_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE UPV INGENIEROS DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

# Estudio de la aceleración y posición en sistemas inerciales por medio de Arduino

**María José Pardilla Márquez**

**Tutor: Ignacio Bosch Roig**

**Cotutor: Jorge Gosálbez Castillo**

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2015-16

Valencia, 30 de junio de 2016

# Agradecimientos

En primer lugar, me gustaría agradecer a los tutores de este proyecto, Ignacio Bosch Roig y Jorge Gosalbez Castillo su dedicación, esfuerzo, consejos y tiempo para ayudarme a sacar el proyecto a adelante. También quisiera dar las gracias al resto de profesores de la facultad, por enseñarme los conocimientos necesarios para llevar a cabo este estudio.

Quiero dar las gracias, especialmente, a mi familia por todo el apoyo prestado durante la carrera y siempre. A mis padres que siempre me han animado en los momentos más duros de la carrera. A mi hermano Carlos, por todo el tiempo invertido explicándome cualquier duda que tuviera y sus ánimos para mejorar mi proyecto; y a mi hermano David, por sus consejos ante mis primeros retos profesionales.

# Resumen

Este proyecto de la Escuela Técnica Superior de Ingeniería de Telecomunicaciones de la Universitat Politècnica de València tiene como objetivo el estudio de diferentes sistemas inerciales sobre Arduino y su exactitud a la hora de medir aceleraciones en un determinado objeto. Con el fin de ver la viabilidad de este tipo de sistemas a la hora de calcular velocidades y posiciones a partir de estas aceleraciones.

En esta memoria se ha comenzado por describir los conocimientos básicos para entender el funcionamiento de acelerómetros, giroscopios y del sistema Arduino, así como de las plataformas Arduino y Matlab en las cuales procesamos los datos y como a partir de éstos dispositivos y herramientas podemos estudiar la calidad y el error de las medidas de aceleración en sistemas inerciales.

La memoria está dividida en siete capítulos más anexos, en los cuales en el primero se hace una breve introducción acerca del proyecto y las herramientas utilizadas. Tras esto, se expone la gestión del proyecto, así como los tiempos invertidos en cada parte del mismo y el presupuesto necesario para llevarlo a cabo.

Después, se explicará con más detalle cada herramienta y entorno empleado para conseguir nuestro objetivo. Posteriormente se detallará el proceso seguido y los resultados, así como las conclusiones y las propuestas de trabajo futuro.

Toda la memoria está reforzada con imágenes para un mejor entendimiento de los elementos y funcionamiento del proyecto.

# Resum

Aquest projecte de l'Escolta Tècnica Superior d'Enginyeria de Telecomunicacions de la Universitat Politècnica de València té com objectiu l'estudi de diferents sistemes inercials i la seua exactitud a l'hora de mesurar acceleracions en un objecte determinat. A fi de veure la viabilitat d' este tipus de sistemes a l'hora de calcular velocitats y posicions a partir d'estes acceleracions.

La memòria es comença per descriure els coneixements bàsics per entendre el funcionament dels acceleròmetres, giroscopis i del sistema Arduino, així com de les plataformes Arduino i Matlab en les que processem les dades i com a partir d'aquests dispositius i eines podem estudiar la qualitat i l'error de les mesures d'acceleració en sistemes inercials.

La memòria està dividida en set capítols més annexes, als quals al primer es fa una breu introducció sobre el projecte i les eines emprades. Seguidament, s'exposa la gestió del projecte, així com els temps invertits en cada part del mateix i el presupost necessàri per portar-lo a terme.

Després, s'explicarà amb més detall cada eina i entorn utilitzats per conseguir l'objectiu. Posteriorment, es detallarà el procés seguit i els resultats, així com les conclusions i les propostes de treball futur.

Tota la memòria està reforçada amb imatges per una millor comprensió del elements i funcionament del projecte.

# Abstract

This project of the Escuela Técnica Superior de Ingenieros de Telecomunicaciones, belonging of the Universitat Politècnica de Valencia aims to study different inertial systems and their accuracy measuring accelerations in a given object. With the aim of evaluate the viability of this kind of systems to calculate velocities and positions from these accelerations.

This memory has begun describing the basic knowledge to understand the operation of accelerometers, gyroscopes and the Arduino system, as well as the Arduino and Matlab platforms in which process the data, and how we can study quality and error measures in inertial systems based on these devices and tools.

The memory is divided into seven chapters plus appendices, in which the first chapter is made a brief introduction about the project and the tools used. After this, the project management is explained, as well as the time spent on each part and the budget to carry out it.

Then, it is illustrate in more detail each tool and environment used to achieve our goal. Afterwards, it is describe the process and the results, as well as conclusions and proposals for future work.

The entire memory is strengthened with images for a better understanding of the elements and behaviour of the system.

# Índice

# Capítulo 1. Introducción y objetivos

El presente proyecto de la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València tiene como finalidad comparar distintos sensores, métodos y filtros para analizar los diferentes errores que afectan a los sistemas de navegación inerciales y así observar la calidad y veracidad de las medidas de aceleración en diferentes movimientos.

Para alcanzar este objetivo se plantea la utilización de sensores inerciales combinados con una plataforma de procesado. Los sensores son os encargados de medir los movimientos mientras que las plataforma de procesado es la encargada de recoger los datos de los sensores y aplicar las correcciones pertinentes.

Los dos sensores utilizados para la toma de medidas en este estudio son los sensores MMA7455 y el IMU GY81. El MMA7455 es un sensor inercial compuesto por tres acelerómetros, mientras que el GY81 complementa los tres acelerómetros con tres giroscopios y un magnetómetro. Como veremos a lo largo de los siguientes capítulos, los dos dispositivos tienen características diferentes por lo que serán estudiados de manera diferente, siempre teniendo como objetivo hallar la mejor forma para conseguir aceleraciones correctas.

En cuanto a la plataforma de procesado se ha elegido Arduino, ya que es de código abierto, el coste es reducido y tiene capacidad de cálculo suficiente para almacenar y corregir los datos medidos.

Además, y una vez estimadas las aceleraciones, en este proyecto también se pretende estudiar la viabilidad del cálculo de distancias recorridas con los dispositivos con un nivel aceptable de exactitud. Para ello se harán distintas pruebas de medida con diferentes filtros tanto en el acelerómetro MMA7455 como en la IMU GY81.

Esta memoria se compone de siete capítulos. El primero de ellos es ésta introducción, mientras que en el segundo se explica la metodología seguida en este proyecto y la evolución temporal del estudio. En el tercer capítulo se hace un estudio acerca de la tecnología actual relacionada con este proyecto: sensores, dispositivos y plataformas, entre otros. En el siguiente capítulo de esta memoria se plantean los conceptos teóricos básicos de los sistemas inerciales así como las fórmulas que se emplearán para el cálculo y corrección de aceleraciones y distancia. Más tarde, en el capítulo seis se muestra el desarrollo y programación de los dispositivos, para finalmente en el capítulo siete mostrar los resultados alcanzados y analizar las conclusiones halladas tras las pruebas.

# Capítulo 2. Metodología

## 2.1 Gestión del proyecto

En un primer momento, el proyecto fue diseñado para trabajar con una placa Arduino Mega2560 y el acelerómetro MMA7455 y evaluar la aceleración y distancia recorrida a través de los datos que recogía este sensor. Tras comprobar que todos los dispositivos funcionaban, surgió el primer problema a la hora de conseguir leer las aceleraciones correctamente, lo que se solucionó con una librería del propio acelerómetro MMA7455.

Una vez halladas las aceleraciones, se encontraron varias formas de guardar la información para después realizar las operaciones oportunas. La primera de ellas conllevaba usar el HyperTerminal de Windows, aplicación empleada para conectar nuestro ordenador con otros equipos mediante módem, cable o conexión TCP/IP; mediante esta última conexión el HyperTerminal recibía los datos que enviaba Arduino a través del puerto USB y los guardaba en formato .txt. Con estos datos en formato texto se pueden exportar al programa Microsoft Office Excel 2013 y a partir de ahí manejarlos o pasarlos al software matemático Matlab. Esta forma de obtener los datos era poco eficiente, por lo que finalmente se decidió realizar una comunicación serie directamente desde Matlab para poder manipular la información más cómodamente en esta herramienta.

Una vez con los datos de aceleración, tenía dos opciones para hallar la distancia recorrida:

- Mediante integrales iteradas de ecuaciones de recta entre dos medidas consecutivas de aceleración.
- Mediante las fórmulas de movimiento acelerado uniforme, tomando la aceleración de cada instante como movimientos diferentes y partiendo de una posición inicial.

Debido al alto coste computacional de la primera opción se eligió emplear la segunda.

Tras ver los resultados poco exactos de este acelerómetro, se probó a realizar medidas con otro sensor, un acelerómetro con giroscopio GY81, en el que también se realizó una comunicación serie desde Matlab para después procesar los datos desde esta plataforma. Tras el cálculo y mejora de las medidas de aceleración y distancia recorrida se llegó a las conclusiones mostradas en el último capítulo de esta memoria. Durante todo este proceso se ha ido elaborando la memoria hasta llegar a la actual.

## 2.2 Diagrama temporal

Este proyecto se ha llevado a cabo durante nueve meses, la duración de un curso escolar. Primeramente se comenzó realizando un estudio bibliográfico sobre la materia, para después comenzar con el desarrollo y programación del proyecto, a la vez que se evaluaba el proyecto de forma periódica para realizar cambios y mejoras. Finalmente, en los últimos meses de este proyecto se hizo una optimización del hardware y software hasta llegar a la versión que se muestra en esta memoria.

| | **Meses** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Tarea 1. Estudio bibliográfico. | X | X | X | X | | | | | |
| Tarea 2. Desarrollo hardware | | | X | X | X | | | | |
| Tarea 3. Desarrollo del software y puesta a punto | | | | X | X | X | X | X | |
| Tarea 4. Valoración del sistema | | | X | X | X | X | X | X | |
| Tarea 5. Optimización hardware y software | | | | | | X | X | X | X |
| Tarea 6. Redacción memoria, resultados y coord. | X | X | X | X | X | X | X | X | X |

# Capítulo 3. Estado de la tecnología

En este capítulo se realizará un estudio acerca de los distintos dispositivos que se han empleado, así como los diferentes modelos y alternativas a éstos que están en el mercado actualmente.

## 3.1 Hardware

El presente proyecto se ha realizado con los dispositivos que se detallarán a continuación: la plataforma Arduino, el acelerómetro MMA7455, y el módulo formado por acelerómetro, giroscopio y magnetómetro GY81. De la plataforma Arduino se detallarán los distintos modelos que hay en el mercado, así como la forma de comunicarse con el dispositivo externo y sus características. En el subapartado del acelerómetro MMA7455 se explicarán sus características y su funcionamiento, así como los distintos pasos a seguir para configurarlo. Finalmente, se describirá el sensor inercial GY81, explicando cada uno de sus elementos y cómo interactúan entre ellos.

En este estudio se ha elegido Arduino como plataforma base, pero también hay otras alternativas, la más popular es Raspberry Pi. Raspberry Pi es un ordenador de placa reducida de bajo coste, su función es similar pero sus características son diferentes en cuanto a capacidad, sistema operativo, y salidas operativas.

En este proyecto se ha elegido Arduino por estar más extendido y por ello contar con más librerías. A continuación se comenzará a explicar la plataforma Arduino.

### *3.1.1 Arduino*

Arduino es una plataforma electrónica diseñada para la creación de proyectos hardware y software libres, asequibles y fáciles de usar. En sus orígenes en el año 2005 las placas microcontroladoras Arduino fueron diseñadas para estudiantes de electrónica en el instituto IVREA (Italia), ya que el microcontrolador que usaban hasta el momento, llamado BASIC Stamp, era demasiado costoso para que dichos estudiantes pudieran experimentar y crear sus propios prototipos electrónicos. Poco a poco los dispositivos Arduino se han ido haciendo más ligeros, compactos y potentes, al igual que el resto de dispositivos en la tecnología.

Arduino es una plataforma de hardware libre o de código abierto, lo que quiere decir que su documentación (especificaciones, diseño, diagramas) deben estar publicadas de manera

gratuita  o bajo algún tipo de pago para permitir su distribución, modificación o creación de derivados, los cuales también deben ser libres. Estos principios son similares a los del software libre, características que también tiene el entorno de desarrollo integrado (IDE) de Arduino.

Esta plataforma se ha hecho muy popular en apenas unos años debido a las grandes y numerosas ventajas que tiene con respecto al resto de microcontroladores, algunas de ellas son:

- Precio: Arduino es mucho más asequible comparadas con el resto de plataformas de microcontroladores.
- Entorno de programación: El IDE de Arduino es sencillo, gratis y fácil de usar. Además de estar disponible tanto en Windows, como en Macinstosh OSX y Linux.
- Software y Hardware ampliables y de código abierto: Como hemos mencionado anteriormente, la documentación y diseños están publicados abiertamente para su distribución y modificación.

