.

Para la conexión entre los nodos y el maestro se ha utilizado:

- Cable unifilar rígido de 0.6mm
- Conectores molex de 4 pines aéreo



### 9.1.3. Fijaciones:

Para las fijaciones se han utilizado tiras de nylon de 5cm de grosor, corchetes para regulación y velcro para la fijación y sujeción de los nodos en estas.



## 9.2. Software:

### 9.2.1. Configuración del entorno de desarrollo

#### - **Java SDK 1.7**

Para la programación de las apps de demostración vamos a utilizar el IDE Eclipse, y un requisito fundamental es tener instalado el SDK de Java, se descarga de:

<http://www.oracle.com/technetwork/es/java/javase/downloads/jdk7-downloads-1880260.html>

Es una instalación común, siguiendo el wizard correspondiente no tiene más complicación.

#### - **Eclipse.**

Es **uno de los IDEs más extendidos** actualmente para el desarrollo de aplicaciones Java principalmente aunque soporta otros lenguajes y la posibilidad de aumentar su funcionalidad mediante plugins ya que es código abierto. En la fecha de la realización de este proyecto la última versión y la utilizada es Eclipse Mars 2. Eclipse se distribuye bajo “**Licencia Pública Eclipse (EPL)**, es una licencia de software de código abierto utilizada por la Fundación Eclipse para su software. Sustituye a la Licencia Pública Común (CPL) y elimina ciertas condiciones relativas a los litigios sobre patentes

#### - **Plugin: PyDev**

Es un **plugin para Eclipse** que permite el desarrollo de aplicaciones en Python con este IDE.

- Se instala desde el Marketplace de Eclipse:

*help > install new software > <http://pydev.org/updates>*

- O desde la web [www.pydev.org/](http://www.pydev.org/) donde también podemos encontrar una versión Standalone que incluye un eclipse con el plugin ya instalado y configurado



## 9. Elementos Adicionales.

### - **Python 2.7 (32Bits)**

Python es el lenguaje de programación elegido para las aplicaciones de demostración, y necesitamos instalar el core. Python es un lenguaje de programación interpretado, es orientado a objetos y débilmente tipado lo que permite una buena flexibilidad a la hora de programar, es un lenguaje con una potencia expresiva equivalente a otros lenguajes como java pero más sencillo y rápido de programar y al ser interpretado perdemos en velocidad de ejecución pero ahorramos en desarrollo al no ser necesario compilar.

### - **Librería vPython**

Es una librería para mostrar gráficos 3d muy sencilla de programar y muy útil y ampliamente utilizado en la comunidad científica. Es una parte esencial del proyecto y de las aplicaciones de demostración. Se descarga desde la web oficial [vpython.org](http://vpython.org) y la instalación es sencilla, solo seguir el asistente.

### - **Librería PySerial**

Es una librería que proporciona una API de conexión con el puerto serie y las aplicaciones, escrita en Python para Python. Se descarga desde su web oficial.