Arduino cuenta con diferentes productos, los principales tipos son placas, módulos, shields, kits y accesorios. Según el área tecnología o el nivel de conocimiento en el que queramos enfocar nuestro proyecto tenemos diferentes dispositivos:

- Arduino UNO: Una de las placas más extendidas tanto para uso inicial como para un nivel más experimentado además de la primera en salir a la venta. Está basada en el microcontrolador ATmega328P. Es la placa con más librerías de todos los dispositivos Arduino. Este dispositivo puede verse en la figura 1.
- Arduino PRO: Basada en los microcontroladores ATmega168 y ATmega328. Está pensada para usuarios con mayores conocimientos que necesitan precios más bajos y menor potencia.
- Arduino ZERO: Similar a Arduino UNO pero con mayor potencia de procesamiento.
- Arduino MEGA: Está basado en el microcontrolador ATmega2560. Sus características son superiores a los las placas Arduino anteriores. Sus características serán detalladas más adelante. Este modelo se muestra en la figura 2.
- Arduino YÚN: Basado en el microcontrolador ATmega32u4. Se basa en Arduino UNO pero ofrece además capacidades propias para Ethernet, WiFi, USB y micro-

SD. Está pensado para proyectos con conexión de dispositivos, como Internet of Things.

- Arduino GEMMA: Microcontrolador con mucho menor tamaño que el resto basado en el ATtiny85. Está pensado para proyectos relacionados con la tecnología wearable.

Además de estos ejemplos de placas existen módulos para conectarlos a dichos microcontroladores y realizar funciones específicas, como el ARDUINO PRO MINI o el ARDUINO NANO. También están en el mercado los shields, tarjetas de circuitos impresos que conectador a la placa Arduino la proveen de mayor funcionalidad, como es el caso del ARDUINO ETHERNET SHIELD o ARDUINO GSM SHIELD, ambos usados en las placas Arduino con fines de conectividad de dispositivos o Internet of Things.

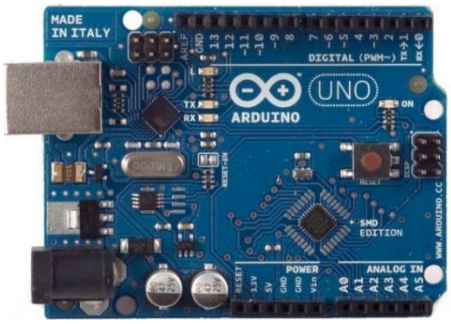



**Figura 1. Arduino Uno**


**Ilustración 2. Arduino Mega 2560**

La placa Arduino usada en este proyecto es Arduino Mega 2560. Dicha placa es una actualización del anterior Arduino Mega. Consiste en una placa basada en el microcontrolador de Atmel de 8 bits ATmega2560. Cuenta con un oscilador de 16 MHz. También posee 54 pines digitales de entrada/salida y 16 entradas analógicas, así como 4 UARTs TTL 0V/5V (puertos de transmisión/recepción serie) (Figura 3). Estos puertos serie están asociados a pines de la placa Arduino, por lo que no podemos utilizarlos como tal si estamos con los puertos de comunicación serie en uso. A continuación se muestran las especificaciones técnicas.

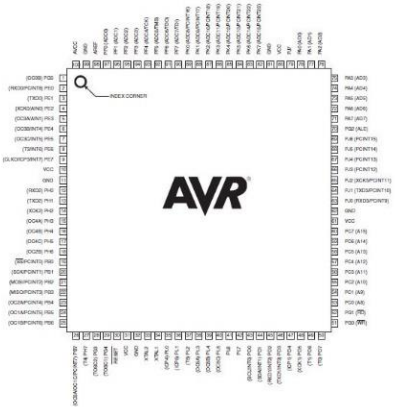



**Figura 3. Configuración pins ATmega2560**

| Especificaciones Técnicas | |
|---|---|
| Microcontrolador | ATmega2560 |
| Tamaño | 101.51 x 53.3 mm |
| Peso | 37 g |
| Voltaje de funcionamiento | 5 V |
| Voltaje de entrada (límite) | 6-20 V |
| Pins Digitales I/O | 54 (15 con salida PWM) |
| Pins Analógicos de entrada | 16 |
| Corriente DC por pin | 20 mA |
| Memoria Flash | 256 KB |
| SRAM | 8KB |
| EEPROM | 4 KB |
| Frecuencia Reloj | 16 MHz |

**Tabla 1. Características técnicas Arduino Mega 2560**

#### *3.1.1.1 Comunicación Arduino - Ordenador*

La forma de conexión de la placa Arduino MMA7455 con nuestro ordenador se hace a través de comunicación serie por nuestro puerto USB. Mientras que la norma creada para la comunicación en serie con otros periféricos en la que se especifica el protocolo, la velocidad y niveles de tensión que deben seguir se denomina I2C. Este bus de comunicación se utiliza principalmente para la conexión de elementos periféricos entre sí o con otros dispositivos. En la actualidad la mayoría de su uso está enfocado a la comunicación entre un microcontrolador y sensores periféricos.

El I2C está constituido por dos líneas: SDA y SCL. El SDA corresponde a los datos enviados, y el SCL al reloj, ambos son pines de la placa Arduino a los que tenemos que conectar nuestro sensor para comunicarnos. La línea SDA envía los datos en las dos direcciones, por lo que necesitamos un control de acceso y direccionamiento al bus. Por este motivo, I2C está diseñado con el esquema maestro-esclavo El maestro siempre inicia la comunicación y genera la señal de reloj (es el único que puede controlar la línea SCL), la cual es diferente para cada modo. Algunos de estos modos son:

- Estándar Mode (Sm): Modo bidireccional con una velocidad de transmisión máxima de 0.1 Mbit/s.
- Fast Mode (Fm): Modo bidireccional con una velocidad de transmisión máxima de 0.4Mbit/s.
- High Speed Mode (HSm): Modo bidireccional con una velocidad de transmisión máxima de 3.4 Mbit/s.
- Ultra Fast Mode (UFm): Modo unidireccional con una velocidad de transmsión máxima de 5 Mbit/s.

La trama del bus es la siguiente:

1- Bit de inicio de comunicación. Mientras el bit SCL esté a nivel bajo SDA no puede cambiar por lo que no se inicia la comunicación hasta que el bit SCL se modifique.
2- Dirección de 7 bits a la que enviamos o solicitamos la información.
3- Octavo bit de dirección en el que comunicamos si la información es enviada o solicitada.
4- Bit ACK enviado al final de cada byte para saber si la comunicación es correcta.

5- Datos enviados, donde normalmente definimos el registro del que queremos leer o en el que queremos escribir. A partir de aquí, enviamos todos los bytes de datos que necesitemos, cada uno de ellos seguido por un bit ACK.
6- Bit de finalización de la comunicación, similar al bit de inicio, ya que hay que cambiar la línea SCL de forma contraria para que SDA no emita datos.

En la placa Arduino Mega 2560 los pines SDA y SCL se encuentran respectivamente en los pines 20 y 21. Para poder usar esta comunicación es necesaria la librería Wire, ya incluida por defecto en el software de Arduino. Explicaremos el proceso de comunicación más adelante.

### *3.1.2 Acelerómetro MMA7455*

Los acelerómetros son dispositivos capaces de medir la aceleración a la que están sometidos. Dicha aceleración es la variación de la velocidad por unidad de tiempo, medido en $m/s^2$ o en fuerzas (G), con $G=9.8m/s^2$, la gravedad de la tierra.

$$aceleración = \frac{dV}{dt} \quad \textbf{(1)}$$

Estos dispositivos funcionan con tecnología MEMS. Contienen placas capacitivas tanto fijas como móviles. Los elementos móviles se mueven unos respecto a otros y según las aceleraciones que van sufriendo. Con este movimiento surgen las diferencias de capacitancia y con ello podemos saber la aceleración a la que está sometida el sensor.

Los acelerómetros pueden ser analógicos o digitales. Los sensores analógicos muestran un voltaje respecto a la aceleración sufrida en cada uno de los ejes, la cual está normalmente en torno a la tierra y Vcc. Los acelerómetros digitales obtienen unos resultados con menor ruido, dichos sensores se comunican a través de conexión SPI o I2C.

En cuanto al consumo los acelerómetros no precisan de grandes potencias, además de contar con una corriente necesaria del orden de microamperios o miliamperios, dependiendo del modo de funcionamiento en el que se programe cada sensor.

Una característica de los acelerómetros que complica su uso es la gran sensibilidad a vibraciones que tienen, por lo que las medidas suelen contener un error considerable y las hace difíciles de separar de las aceleraciones reales.

Uno de los sensores de los que vamos a disponer para este proyecto es el acelerómetro digital de 3 ejes MMA7455. Este acelerómetro es un sensor de baja potencia y bajo perfil

capacitivo, con filtro paso bajo, compensación de temperatura y sensibilidad ajustable. Cuenta con dos pines de interrupción para configurarlos en caída libre o movimiento rápido. Las características técnicas son las siguientes:

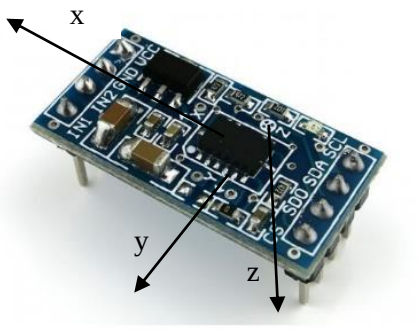



**Figura 4. Acelerómetro MMA7455**

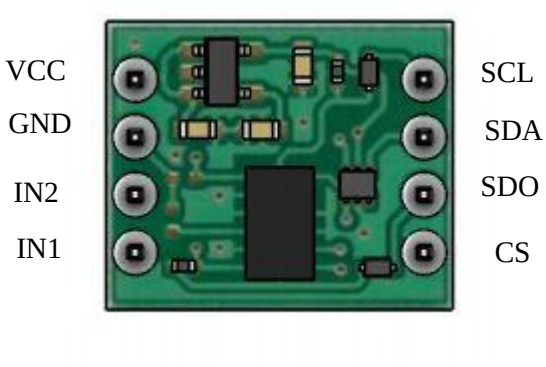



**Figura 5. Acelerómetro MMA7455**

| Parámetro | Valor |
|---|---|
| Voltaje requerido | 3.3 – 5V |
| Corriente requerida | 437 μA |
| Temperatura requerida | -40 – 85 °C |
| Sensibilidad | 2G, 4G, 8G |

**Tabla 2. Características técnicas acelerómetro MMA7455**

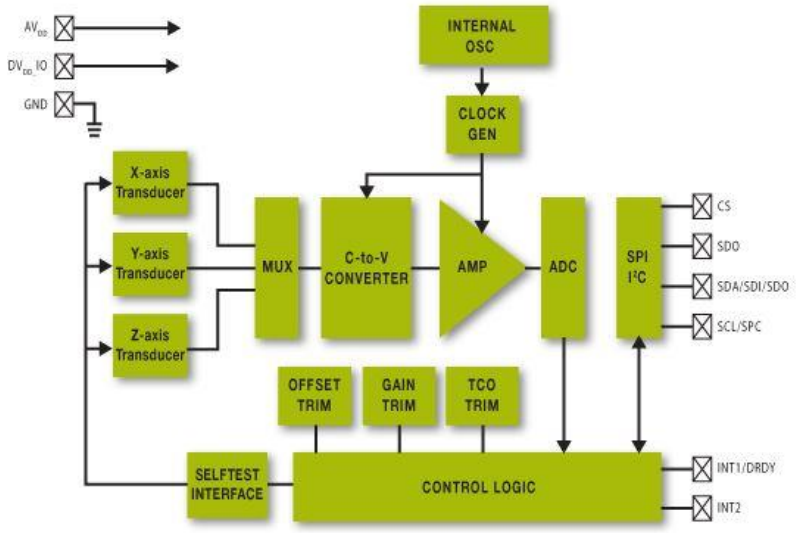



**Figura 6. Diagrama de bloques interno MMA7455**

El sistema de medición del acelerómetro MMA7455 es, de forma general, el explicado anteriormente. La superficie del sensor consiste en una célula capacitiva formada por materiales semiconductores, y un ASIC, Circuito Integrado para Aplicaciones Específicas. Dicha célula se puede modelar como un conjunto de placas capacitivas tanto fijas como móviles. Las láminas móviles pueden ser desplazas de su posición de reposo cuando el sistema está sometido a fuerzas de aceleración; como éstas están unidas a las placas fijas centrales, la distancia entre las láminas móviles de un lado y las placas centrales cambiará en relación a las del otro lado. Al ser elementos capacitivos, esta distancia se puede relacionar con cambios en la capacitancia, de la forma

$$C = \frac{A\varepsilon}{D} \qquad \textbf{(2)}$$

Donde A es el área de la placa, ε la constante dieléctrica y D la distancia entre las placas. El circuito integrado ASIC mide la capacitancia de los condensadores y extrae los datos de la aceleración a partir de la diferencia de capacitancia entre ellos. Además, ASIC filtra la señal para proporcionar una señal digital de aceleración correcta. El funcionamiento del sensor puede verse en la figura siete.

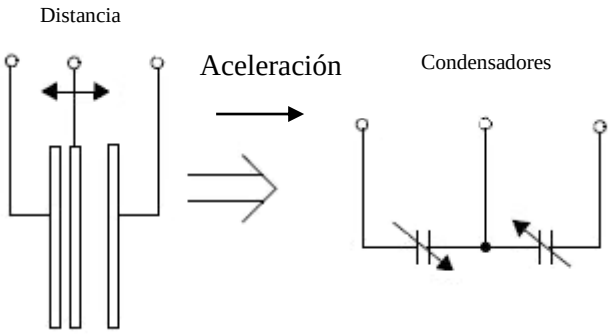



**Figura 7. Esquema funcionamiento acelerómetro MMA7455**

### *3.1.2.1 Autoexamen*

El sensor posee una función de autoexamen con el que puede verificar la integridad mecánica y eléctrica del acelerómetro en cualquier momento antes o después de la instalación. Cuando este test se inicia, se le aplica una fuerza electrostática a cada eje para que se desvíe, así, el eje z se recorta hasta que queda a una aceleración de 1g. De esta forma aseguramos que ambas partes de los componentes, mecánica y eléctrica funcionan correctamente.