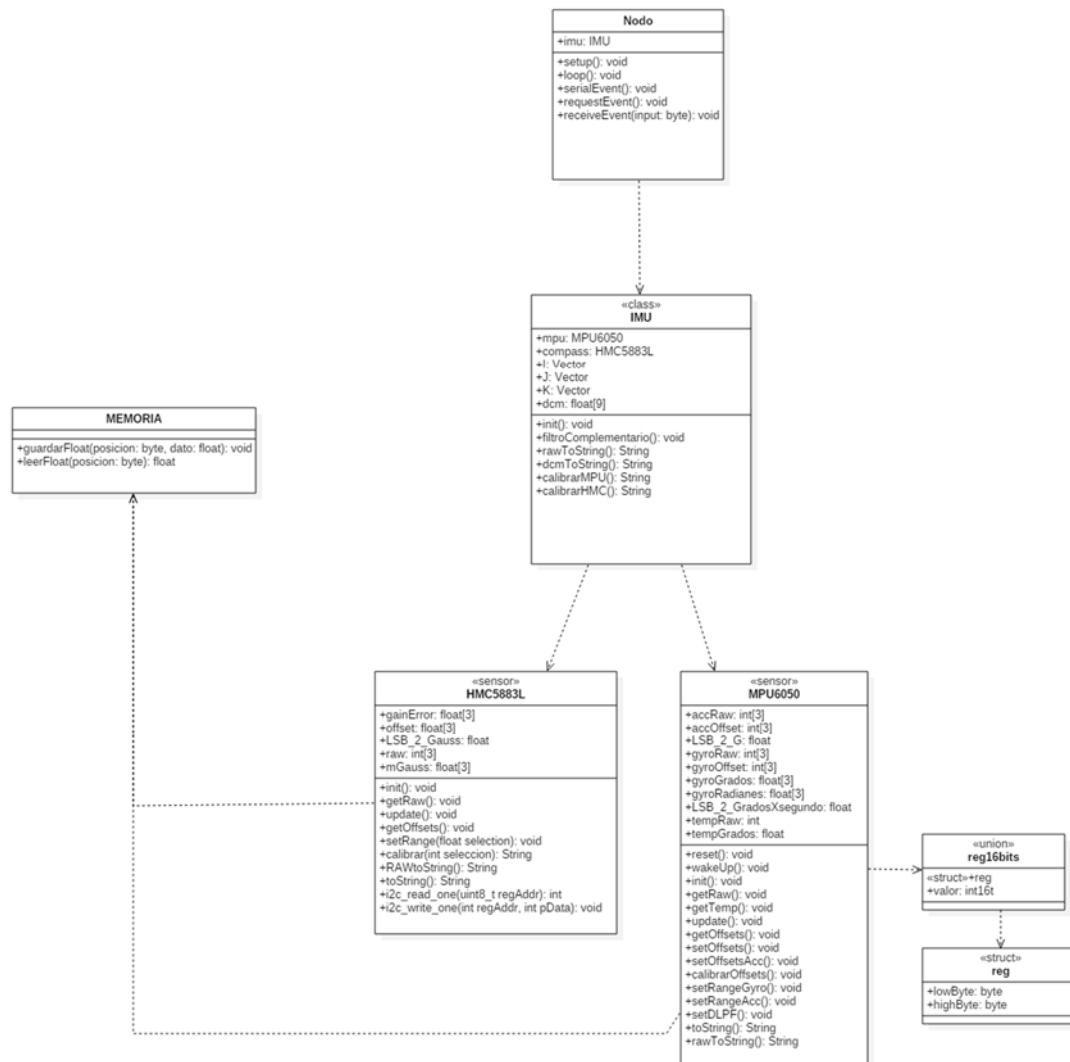
### - **Control de Versiones, SVN Subversion y Tortoise SVN.**

Aunque no es un requisito indispensable para el desarrollo, el control de versiones resulta muy útil a la hora de desarrollar, ya que proporciona una metodología, y un apoyo a la hora de gestionar posibles problemas, versiones anteriores, uso en diferentes dispositivos y copias de seguridad.

Para la realización de este proyecto se ha utilizado un servidor SVN proporcionado con una cuenta gratuita en la página web [asamblea.com](http://asamblea.com), y un cliente SVN instalado en cada equipo utilizado para el desarrollo de este proyecto.

Se barajó también la posibilidad de una cuenta github, pero este tipo de repositorios Git en su versión gratuita hacen obligatorio que el proyecto sea público y se descartó existiendo una posibilidad gratuita y privada como [asamblea.com](http://asamblea.com)

### 9.2.2. Diagrama UML del Firmware para Arduino.





# 10. Conclusiones.

## **El mundo no se mide se estima.**

Una palabra recurrente en la teoría que gira alrededor de la medición de estados del mundo físico a través de sensores es *estimación*, ya que se da por hecho que los datos que nos entregan los sensores, por precisos que sean, no se deben considerar como medidas validas, incluso después del filtrado y procesado, se da por hecho que es imposible tener las medidas exactas de la magnitud.