### *3.1.2.2 Sensibilidad*

Una característica del acelerómetro muy útil en nuestro proyecto es la selección de sensibilidad. La sensibilidad es la capacidad de cambio de la salida del sensor respecto de las cambios de aceleración,

$$S = \frac{\Delta Vout}{\Delta g} \qquad (3)$$

En el sensor MMA7455 podemos elegir entre las sensibilidades 2g, con una sensibilidad de 64 LSB/g (Least Significant Bit/g); 4g, con una sensibilidad de 32 LSB/g; u 8g, con una sensibilidad de 16 LSB/g, siendo este el de mayor precisión. Las sensibilidades 2g y 4g toman las medidas con 8 bits de información, mientras que la sensibilidad 8g también tiene la opción de medir con 10 bits de datos. Esta característica se modifica en el registro de control de modo, cambiando los dos bits GLVL, como podemos ver en la tabla tres.

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Bit |
|---|---|---|---|---|---|---|---|---|
| --- | DRPD | SPI3W | STON | GLVL[1] | GLVL[1] | MODE[1] | MODE[0] | Function |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Default |

**Tabla 3. Registro control de modo MMA7455**

En la siguiente tabla se mostrará las modificaciones necesarias para ajustar la sensibilidad del acelerómetro.

| GLVL[1:0] | g-Range | Sensitivity |
|---|---|---|
| 00 | 8g | 16 LSB/g |
| 01 | 2g | 64LSB/g |
| 10 | 4g | 32LSB/g |

**Tabla 4. Configuración sensibilidad MMA7455**

#### *3.1.2.3 Modo Standby*

Este modo se utiliza en aplicaciones en la que no necesitamos que el acelerómetro esté recogiendo medidas continuamente. Cuando esta opción está activa, todas las salidas del dispositivo están apagadas, por lo que no se recogen nuevas mediciones aunque sí puede seguir escribiendo y leyendo de los registros de comunicación, con lo que la corriente se reduce a 26 µA. Este modo se activa modificando los dos bits de MODE en el registro de control de modo, como se pude ver en la tabla 5.

| MODE[1:0] | Function |
|---|---|
| 00 | Standby Mode |
| 01 | Measurement Mode |
| 10 | Level Detection Mode |
| 11 | Pulse Detection Mode |

**Tabla 5. Configuración modo MMA7455**

#### *3.1.2.4 Modo medida*

En este modo el acelerómetro recoge medidas de los tres ejes XYZ continuamente con la sensibilidad especificada en el registro de control de modo. La tasa de muestreo es de 125 Hz con un filtro de 62.5 de Hz de ancho de banda, o de 250 Hz con un filtro de 125 Hz de ancho de banda. Cuando la medida está completa, se activa el pin DRDY indicando que las medidas están preparadas para ser enviadas; hasta que una medida no está enviada no se envía la siguiente. Es posible elegir si queremos mostrar todos los ejes o algunos de ellos en concreto, además de decidir si queremos mostrar las aceleraciones en valor absoluto o con signo positivo o negativo.

### 3.1.2.5 Modo de impulsos

Este modo puede estar activo a la vez que el modo de medida. En esta opción se configuran los pines de interrupción INT1 e INT2 para detectar movimientos rápidos o caída libre. Si queremos detectar movimientos rápidos podemos configurar los pines de interrupción para que ambos detecten estas mediciones. Por defecto están activos los tres ejes con una sensibilidad de 8g, aunque se puede modificar de qué ejes queremos tomar las medidas. Para modificar los pines que queremos detectar se usan los bits XDA, YDA y ZDA, en el registro de control de detección de impulsos, donde también se modifican los pines de interrupción con los bits INTREG e INTPIN, lo cual podemos ver en la tabla 6.

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Reg $18 |
|---|---|---|---|---|---|---|---|---|
| DFBW | THOPT | ZDA | YDA | XDA | INTREG[1] | INTREG[0] | INTPIN | Function |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Default |

**Tabla 6. Registro de modo impulso MMA7455**

### 3.1.2.6 Respuesta

La aceleración del sensor MMA7455 sigue unos patrones debido a la gravedad de la tierra, por lo que haciendo las comprobaciones oportunas podemos comprobar si nuestro acelerómetro está mostrando la aceleración correcta. Estos patrones se muestran en la figura ocho.

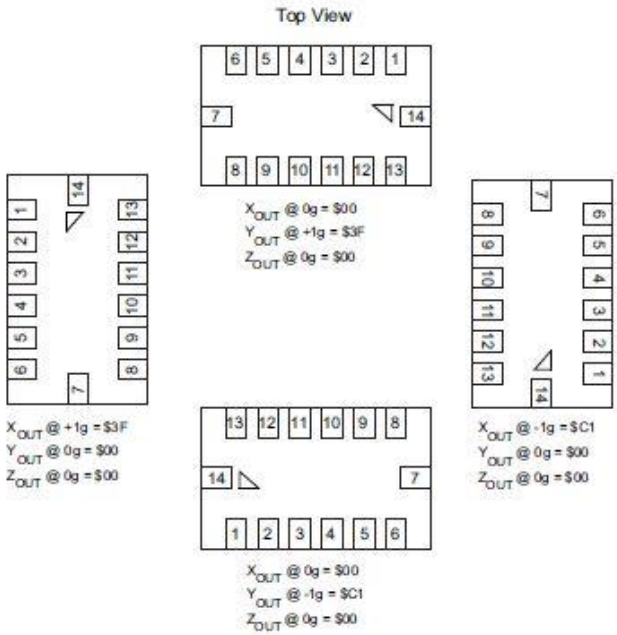



**Figura 8. Patrón aceleración según posición MMA7455**

Si mantenemos el acelerómetro en posición horizontal, la única gravedad que mostrará el sensor MMA7455 será en el eje z, la cual será de 1g, igual a la gravedad de La Tierra (1g=9.8m/$s^2$). Así, variando las distintas posiciones podremos comprobarla si la aceleración es correcta en cada eje.

### *3.1.3 Módulo de control de vuelo Gy-81 3205*

El segundo dispositivo a estudiar es la IMU GY-81 3205 diseñado por Bosch, la cual se compone de los siguientes sensores:

- Acelerómetro 3 ejes BMA180
- Giroscopio ITG3205
- Magnetómetro HMC5883L

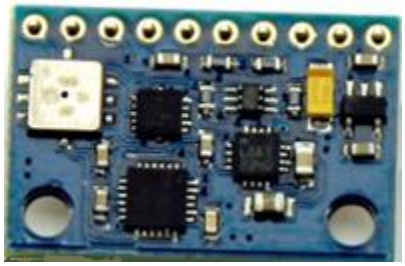


**Figura 9. Módulo GY81 frontal**

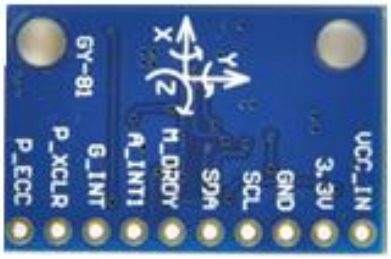



**Figura 10. Módulo GY81 trasero**

Para el proyecto actual no es necesario el uso del magnetómetro, por lo que a continuación se describirán solamente el acelerómetro y giroscopio que componen este módulo

#### *3.1.3.1 Acelerómetro 3 ejes BMA180*

El sensor BMA 180 es un acelerómetro digital con un funcionamiento similar al acelerómetro MMA7455 y un sensor de temperatura integrado. La comunicación se realiza en serie vía SPI o I2C. Dispone de filtros paso bajo, banda o alto ajustables desde 10 Hz hasta 1200Hz, lo que le confiere unos resultados con bajo nivel de ruido. Las medidas tienen una exactitud de 14 bits ADC en complemento a dos. Se puede leer solo en MSB (Most Significant Bit) en el que se usan solo 8 bits, o en LSB (Less Significant Bit) y MSB a la vez, en el cual se utilizan 16 bits con 14 bits de datos. En este proyecto se utilizará esta última configuración. A continuación se muestra la configuración de pins del acelerómetro BMA180, así como las especificaciones técnicas y el diagrama de bloques del acelerómetro.

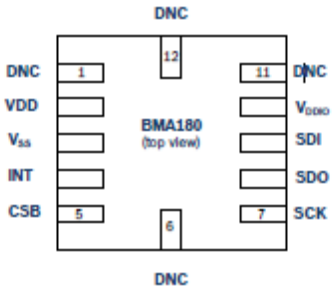



**Figura 11. Configuración de pin BMA180**

| Parámetro | Valor |
|---|---|
| Voltaje requerido | 1.62 V - 3.6V |
| Corriente requerida | 650µA |
| Temperatura requerida | -40 – 85 °C |
| Sensibilidad | 1G, 1.5G, 2G, 4G, 8G, 16G |

**Tabla 7. Características técnicas acelerómetro BMA180**

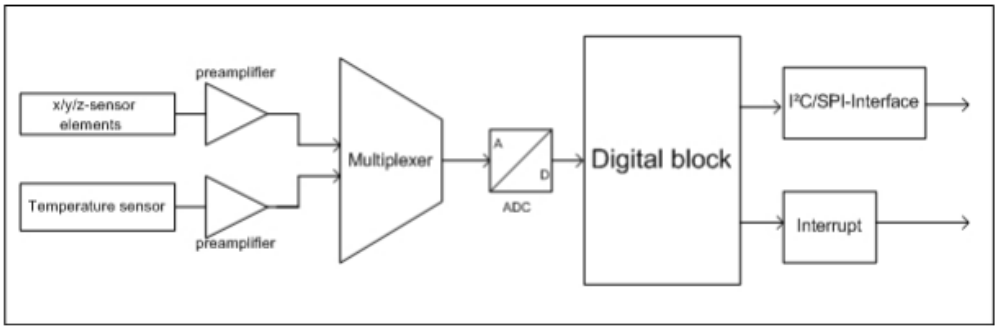



**Figura 12. Diagrama de bloques GY81**

#### *3.1.3.1.1.1 Modos de funcionamiento*

*Modo estándar*

El sensor BMA180 tiene 4 modos estándar según las distintas necesidades. Los datos son enviados vía I2C pudiéndose ajustar las interrupciones durante las medidas. Los 4 modos estándar son modo baja potencia, modo bajo ruido y dos modos intermedios: bajo ruido con baja potencia y bajo ruido con pequeño ancho de banda. En estos dos últimos casos la capacidad del sensor es más limitada.

*Modo sleep*

Con este modo la comunicación con el sensor se reduce lo máximo posible. Se activa y desactiva ajustando un bit en los bits de control. Esta opción es útil en el caso de que el sensor no se utilice a tiempo completo o para aplicaciones con poco ancho de banda, ya que permite ahorrar una gran cantidad de potencia y corriente intercalando modo estándar y modo sleep

*Modo wake-up*

Este modo de funcionamiento permite el cambio de modo sleep a estándar cuando detecta una determina aceleración en alguno de los ejes del sensor, por lo que el acelerómetro permanece con un consumo muy bajo sin perder ninguna medida. Este modo no consume corriente, por lo que la cantidad de corriente y potencia consumida es la media del modo sleep y modo estándar.

#### *3.1.3.1.1.2 Respuesta*

Tal y como se ha explicado con el sensor MMA7455, los acelerómetros tienen unos patrones de gravedad según su posición, por lo que podemos comprobar si la calibración de dicho sensor es correcta. Estos patrones aparecen en la figura trece.

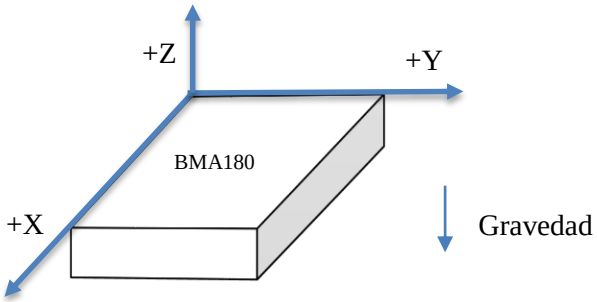



**Figura 13. Ejes GY81**

Si situamos el sensor en la posición que aparece en la figura 15 los datos obtenidos serán de -1G en el eje Z y 0G en los ejes X e Y. Esta será la primera calibración que debe hacerse al iniciar el dispositivo.

#### *3.1.3.2 Giroscopio ITG3205*

Los giroscopios son dispositivos capaces de medir la velocidad angular de un cuerpo en cada momento, es decir, la variación del desplazamiento angular por unidad de tiempo.

$$\omega = \frac{d\theta}{dt} \quad \textbf{(4)}$$

Según la forma de obtener las medidas, los giroscopios se dividen en dos categorías: mecánicos y electrónicos.

Los giroscopios mecánicos están basados en el principio de conservación del momento angular: si el momento de las fuerzas externas que actúan sobre un cuerpo es cero, su momento angular no varía. Estos sensores se implementan en aviones y satélites, donde el giroscopio es de un tamaño considerable además de costoso.