En la realización de este proyecto he podido constatar la diferencia entre escribir una formula en papel, e implementarla en el mundo real, que el ruido en las señales es un problema real y que a pesar de que los sensores cada vez son más precisos (y baratos) la tendencia cada vez más aceptada es que el ruido es inevitable, que el mundo real no se puede modelar y medir de una manera precisa, lo que nos obliga a implementar técnicas de estimación de estados posibles, filtrado de señales, calibración de sensores, fusión de sensores u otras técnicas probabilísticas para producir sistemas robustos, que sean capaces de lidiar con la imprevisibilidad del mundo que nos rodea.

## **Saber aprovechar los puntos fuertes y esquivar las debilidades**

Por otra parte en el proyecto se ha demostrado la fortaleza de saber lidiar con las debilidades de cada componente intentando evitarlas, y hacer uso de sus puntos fuertes, y es así como se pueden llegar a resultados válidos, ya que ninguna solución es perfecta y menos en la medición del mundo real.



# 11. Bibliografía y referencias.

- [1] ATmega168A Pulse Width Modulation – PWM. Protostack. (2011). Recuperado el 13 de marzo de 2016 de: <http://www.protostack.com/blog/2011/06/atmega168a-pulse-width-modulation-pwm/>
- [2] Barrientos, A. Peñin, L. Balaguer, C. & Aracil, R. (2007). *Fundamentos de Robótica*. Madrid: McGraw Hill.
- [3] Calibration Instructions. Gulf Coast Data Concepts. (2015). Recuperado el 15 de enero de 2016 de: <http://www.gcdataconcepts.com/calibration.html>
- [4] Compass heading using magnetometers. (s. f.) Honeywell. Recuperado el 10 de enero de 2016 de:  
<http://textlab.io/doc/3516186/compass-heading-using-magnetometers>
- [5] Domínguez, F. (2014). *Filtro complementario con DCM*. Madrid: Universidad Rey Juan Carlos.
- [6] Herramientas Matematicas para localización espacial. Departamento de industria y negocio. (2004). Universidad de Atacama Chile. Recuperado el 12 de febrero de 2016 de:  
[http://www.industriaynegocios.cl/Academicos/AlexanderBorger/Docts%20Docencia/Seminario%20de%20Aut/trabajos/2004/Rob%F3tica/seminario%202004%20robotica/Seminario\\_Robotica/Documentos/Herramientas%20Matematicas.htm](http://www.industriaynegocios.cl/Academicos/AlexanderBorger/Docts%20Docencia/Seminario%20de%20Aut/trabajos/2004/Rob%F3tica/seminario%202004%20robotica/Seminario_Robotica/Documentos/Herramientas%20Matematicas.htm)
- [7] Mahony, R.
  - a. (s. f.). A Complementary Filter for Attitude Estimation of a Fixed-Wing UAV
  - b. (s. f.). Complementary filter design on the Special Euclidean group SE (3).
- [8] (2006). A coupled estimation and control analysis for attitude stabilization of mini aerial vehicles.
- [9] (2008). Nonlinear Complementary Filters on the Special Orthogonal Group . *IEEE Transactions on Automatic Control*, vol. 53, no. 5.
- [10] MPU-6050 Accelerometer + Gyro. Arduino. Cc. (s. f. ). Recuperado el 10 de enero de 2016 de <http://playground.arduino.cc/Main/MPU-6050>

- [11] Ozyagcilar, T. (2012). Implementing a Tilt-Compensated eCompass using Accelerometer and Magnetometer Sensors. Document Number: AN4248 Rev. 3, 01/2012. Recuperado el 15 de marzo de 2016 de:  
[http://cache.freescale.com/files/sensors/doc/app\\_note/AN4248.pdf](http://cache.freescale.com/files/sensors/doc/app_note/AN4248.pdf)
- [12] Seguimiento de movimiento con sensores inerciales/magnéticos. Gradiant. Galician Research and Development Center in Advanced Telecommunications. (2010). Recuperado el 12 de marzo de 2016 de:  
<http://gradiant.org/noticia/seguimiento-de-movimiento-con-sensores-inercialesmagneticos-2/>
- [13] Sensores inerciales: El mundo en movimiento. Neo-Teo. (2011). Recuperado el 1 de mayo de 2016 de <http://www.neoteo.com/21690-sensores-inerciales-el-mundo-en-movimiento>
- [14] Prometec.net.
- a. (s. f.). Usando el MPU6050. Primeras pruebas con GY-201. Recuperado el 2 de febrero de 2016 de <http://www.prometec.net/usando-el-mpu6050/>
  - b. (s. f.). Los Sistemas de Medida Inercial. Acelerómetros y Giróscopos. Recuperado el 2 de febrero de 2016 de <http://www.prometec.net/imu-mpu6050/>
  - c. (s. f.). Modulo Bluetooth Hc-05. Recuperado el 21 de junio de 2016 de <http://www.prometec.net/bt-hc05/>
- [15] Renaudin, V. & Combettes, C. (2014). Magnetic, Acceleration Fields and Gyroscope Quaternion (MAGYQ)-Based Attitude Estimation with Smartphone Sensors for Indoor Pedestrian Navigation. *Sensors* 2014, 14, 22864-22890; doi:10.3390/s141222864.
- [16] Viltres, V. (2012). *Control de posición de un balancín con motor y hélice*. Valladolid: Universidad de Valladolid. Ingeniería Técnica industrial, Departamento de Ingeniería de Sistemas y Automática.
- [17] Vincent, D. (2014). Accurate Position Tracking Using Inertial Measurement Units. PNI Sensor Corporation in collaboration with Miami University. Recuperado el 20 de enero de 2016 de <http://www.pnicorp.com/wp-content/uploads/Accurate-PositionTracking-Using-IMUs.pdf>

## 11. Bibliografía y referencias.

- [18] Watson, M. (2012). MPU6050 Setup + Data Adquisition. Recuperado el 15 de enero de 2016 de <http://www.botched.co.uk/pic-tutorials/mpu6050-setup-data-aquisition/>
- [19] Williams, H. (2016). Amazing IMU-BASED motion capture suit turns you into a cartoon. Recuperado el 10 de marzo de 2016 de <http://hackaday.com/2016/01/23/amazing-imu-based-motion-capture-suit-turns-you-into-a-cartoon/>
- [20] 5Hertz. (2014). ABC del acelerómetro. Recuperado el 1 de febrero de 2016 de <http://5hertz.com/tutoriales/?p=228>
- [21] Blender.org\_(2016). Recuperado 9 de febrero 2016 de [www.blender.org](http://www.blender.org).
- [22] Honeywell.com (s.f) Recuperado 9 de febrero de 2016 de:  
<https://aerospace.honeywell.com/en/products/navigation-and-sensors/3-axis-compass-integrated-circuits>.
- [23] Invensense.com\_(s.f) Recuperado 19 de febrero de 2016 de:  
<https://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/>
- [24] \_Arduino.cc (2016). Recuperado el 10 de febrero 2016 de <https://www.arduino.cc/en/Main/ArduinoBoardProMini>.
- [25] \_ESP8266.com (2016) Recuperado el 21 de junio de 2016 de <http://www.esp8266.com/>
- [26] \_Pjrc.com (s.f) Recuperado el 21 de junio de 2016 de <https://www.pjrc.com/teensy/teensy31.html>
- [27] \_Sergiu Baluta\_\_(DCM tutorial – an introduction to orientation kinematics. Recuperado el 21 de junio de 2016 de [http://www.starlino.com/dcm\\_tutorial.html](http://www.starlino.com/dcm_tutorial.html)



# 12. Siglas, abreviaturas y acrónimos.

**DOF:** Degrees of Freedom, grados de libertad

**DCM:** (Direction Cosine Matrix)

**IMU:** (inertial measurement unit) Unidad de medición inercial

**FTDI:** (Future Technology Devices International) Marca escocesa especializada en Bus seriales

**TTL-232R-5v:** Cable distribuido por FTDI, para conversión USB-Puerto serie

**8-N-1.:** parámetros databits, paridad, y stopbits para comunicación serie