Los giroscopios electrónicos hacen uso de la tecnología MEMS. Son de mucho menor tamaño y se utilizan en numerosos objetos que usamos día a día, como los teléfonos móviles. Estos dispositivos contienen pequeñas masas unidas directamente al chip de silicio sensibles a la vibración basadas en el principio de Coriolis, el cual nos dice que la aceleración de un objeto dentro de un sistema de rotación aumenta o disminuye según si la dirección es hacia el eje de giro o en contra. Estos elementos miden las variaciones en la vibración según la velocidad angular, información que es transformada en señales eléctricas para poder ser leídos correctamente.

El sensor integrado en el módulo GY81 es el giroscopio electrónico de tres ejes ITG3205 con una salida de 16 bits ADC, por lo que se puede conseguir desde una medida de 8.000 muestras/seg hasta tan solo 3,9 muestras/seg. Cuenta con un sensor de temperatura y filtros paso bajo ajustables para mejorar la precisión. Al igual que el acelerómetro, la comunicación con el microcontrolador Arduino se realiza vía I2C.

La salida del giroscopio se mide en º/seg, pudiéndose pasar a rad/seg para un procesamiento más sencillo de los datos.

Al igual que los sensores anteriores cuenta con un “modo sleep” en el cual el consumo de potencia y corriente disminuye de 6.5 mA a 5 µA. El resto de especificaciones técnicas se detallan a continuación, así como los ejes de medida del sensor, en la figura catorce.

| Parámetro | Valor |
|---|---|
| Voltaje requerido | 2.1 V - 3.6 V |
| Corriente requerida | 6.5 mA |
| Temperatura requerida | -40 – 85 °C |
| Rango | 2000º /seg |

**Tabla 8. Características técnicas giroscopio ITG3205**

La orientación de los ejes de medida es la siguiente.

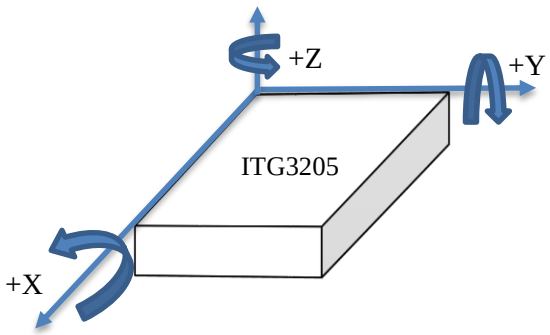



**Figura 14. Ejes ITG3205**

## 3.2 Software

El software que vamos a usar se ejecuta en la plataforma propia de Arduino para configurar el acelerómetro y el Arduino Mega 2560 así como obtener las mediciones de aceleración correspondientes a cada eje. Además, vamos a emplear el programa matemático Matlab para realizar los cálculos y los gráficos de la información obtenida, con el fin de poder obtener mejores resultados.

### *3.2.1 Software Arduino*

Arduino es un entorno de desarrollo integrado (IDE). Un IDE es un software informático creado para mejorar y facilitar la experiencia del programador en el desarrollo de aplicaciones, así como maximizar su productividad. Un IDE está formado principalmente por un editor de código y un depurador que nos informa de los errores cometidos, además de herramientas de construcción automática para reducir el tiempo de configuración y escritura de código. Algunos IDE están creados para lenguajes específicos, para así poder ofrecer herramientas más cercanas a dicho lenguaje, aunque lo más populares soportan multitud de lenguajes, como los entornos de desarrollo Eclipse o Microsoft Visual Studio.

El software Arduino es gratuito y fácil de usar, implementado en Java y basado en Processing, entorno de desarrollo para la producción de proyectos multimedia y de diseño también creado en Java.

La plataforma Arduino tiene un lenguaje propio basado en C/C++, con lo que las la mayoría de funciones estandar de C y C++ son soportadas por Arduino. A pesar de esto, podemos programar en el entorno Arduino con otros lenguajes de programación como Java, Python, Matlab o Processing entre otros, debido a que Arduino se comunica a través de conexión serie con el microcontrolador, y la mayoría de lenguajes de programación puede trabajar con esta comunicación. Los que no soportan esta comunicación serie también pueden utilizarse en el entorno de Arduino con un programa intermediario que permita la conexión entre ambos. Ésta es otra de las grandes ventajas que nos ofrece el software y microcontrolador Arduino.

La IDE Arduino cuenta con múltiples librerías estándar ya implementadas en su software, con lo que solo debemos escribir la orden de añadirlas para poder hacer uso de ellas en nuestro programa. Las librerías más importantes son:

- EEPROM: Librería empleada para la lectura y escritura en la memoria EEPROM, una memoria permanente de sólo lectura que puede ser modificada mediante tensiones eléctricas.[2]
- Ethernet: Permite comunicar el Arduino con Internet o una red doméstica mediante la conexión a través de servidores.
- LiquidCrystal: Librería utilizada para la escritura o borrado en pantallas LCD.
- Servo: Permite al Arduino controlar servomotores, motores eléctricos con distintas funcionalidades.
- SoftwareSerial: Librería empleada para crear puertas software, es decir, conseguir que cualquier pin del Arduino actúe como puerto serie.
- Wire: Esta librería se utiliza para la conexión I2C de Arduino. Permite al dispositivo actuar como master o esclavo y enviar y recibir información mediante comunicación serie.

En este proyecto, la comunicación entre la plataforma de Arduino y Matlab se realiza por parte de Arduino mostrando por pantalla las medidas, las cuales quedan reflejadas en el monitor serie y se envían al software Matlab, donde se reciben siguiendo el proceso descrito en el capítulo

### *3.2.2 Software y lenguaje Matlab*

Matlab es un entorno de desarrollo integrado (IDE) con licencia de pago diseñado como herramienta matemática por MathWorks. Soporta los sistemas operativos Windows, Mac y Linux. Cuenta con un lenguaje propio llamado Matlab, el cual admite operaciones, funciones y programación orientada a objetos. Además, cuenta con diferentes interfaces gráficas llamadas toolboxes con funcionalidades especiales a través de Simulink, un módulo diseñado en Matlab para simular sistemas dinámicos. Algunos ejemplos de toolboxes son: Signal Processing and Communications, Image Processing and Computer Vision, Code Generation and Verification, enter otros.

Las funciones y programas diseñados en Matlab pueden realizarse desde la propia interfaz gráfica de Matlab o mediante scripts, archivos con extensión .m. Además, soporta funciones en lenguaje C o Fortran a través de una función intermediaria que pase esos datos a formato Matlab.

En nuestro proyecto, usaremos Matlab para la comunicación serie con la placa Arduino, de donde recibiremos los datos tales como aceleraciones, velocidades angulares, ángulos y temperaturas. En la plataforma Matlab se procesarán los datos y se harán los cálculos y gráficos necesarios para el completo estudio de nuestro objetivo.

La comunicación de Matlab con Arduino se realiza mediante una comunicación serie a través de un puerto USB. A continuación se muestra el código para establecer comunicación a través del puerto COM4 de nuestro ordenador.

```
%% Conexion
%Buscamos conexiones abiertas
a = instrfind;
%Eliminamos conexiones encontradas
delete(a)
%Creamos conexion por puerto serial COM4
s = serial('COM4','BaudRate',9600,'Terminator','CR/LF');
%Abrir el puerto
fopen(s);
```

**Figura 15. Conexión Matlab - Arduino**

# Capítulo 4. Conceptos teóricos de los sistemas inerciales

El segundo dispositivo con el que se realiza este proyecto, el módulo GY81 es un sensor IMU, mide la aceleración y la rotación del elemento en el que se instala el sensor teniendo como referencia un determinado sistema de coordenadas.

En este estudio se usará el sistema de coordenadas donde los ejes tienen como centro de masas el dispositivo en el que se integran los sensores, moviéndose con él. Se puede pasar a otros sistemas de coordenadas como el sistema de coordenadas de Navegación, donde sus ejes se mueven según la longitud y latitud en la que se encuentra el dispositivo, con un simple sistema de ecuaciones matriciales.

Como se ha comentado en el capítulo anterior la aceleración que medirá el acelerómetro dependerá de la posición en la que se encuentra el sensor, por lo que sabiendo que en la posición base (esto es parado y paralelo a la superficie) la aceleración en los ejes X e Y será 0g, y la aceleración en el eje Z será la de la gravedad. A partir de estas aceleraciones en reposo, podemos determinar la orientación que tiene el objeto a medir mediante el cálculo de los ángulos pitch, roll y yaw.

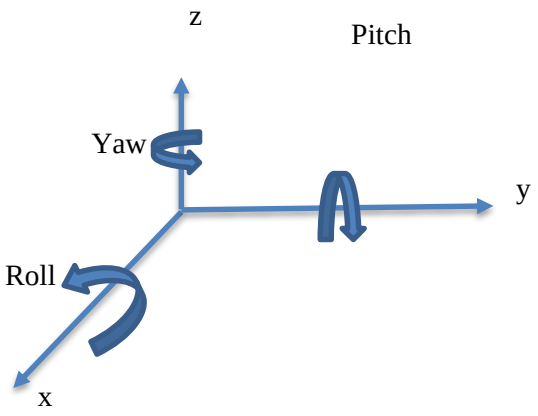



**Figura 16. Ángulos pitch, roll y yaw**

Con el acelerómetro y el cuerpo en reposo sólo se pueden hallar los ángulos pitch y roll dado que los ángulos se calculan a partir de la gravedad, por lo que en el eje z se necesitaría un sensor más para poder medir este ángulo. Las fórmulas para este cálculo son:

$$Ángulo\ pitch = \ \text{atan}\left(\frac{a_x}{\sqrt{a_y^2 + a_z^2}}\right) \quad \textbf{(5)}$$

$$Ángulo\ roll = \ \text{atan}\left(\frac{a_y}{\sqrt{a_x^2 + a_z^2}}\right) \quad \textbf{(6)}$$

Siento $a_x$ la aceleración medida en el eje X, $a_y$ la aceleración en el eje Y y $a_z$ la aceleración en el eje Z.

La evolución de la orientación del objeto también puede hallarse a través del giroscopio mediante el uso de integrales. De esta forma si se podrían obtener los ángulos de los tres ejes, aunque en este proyecto solo se calculan los ángulos de los ejes x e y ya que son los únicos que nos interesan para combinarlos con el acelerómetro.

$$\theta_{x(n)} = \theta_{x(n-1)} + \ \omega_{\ x(n)} \times T \quad \textbf{(7)}$$

$$\theta_{y(n)} = \theta_{y(n-1)} + \ \omega_{\ y(n)} \times T \quad \textbf{(8)}$$

Siendo θ el ángulo en cada muestra, ω la velocidad angular y T la diferencia de tiempo entre muestras.

El giroscopio es muy preciso en sus medidas, pero al tener que hallar el ángulo con integrales el pequeño error que se vaya produciendo se irá acumulando produciéndose el denominado efecto drift, lo que provocará medidas completamente alejadas de la realidad.

La razón de calcular los ángulos con ambos sensores para después combinarlos es que ambos acumulan un error considerable, por lo que si solo utilizáramos uno de ellos el ángulo hallado diferiría considerablemente del real.

La forma de combinarlos es utilizando un filtro complementario, uno de los filtros más usados en la plataforma Arduino ya que requiere poco coste computacional y los resultados son buenos. Este filtro implementa un filtro paso alto en el ángulo hallado por el giroscopio para erradicar el drift, y un filtro paso bajo para eliminar el ruido en el ángulo calculado a partir del acelerómetro, así, estos dos filtros se combinan para dar unos resultados más exactos.

$$Ángulo\ total\ pitch\ _{(n)} = 0.98 \times \left(Ángulo\ total\ pitch_{(n-1)} + \ \theta_{x(n)}\right) + 0.02 \times pitch_{(n)}$$

**(9)**

$$Ángulo\ total\ roll\ _{(n)} = 0.98 \times \left(Ángulo\ total\ roll_{(n-1)} + \ \theta_{y(n)}\right) + 0.02 \times roll_{(n)}$$

**(10)**

Una vez calculados estos ángulos podemos hallar la aceleración de la gravedad en cada eje según la orientación en la que se encuentre con referencia siempre al estado de reposo y paralelo a la superficie.

$$g_x = \ -g \times sen(pitch) \quad \textbf{(11)}$$

$$g_y = \ g \times cos(pitch) \times sen(roll) \quad \textbf{(12)}$$

$$g_z = \ g \times sen(pitch) \times cos(roll) \quad \textbf{(13)}$$

Siendo $g_x$ la aceleración por gravedad en el eje X, $g_y$ la aceleración por gravedad en el eje Y, y $g_z$ la aceleración por gravedad en el eje Z. La letra g define el módulo de la aceleración, ya que en estado de reposo y sea cual sea su orientación el módulo permanece constante.

Con estos datos para cada muestra podemos hallar la aceleración más próxima a la real cuando al objeto en cuestión se le aplica una determinada aceleración $a$.

# Capítulo 5. Desarrollo y programación

En este capítulo se mostrará del desarrollo seguido durante el proyecto para conseguir los objetivos propuestos, así como el conexionado y programación de cada dispositivo y plataforma.

Las características de cada dispositivo estudiado en este proyecto han sido diferentes, por lo que el procedimiento para cada uno de ellos también lo ha sido. Algunos pasos como la medida y envío de los datos ha sido similar en ambos sensores.

## 5.1 Desarrollo MMA7455

El esquema de conexionado con la placa Arduino para tomar las medidas de aceleraciones ha sido el que se muestra a continuación en la figura 17.

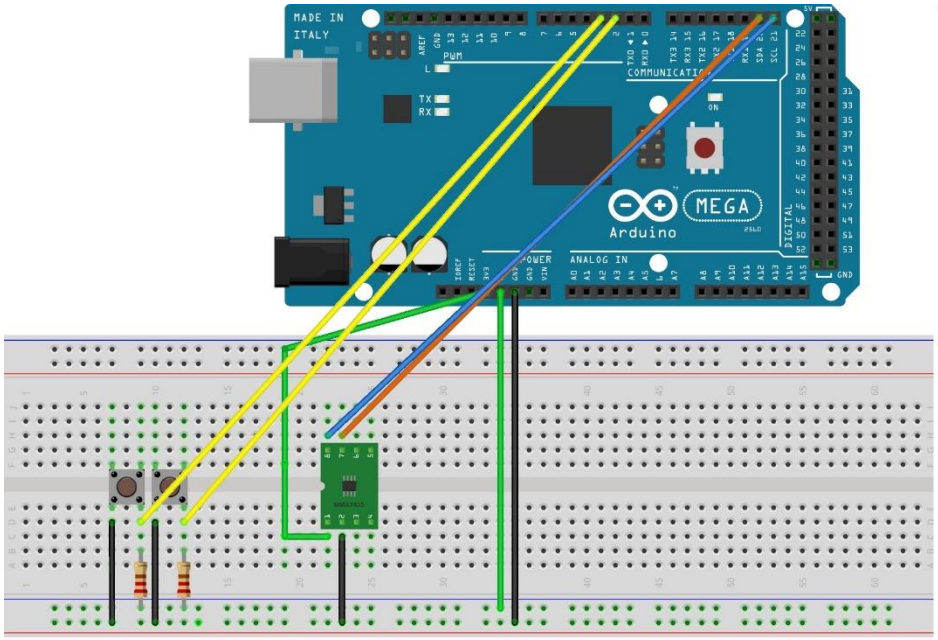



**Figura 17. Conexión Arduino Mega y MMA7455**

La comunicación serie del sensor se ha realizado a través de Arduino con las librerías Wire y MMA-7455. En esta plataforma se ha configurado la sensibilidad del sensor a 2G debido a que es la más sensible a cambios de aceleración y la velocidad de transmisión de datos a 9600. Tal y como se ha explicado en el capítulo 4.1.2 antes de comenzar a tomar medidas debemos configurar en los registros de control el modo de funcionamiento, en este caso modo medida declarando los ejes en los que queremos medir las aceleraciones; también debemos ajustar la sensibilidad y la comunicación serie con el Arduino.

```
#define MMA_7455_ADDRESS 0x1D //I2C Address for the sensor
#define MMA_7455_MODE_CONTROLL 0x16 //Call the sensors Mode Control

#define MMA_7455_2G_MODE 0x05 //Set Sensitivity to 2g
#define MMA_7455_4G_MODE 0x09 //Set Sensitivity to 4g
#define MMA_7455_8G_MODE 0x01 //Set Sensitivity to 8g

#define X_OUT 0x06 //Register for reading the X-Axis
#define Y_OUT 0x07 //Register for reading the Y-Axis
#define Z_OUT 0x08 //Register for reading the Z-Axis
```

**Figura 18. Registros MMA7455**

Las medidas se recogen en dos etapas controladas por dos pulsadores instalados en la placa protoboard junto al sensor. A continuación se describen ambos pasos del proceso.

### *5.1.1 Calibración*

Para esta etapa es necesario que el dispositivo se encuentre en estado de reposo y paralelo a la superficie en la cual se van a realizar las mediciones. Al pulsar el primer botón se inicia un tiempo de espera de 4 segundos para evitar rebotes del pulsador o vibraciones del movimiento de la persona al tocar el dispositivo. Tras esto se envían medidas durante un tiempo de 2 segundos a Matlab donde se realizan los cálculos para calibrar correctamente el dispositivo según el tipo de error:

- Error de offset: Este error es interno al dispositivo, tiene que restarse a las medidas en bruto para conseguir las aceleraciones reales. Para solucionar este error realizamos la media de las mediciones y restaremos este valor a las aceleraciones halladas en la siguiente etapa.
- Ruido por vibraciones: Los acelerómetros son muy sensibles a todas las aceleraciones, por lo que en las medidas tomadas con estos dispositivos siempre hay un ruido de menor o mayor tamaño el cual hay que subsanar. Para ello, y mediante herramientas estadísticas como la varianza, se ha diseñado un filtro paso bajo para así eliminar en la medida de lo posible las aceleraciones por vibraciones y poder diferenciar mejor la aceleración del movimiento.

### *5.1.2 Medidas*

Tras la etapa de calibración se comienza la toma de medidas para las distintas pruebas que se explicarán en el capítulo 6. Para obtener un resultado lo más aislado al movimiento posible se ha probado con varios filtros:

- Filtro smooth: realiza la media de las medidas hasta el momento en cada iteración.

$$x_{s(1)} = x_{(1)} \tag{14}$$

$$x_{s(2)} = \frac{x_{(1)} + x_{(2)}}{2} \tag{15}$$

$$x_{s(3)} = \frac{x_{(1)} + x_{(2)} + x_{(3)}}{3} \tag{16}$$

- Filtro mediana: similar al smooth pero en vez de utilizar la media sobre las muestras aplica la mediana. En este proyecto se ha utilizado el filtro de mediana de una dimensión.

Después de la toma de medidas se ha añadido al procesamiento de las aceleraciones un detector de movimiento, para que si el sistema detecta un número determinado de aceleraciones cuyo valor absoluto es muy inferior a la varianza determine que el sistema está parado y por lo tanto todas las variaciones son debidas a error de vibraciones.

Como se puede ver todos los algoritmos presentados tratan de mejorar la relación señal a ruido de las aceleraciones.

## 5.2 Módulo GY81

Este sensor es más completo que el MMA7455 ya que dispone de un giroscopio el cual nos servirá para poder separar las aceleraciones por movimiento de los errores por vibraciones y offset y realizar una mejor calibración y procesamiento de datos.

El esquema de conexionado entre el módulo GY81 y el Arduino es el mostrado en la figura 19.

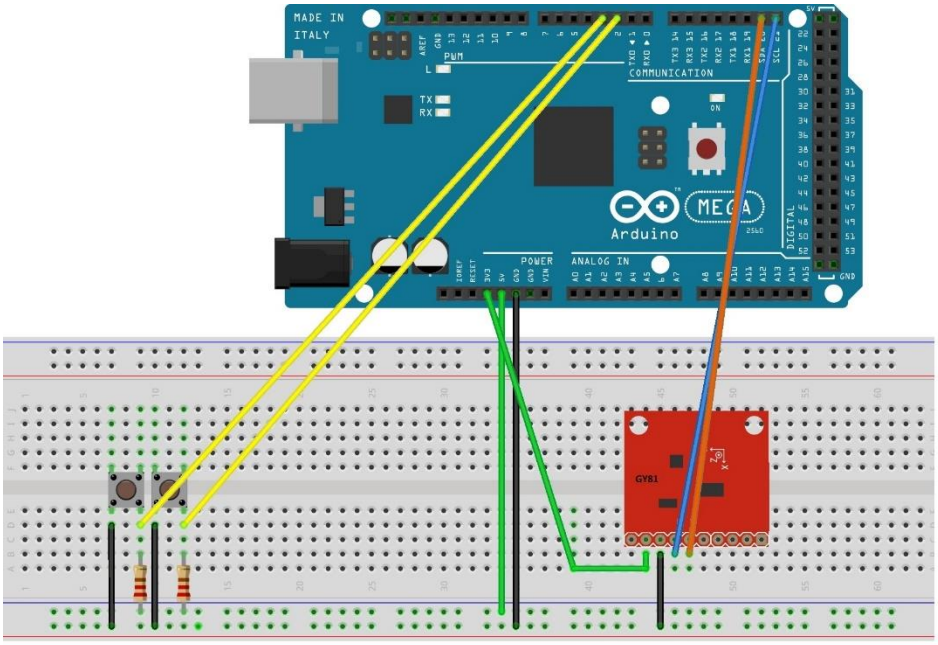



**Figura 19. Conexión Arduino Mega y GY81**

El proceso seguido para la medida correcta de las aceleraciones ha sido diferente al utilizado en el MMA7455. En este caso se ha diseñado un sistema con cinco etapas.

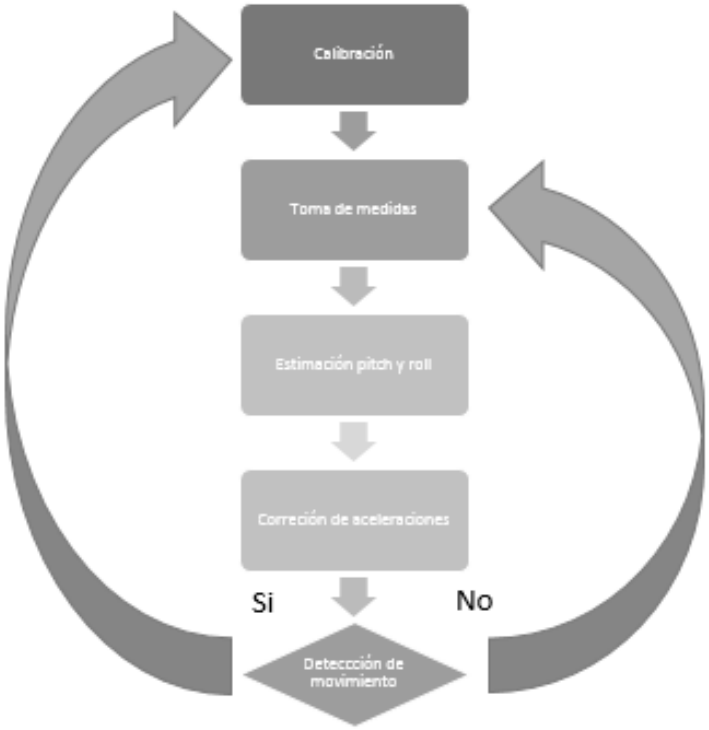



**Figura 20. Etapas módulo GY81**

### *5.2.1 Calibración*

La primera etapa es la calibración, en este caso es similar a la del sensor MMA7455, con dos pulsadores, uno para la calibración con un tiempo de espera de 4 segundos, y otro para la toma de aceleraciones.

En estado de reposo y paralelo a la superficie se miden las aceleraciones, velocidades angulares y temperatura, ésta última para comprobar que ambos sensores funcionan correctamente y no se sobrecalientan. Debido al estado de reposo del objeto, tanto la aceleración como la velocidad angular debería ser cero exceptuando la aceleración gravitacional.

Asimismo se obtienen los cálculos de pitch, roll y la media de cada uno de ellos para estimar la orientación a la cual está sometido el objeto. Lo ideal sería que la salida de ambos ángulos fuera cero, lo que significaría que toda la aceleración de la gravedad recae sobre el eje z y que la superficie está perfectamente alineada.

Con esta calibración se estiman los errores por offset y el ruido por vibraciones.

### *5.2.2 Toma de medidas, estimación de pitch y roll y corrección de aceleraciones*

Tras la calibración, y pulsando el segundo botón comienza la toma de medidas. En esta etapa se sigue calculando el pitch y roll total, el cual se va acumulando a la media de los ángulos hallados en la calibración. De esta forma, se puede saber además de la aceleración la orientación en la cual se está moviendo, y así separar la gravedad de las medidas reales con las siguientes fórmulas.

$$a_x = a_{x\_medida} - g_x \quad \textbf{(17)}$$

$$a_y = a_{y\_medida} - g_y \quad \textbf{(18)}$$

$$a_z = a_{z\_medida} - g_z \quad \textbf{(19)}$$

Donde $a_x$, $a_y$ y $a_z$ son las aceleraciones totales del movimiento en cada eje; $g_x$, $g_y$ y $g_z$ son las aceleraciones gravitacionales explicadas en el capítulo 5, y, por último, $a_{x\_medida}$, $a_{y\_medida}$, $a_{z\_medida}$ son los datos medidos en bruto tras la calibración.

Para un mejor procesado se utilizan también los filtros de mediana y smooth explicados en el capítulo 5.1.2 con las aceleraciones $a_x$, $a_y$ y $a_z$.

También y para un mejor resultado del estudio, se han calculado las distancias recorridas durante el movimiento del objeto, viendo cómo y en qué condiciones las mediciones son más exactas. La distancia recorrida se calcula mediante dobles integrales, la primera para hallar la velocidad y la segunda para calcular la distancia.

$$v_{x(nt)} = v_{x(nt-1)} + a_{x(nt)} \times T \quad \textbf{(20)}$$

$$d_{x(nt)} = d_{x(nt-1)} + v_{x(nt)} \times T \quad \textbf{(21)}$$

Siendo $v_{x(nt)}$ la velocidad en cada instante $n \cdot T$ en el eje X, $d_{x(nt)}$ la distancia en cada instante $n \cdot T$ en el eje Y, y $T$ la diferencia de tiempo entre muestras.

Como se puede ver, los pequeños errores que haya en la aceleración se acumularán provocando drift y haciendo que tanto los valores de velocidad como de distancia estén alejados de los reales.

### *5.2.3 Detector de movimiento*

En este dispositivo también se ha implementado un detector de movimiento. Cuando se detecta que las aceleraciones en valor absoluto son muy pequeñas comparadas con la varianza calculada en la calibración, el sensor deja de medir y queda a la espera de reiniciar el sistema con la nueva calibración. Podemos ver un ejemplo de implementación en la figura 21.

```
% Detectar movimiento o stop eje x
cont_mov=0;
for i=1:nmuestras
if abs(ax(i))<=2*var_ax
    cont_mov=cont_mov+1;
    if cont_mov>=15;
        detector_mov=0;
    else
        detector_mov=1;
    end
end
end
```

**Figura 21. Detector de movimiento en Matlab**

## 5.3 Presupuesto

En esta tabla se indica el presupuesto necesario para llevar a cabo este proyecto, incluyendo tanto software como hardware y el coste de las horas necesarias para llevarlo a cabo.

| EQUIPAMIENTO SOFTWARE | Cantidad | Costo |
|---|---|---|
| Licencia Matlab | 1 | 300€ |
| Software Arduino | 1 | 0€ |
| **Total** | **2** | **300€** |

| EQUIPAMIENTO HARDWARE | Cantidad | Costo |
|---|---|---|
| PC | 1 | 300 € |
| Placa Arduino Mega 2560 | 1 | 40€ |
| Acelerómetro MMA7455 | 1 | 5€ |
| Sensor de vuelo GY81 | 1 | 25€ |
| Conjunto cables | 1 | 2€ |
| **Total** | **5** | **372€** |

| PERSONAL | Horas | Precio/Hora | Total |
|---|---|---|---|
| Ing. Téc. Telecomunicación | 200 | 35,80*0.65 | 4.654,00€ |

| TOTAL DE COSTO REAL | 5326 € |
|---|---|

El presupuesto del proyecto asciende a CINCO MIL TRESCIENTOS VEINTISEIS euros.

# Capítulo 6. Resultados

En este capítulo se van a mostrar los resultados obtenidos en cada etapa con cada uno de los dispositivos. Comenzando por el acelerómetro MMA7455 para a continuación pasar a mostrar los resultados más relevantes del GY81.

## 6.1 Acelerómetro MMA7455

Como se ha indicado en el capítulo anterior, tras la calibración se muestra la aceleración sin filtro y la aceleración filtrada en los tres ejes en la misma posición que aparece en la ilustración 26 y en estado de reposo. Ver figuras 22,23 y 24.

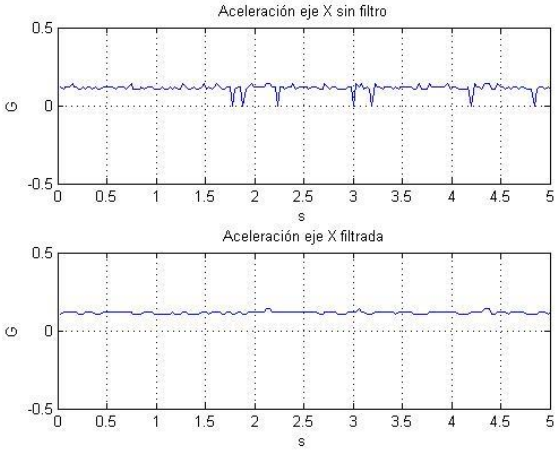



**Figura 22 Aceleraciones eje X MMA7455**

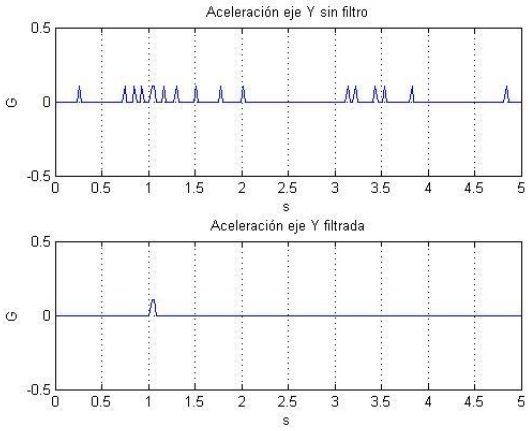



**Figura 23. Aceleraciones eje Y MMA7455**

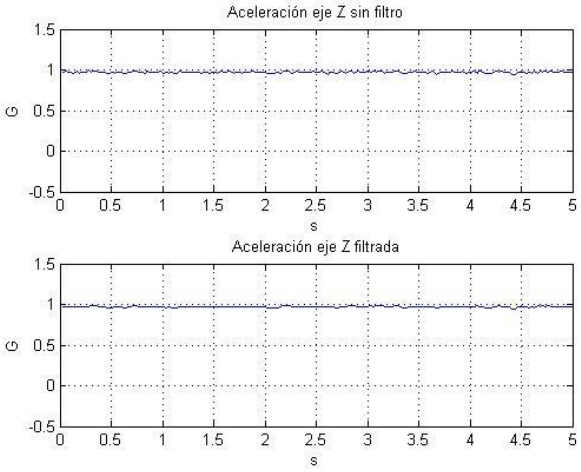



**Figura 24. Aceleraciones eje Z MMA7455**

Como se puede observar a pesar de no haber grandes oscilaciones hay mucho ruido de fondo en todas las medidas lo que dificulta mucho el cálculo de datos a partir de estas medidas y su veracidad. Al no disponer de giroscopio no podemos contrarrestar de otra forma estos errores en las medidas.

La última prueba realizada por este sensor es el cálculo de la distancia recorrida. Para ello, se aplicará una aceleración en una distancia de 30 centímetros.

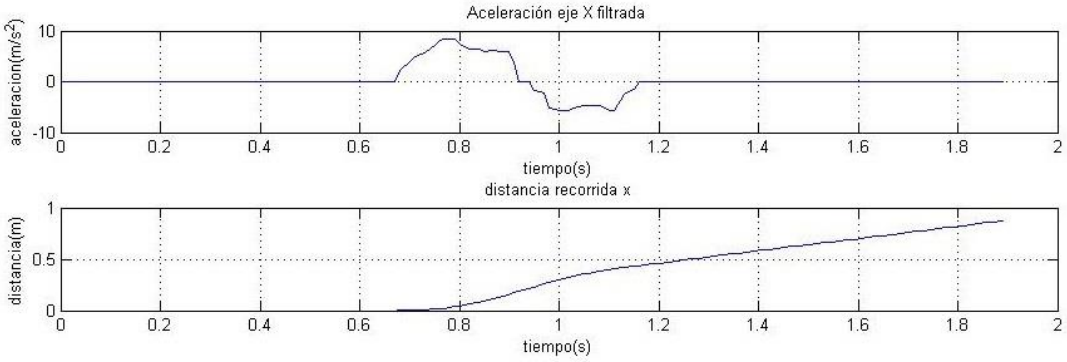



**Figura 25. Distancia 30cm MMA7455**

La aceleración se ha graficado en metro/segundo$^2$ para que sea más fácil de analizar. Se puede observar como si la aceleración es suficientemente fuerte el filtro elimina el ruido de fondo y la distancia, aunque incorrecta, es coherente.

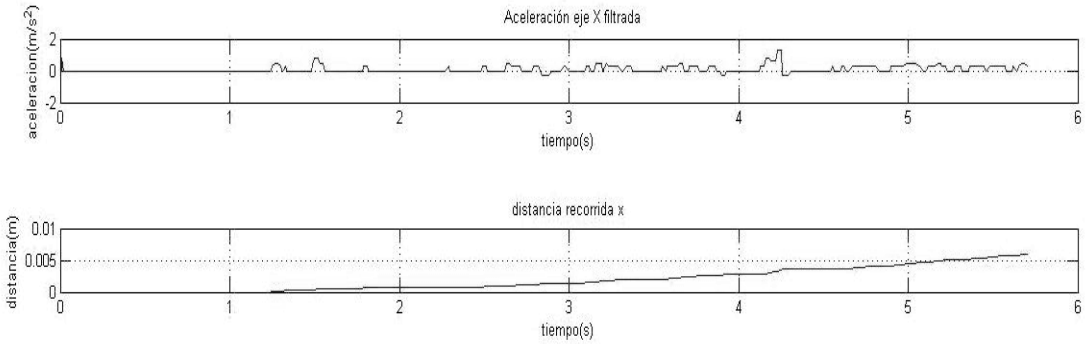



**Figura 26. Distancia 30 cm MMA7455**

Como se puede observar en la figura 26, si avanzamos a una velocidad más lenta el acelerómetro no recoge aceleraciones bruscas y no es capaz de distinguir el ruido de las

aceleraciones por movimiento, por lo que sale una aceleración completamente alejada de la realidad.

## 6.2 GY81

En este apartado se detallarán los resultados del acelerómetro más giroscopio GY81. Este sensor cuenta con una mayor precisión, y, con la suma del giroscopio las medidas pueden ser mejoradas, al contrario que con el acelerómetro MMA7455.

Primeramente, se mostrarán las aceleraciones de las calibraciones para observar el ruido bruto que afecta a cada muestra, sin ningún tipo de filtro ni procesado. En este caso el dispositivo está colocado paralelo a la superficie y en estado de reposo. Ver figura 27.

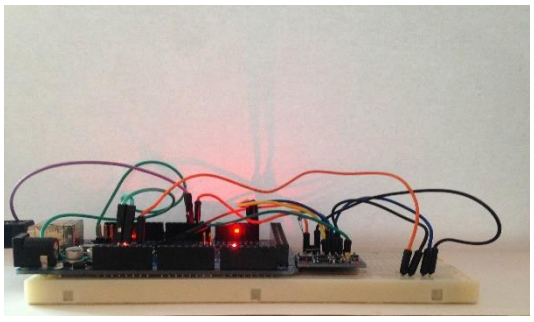


**Figura 27. Posición GY81**

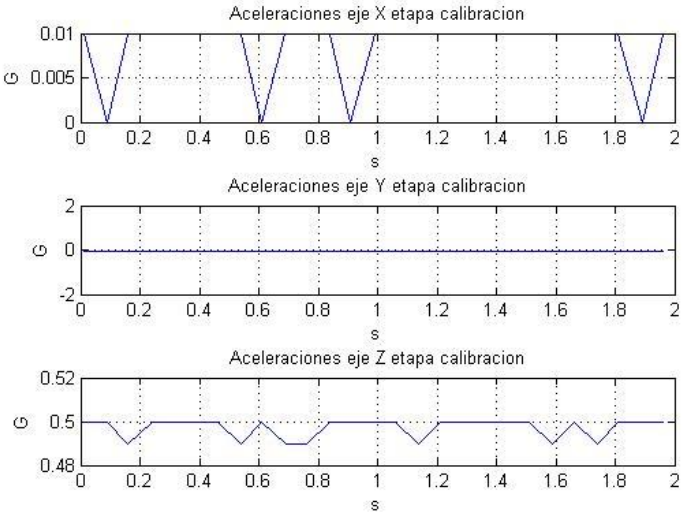



**Figura 28. Aceleraciones calibración GY81**

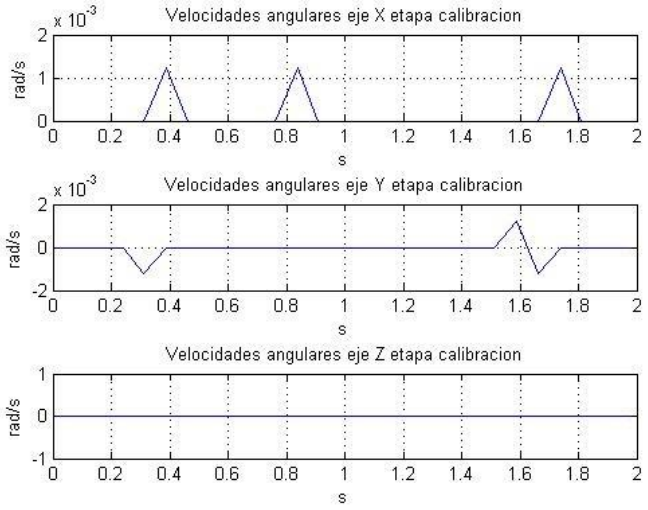



**Figura 29. Velocidades angulares calibración GY81**

Como se puede observar en las figuras 28 y 29 prácticamente toda la aceleración recae sobre el eje Z, en el cual el valor es de 0.5G con algunas oscilaciones debido a pequeños ruidos que se solucionarán con los filtros que se añadirán posteriormente.

También se puede ver como el ruido y oscilaciones de este dispositivo es mucho menor que en el acelerómetro MMA7455, motivo por el que se eligió este sensor para el estudio.

El módulo de la aceleración es 0.50G; este valor debería permanecer constante en cualquier posición en estado de reposo. Para esclarecer las fuentes de los errores que se aprecian se probará tomando la medida en otras posiciones.

En cuanto a las velocidades angulares se puede ver como prácticamente son cero en todos los ejes, ya que la velocidad angular más alta está en el eje X con algunas variaciones y el valor es $1.2 \times 10^{-3}$ rad/seg; estos valores pueden quedarse a cero completamente tras su paso por los filtros.

Cuando se coloca el módulo con la misma orientación pero en sentido contrario (ver figura 30), se ve como en los ejes X e Y la aceleración continúa siendo 0G, pero en el eje Z ha pasado a ser -0.46G. Esto no puede ser debido a una orientación distinta ya que, si así fuera, los ejes X e Y mostrarían una aceleración mayor, además, la fuente de esta aceleración tampoco puede ser por offset, ya que el offset es algo inherente al dispositivo y aparecería en todas las medidas, lo que no ocurre en el primer caso. Las velocidades angulares se mantienen prácticamente en 0 con algunas oscilaciones que mejoraran con los filtros. Ver figuras 31 y 32.

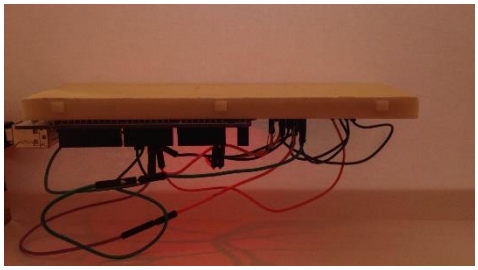


**Figura 30. Posición calibración GY81**

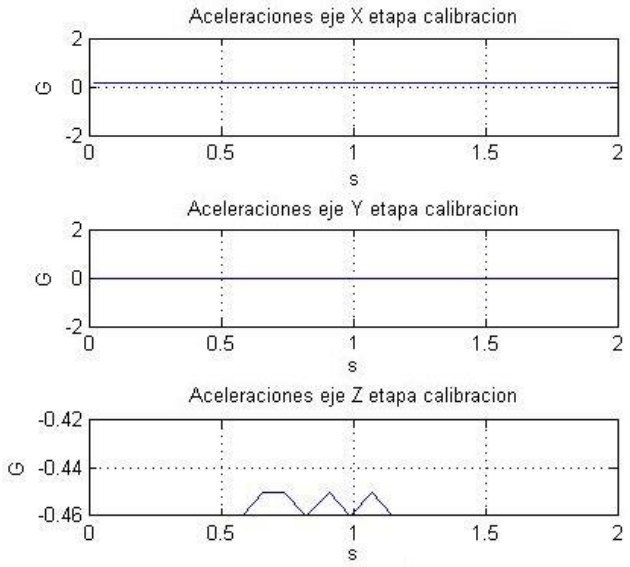



**Figura 31. Aceleraciones calibración GY81**

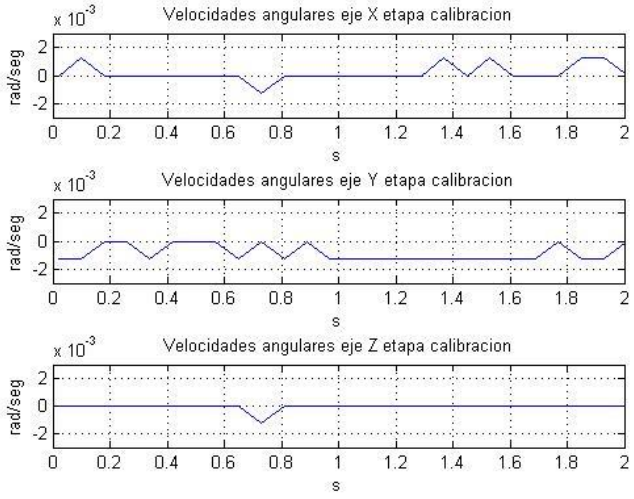



**Figura 32. Velocidades angulares calibración GY81**

Para detallar mejor las causas de estas diferencias en la aceleración se hace la prueba en otro eje. Ver figura 33 y 34.

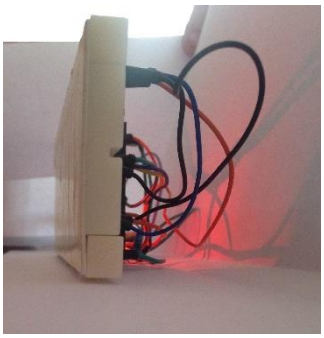


**Figura 33. Posición calibración GY81**

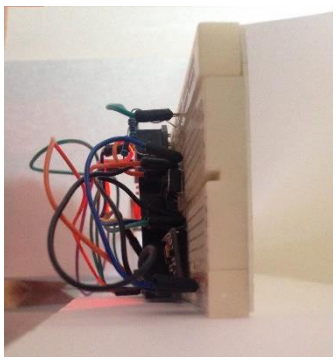


**Figura 34. Posición calibración GY81**

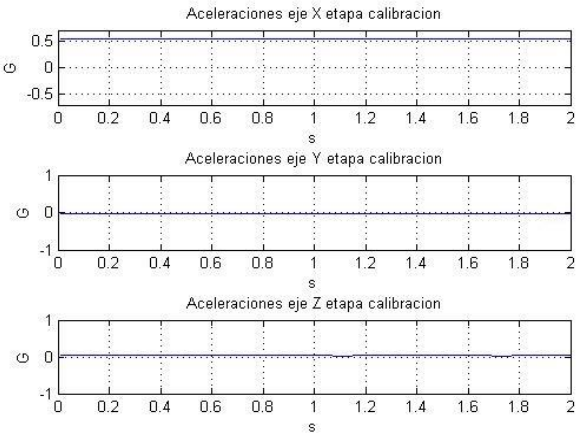



**Figura 35. Aceleraciones calibración GY81**

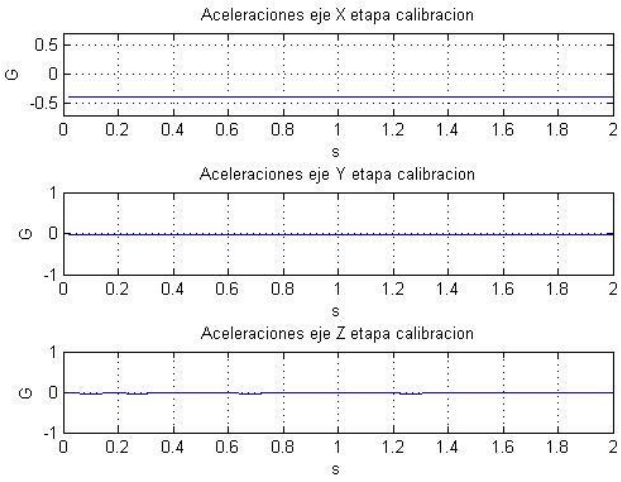



**Figura 36. Aceleraciones calibración GY81**

En las figuras 35 y 36 se puede observar como la aceleración de la gravedad recae sobre el eje X, aunque al igual que en las dos primeras imágenes, el módulo de la aceleración en X positiva no es el mismo que la negativa aun teniendo la misma orientación. Tras diversas pruebas en todos los ejes los resultados son similares, lo que lleva a pensar que el dispositivo no tiene el mismo rango en aceleraciones negativas como en positivas. A pesar de esto, calibraremos y filtraremos la señal para mitigar este error además de los errores de oscilaciones. A partir de ahora los resultados se mostrarán con el dispositivo en la posición de la figura 27.

Una vez calibrados, y continuando en estado de reposo las aceleraciones filtradas resultan de la forma que aparece en las figuras 37, 38 y 39.

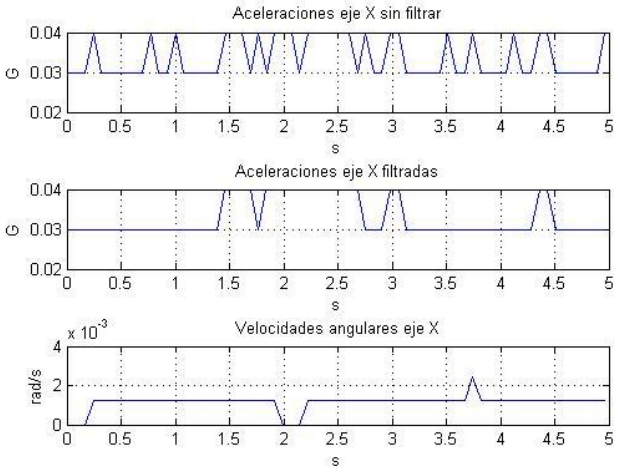



**Figura 37 Aceleraciones reposo eje X GY81**

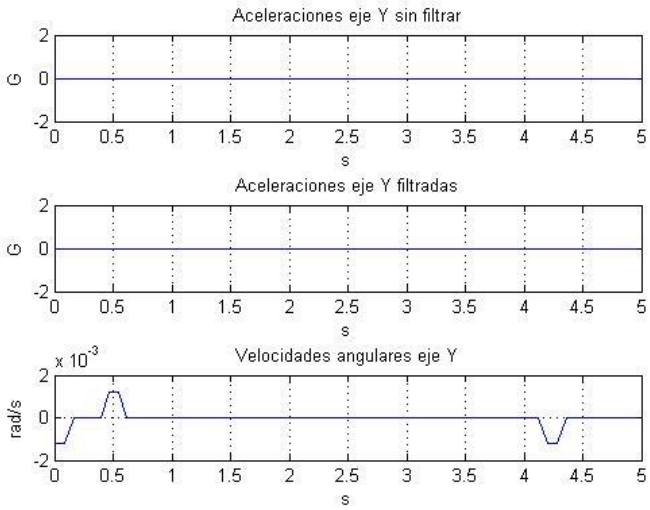



**Figura 38. Aceleraciones reposo eje Y GY81**

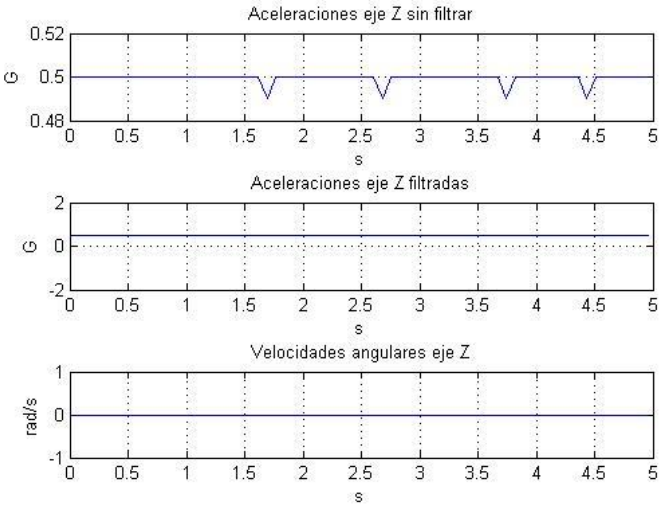



**Figura 39. Aceleraciones reposo eje Z GY81**

Tras estas pruebas se puede observar como las medidas filtradas contienen menos oscilaciones y ruido que las no filtradas, a pesar de que aún siguen quedando algunos errores residuales, lo que dificultará el cálculo de datos a partir de las medidas.

En las siguientes pruebas utilizaremos la teoría explicada en el Capítulo 4 con los ángulos de pitch y roll, por lo que comprobaremos la exactitud del giroscopio en movimientos, ya que anteriormente sólo se ha mostrado en estado de reposo. Las medidas que se realizarán serán en la posición de la figura 27 y haciendo un movimiento del ángulo roll (movimiento en el eje x).

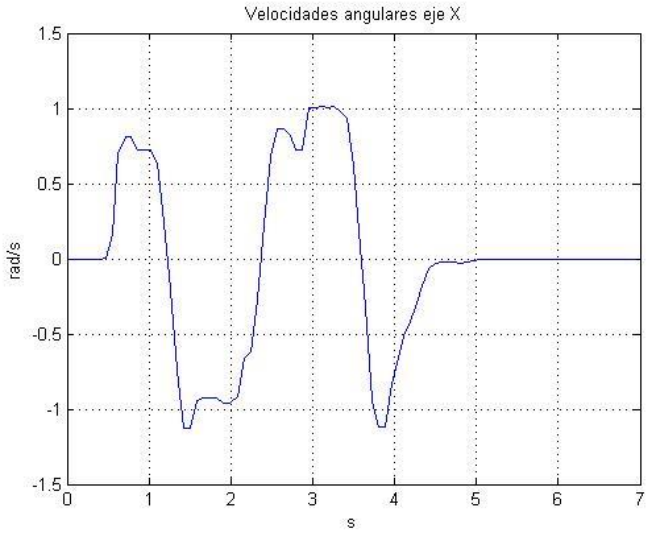



**Figura 40. Velocidades angulares movimiento GY81**

En la figura 40 se puede ver como el dispositivo se mueve a cada lado para después dejarlo en reposo con una exactitud bastante aceptable. Con esto, podemos concluir que es posible utilizar el giroscopio para hallar los ángulos pitch y roll y así mejorar la calidad de las medidas de la aceleración.

La última prueba con este sensor incluirá la corrección de aceleraciones mediante los ángulos pitch y roll en una situación de movimiento del dispositivo y el cálculo de la distancia recorrida tras mejorar todo lo posible la calidad de la aceleración. Para ello, se aplicará una aceleración inicial al dispositivo para que recorra una distancia de 30 centímetros. El resultado se muestra en la figura 41.

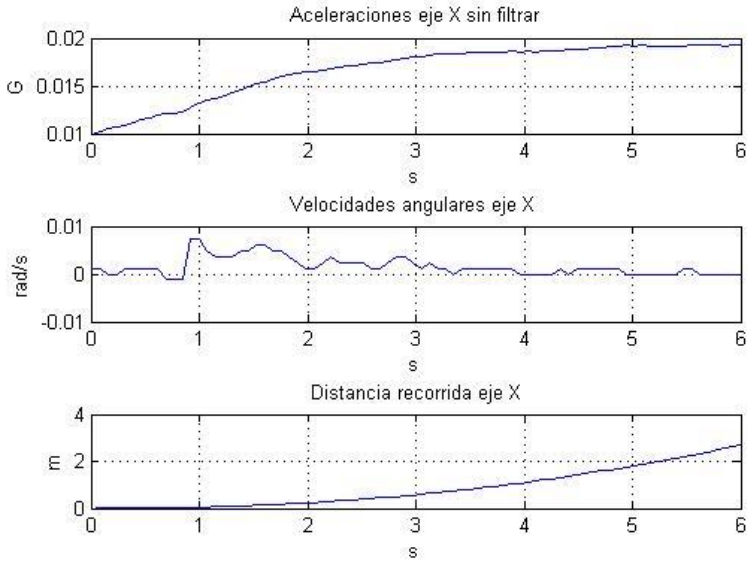



**Figura 41. Distancia recorrida GY81**

Tras los filtros y las compensaciones se puede ver como al principio se ha aplicado una aceleración inicial y después la aceleración se ha mantenido constante. A pesar de ello, al hallar la distancia con doble integral los pequeños errores se acumulan, lo que resulta en una distancia recorrida completamente alejada de la realidad.

# Capítulo 7. Conclusiones y propuesta de trabajo futuro

## 7.1 Conclusiones

Del trabajo realizado en el presente proyecto de fin de grado se pueden realizar las siguientes consideraciones a modo de conclusiones:

- En aceleraciones bajas los sensores no proporcionan la suficiente fiabilidad como para utilizarlos en medidas reales, debido a la gran sensibilidad que experimentan lo que hace que aparezcan multitud de oscilaciones y ruido que dificulta enormemente la distinción de medidas reales.
- El módulo GY81 da una mejor respuesta que el acelerómetro MMA7455, además de disponer de giroscopio, lo que nos proporciona más información acerca del movimiento para poder combatir los errores de los que hemos hablado durante el estudio.

Por lo tanto, como conclusión final podemos indicar que desgraciadamente no es posible calcular la distancia con un error aceptable, ya que ambos sensores proporcionan una distancia completamente alejada de la real.

## 7.2 Propuesta de trabajo futuro

Una posible continuación de este estudio sería la implantación de alguno de estos sensores junto con otros con características diferentes para proporcionar más información acerca del movimiento y así poder conseguir una mejor calidad de procesamiento del mismo, tanto para hallar aceleraciones, como distancias u otras medidas relacionadas con el movimiento. Algunos ejemplos de sensores podrían ser el sensor ultrasónico, el cual calcula la distancia mediante el uso del efecto Doppler, o una cámara con tratamiento de imagen, para poder analizar perfectamente todas las características del movimiento.

Otra vía de estudio podría ser la implementación del filtro Kalman en lugar del filtro complementario. Sus resultados son similares el filtro Kalman, pero éste último proporciona una respuesta más precisa, aunque también una más difícil implementación y un alto coste computacional, lo cual es una característica poco ventajosa cuando se trabaja con Arduino.

# Bibliografía

[1] Arduino. Home Page Arduino. https://www.arduino.cc/

[2] Arduino. MMA7455 Accelerometer. http://playground.arduino.cc/Main/MMA7455

[3] Datasheet MMA7455 Accelerometer. https://www.nxp.com/files/sensors/doc/data_sheet/MMA7455L.pdf

[4] Datasheet Real Accelerometer. http://akizukidenshi.com/download/ds/parallax/28526-MMA7455-3axisAccel-v1.1.pdf

[5] Datasheet BMA180. http://irtfweb.ifa.hawaii.edu/~tcs3/jumpman/jumppc/1107-BMA180/BMA180-DataSheet-v2.5.pdf

[6] Datasheet ITG3205. http://www.geeetech.com/Documents/ITG-3205-00-01.4.pdf

[7] Wikipedia. https://es.wikipedia.org

# Anexo

## Glosario

**MEMS:** *MicroElectroMechanical Systems.* Sistemas microelectromecánicos, usado en acelerómetros, giroscopios, fibra óptica y otros dispositivos de pequeño tamaño.

**Arduino:** Plataforma tecnológica libre compuesta por un microcontrolador y un entorno de desarrollo.

**IDE:** *Integrated Drive Electronics*. Entorno de desarrollo integrado compuesto principalmente por un editor de código y un depurador para la creación de software.

**I2C:** Interfaz de comunicación entre las distintas partes de un dispositivo o con otro elemento electrónico externo.

**SPI:** Interfaz de comunicación al igual que I2C con distintas características.

**IMU:** Dispositivo electrónico compuesto por acelerómetros y giroscopios usado en sistemas inerciales.

**Eje roll:** Rotación del eje x

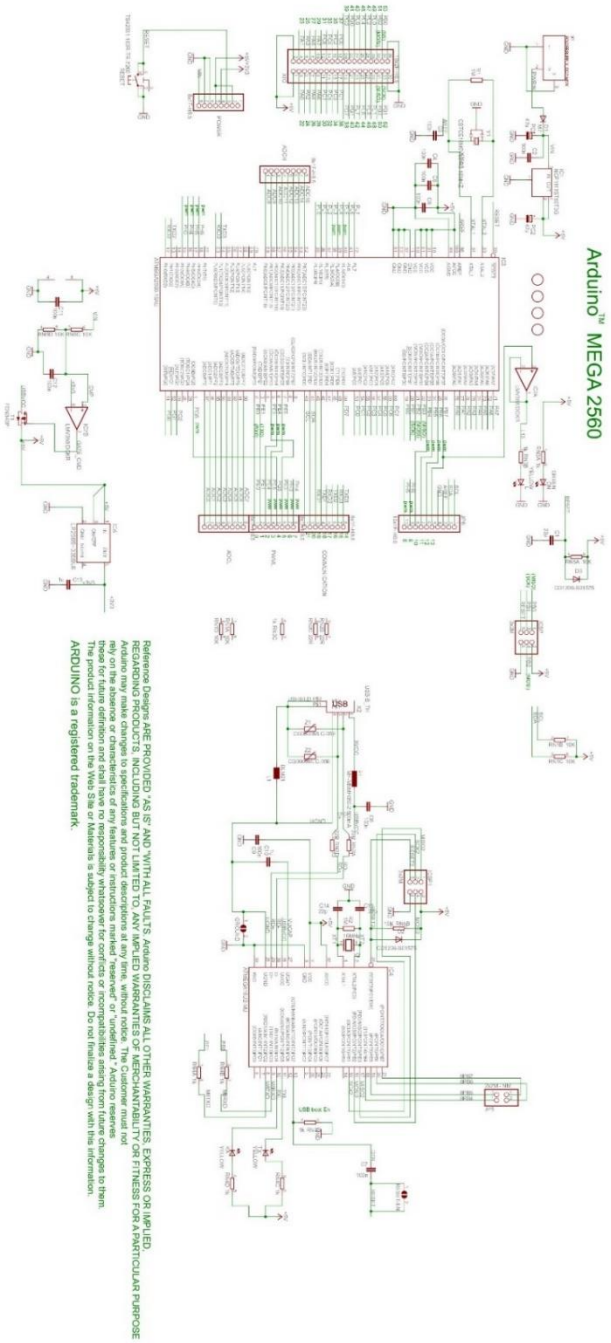
**Eje pitch:** Rotación del eje y.

**Eje yaw:** Rotación del eje z.

**Drift**: Errores acumulativos en medidas.

**UART:** *Universal Asynchronous Receiver Transmitter*. Circuito integrado que controla las comunicaciones serie.

**Esquema conexionado Arduino Mega 2560**



**Librería MMA7455**

```
#include "MMA_7455.h"
#define MMA_7455_ADDRESS 0x1D //I2C Address for the sensor
#define MMA_7455_MODE_CONTROLL 0x16 //Call the sensors Mode Control

#define MMA_7455_2G_MODE 0x05 //Set Sensitivity to 2g
#define MMA_7455_4G_MODE 0x09 //Set Sensitivity to 4g
#define MMA_7455_8G_MODE 0x01 //Set Sensitivity to 8g

#define X_OUT 0x06 //Register for reading the X-Axis
#define Y_OUT 0x07 //Register for reading the Y-Axis
#define Z_OUT 0x08 //Register for reading the Z-Axis

MMA_7455::MMA_7455()
{
  Wire.begin();
}

void MMA_7455::initSensitivity(int sensitivity)
{
  delay(1000);
  Wire.beginTransmission(MMA_7455_ADDRESS);
  Wire.write(MMA_7455_MODE_CONTROLL);
  if(sensitivity == 2)
  {
    Wire.write(MMA_7455_2G_MODE);  //Set Sensitivity to 2g Detection
  }
  if(sensitivity == 4)
  {
    Wire.write(MMA_7455_2G_MODE);  //Set Sensitivity to 4g Detection
  }


  if(sensitivity == 8)
  {
    Wire.write(MMA_7455_2G_MODE);  //Set Sensitivity to 8g Detection
  }
  Wire.endTransmission();
  delay(1000);
}

void MMA_7455::calibrateOffset(char x_axis_offset, char y_axis_offset, char z_axis_offset)
{
  _x_axis_offset = x_axis_offset;
  _y_axis_offset = y_axis_offset;
  _z_axis_offset = z_axis_offset;
}
 unsigned char MMA_7455::readAxis(char axis)
{
```

```
  Wire.beginTransmission(MMA_7455_ADDRESS);
  if(axis == 'x' || axis == 'X')
  {
    Wire.write(X_OUT);
  }
  if(axis == 'y' || axis == 'Y')
  {
    Wire.write(Y_OUT);
  }
  if(axis == 'z' || axis == 'Z')
  {
    Wire.write(Z_OUT);
  }
  Wire.endTransmission();
  Wire.beginTransmission(MMA_7455_ADDRESS);
  Wire.requestFrom(MMA_7455_ADDRESS, 1);
  if(Wire.available())
  {
    _buffer = Wire.read();
  }
  if(axis == 'x' || axis == 'X')
  {
    _buffer = _buffer + _x_axis_offset;
  }
  if(axis == 'y' || axis == 'Y')
  {
    _buffer = _buffer + _y_axis_offset;
  }
  if(axis == 'z' || axis == 'Z')
  {
    _buffer = _buffer + _z_axis_offset;
  }

  return _buffer;
}
```

## Librería BMA180

```
#include <Arduino.h>
#include <wiring_private.h>

#include "BMA180.h"

void BMA180::bma180Write(char add,char data)
{
    Wire.beginTransmission(BMA180_ADDRESS);     // address of the accelerometer
    Wire.write(add);
    Wire.write(data);
    Wire.endTransmission();
}

void BMA180::bma180SoftReset()
{
    bma180Write(0x10,0xB6);
    delay(100);
}

void BMA180::bma180EnableWrite()
{
    bma180Write(0x0D,0x10);
    delay(10);
}

void BMA180::bma180GetIDs(int *id, int *version)
{
    Wire.beginTransmission(BMA180_ADDRESS);     // address of the accelerometer
    Wire.write(0x00);                           // set read pointer to data
    Wire.endTransmission();
    Wire.requestFrom(BMA180_ADDRESS, 2);

    *id = Wire.read();
    *version= Wire.read();


}

void BMA180::bma180SetGSensitivty(GSENSITIVITY maxg) //1, 1.5 2 3 4 8 16
{
    Wire.beginTransmission(BMA180_ADDRESS);     // address of the accelerometer
    Wire.write(0x35);                           // read from here
    Wire.endTransmission();
    Wire.requestFrom(BMA180_ADDRESS, 1);
    byte data = Wire.read();
    Wire.beginTransmission(BMA180_ADDRESS);     // address of the accelerometer
    Wire.write(0x35);
    Wire.write((data & 0xF1) | maxg<<1);// range +/- 2g
    Wire.endTransmission();

    gSense = maxg;
}
```

```
void BMA180::bma180ReadAccel()
{
    Wire.beginTransmission(BMA180_ADDRESS);     // address of the accelerometer
    Wire.write(0x02);                           // set read pointer to data
    Wire.endTransmission();
    Wire.requestFrom(BMA180_ADDRESS,7);

    // read in the 3 axis data, each one is 16 bits
    // print the data to terminal
    //Serial.print("Accelerometer: X = ");
    x = Wire.read();
    x |= Wire.read() << 8;
    x = x >> 2;
    y = Wire.read();
    y |= Wire.read() << 8;
    y = y >> 2;
    z = Wire.read();
    z |= Wire.read() << 8;
    z = z >> 2;
    temp = Wire.read();
}

float BMA180::bma180GetgSense()
{
    switch(gSense)
    {
        case G1:    return 1.0 * 0.0001250;
        case G15:   return 1.5 * 0.0001875;
        case G2:    return 2.0 * 0.0002500;
        case G3:    return 3.0 * 0.0003750;
        case G4:    return 4.0 * 0.0005000;
        case G8:    return 8.0 * 0.0009900;
        case G16:   return 16.0 * 0.0019800;
    }
short BMA180::bma180FloatX()
{   return x;
}
short BMA180::bma180FloatY()
{   return y;
}
short BMA180::bma180FloatZ()
{   return z;
}
float BMA180::bma180GravityX()
{float data = bma180GetgSense();
    return x * data;
}
float BMA180::bma180GravityY()
{   float data = bma180GetgSense();
    return y * data;
}
float BMA180::bma180GravityZ()
{   float data = bma180GetgSense();
    return z * data;
}
float BMA180::bma180Temp()
{   return map((int8_t)temp,-128,127,-400,875)/10.0;;
}
```

**Librería ITG3205**

```
#include <Arduino.h>
#include "ITG3205.h"

void ITG3205::itg3205initGyro()
{
   Wire.beginTransmission(ITG3205_Address);
   Wire.write(0x3E);
   Wire.write(0x00);
   Wire.endTransmission();

   Wire.beginTransmission(ITG3205_Address);
   Wire.write(0x15);
   Wire.write(0x07);
   Wire.endTransmission();

   Wire.beginTransmission(ITG3205_Address);
   Wire.write(0x16);
   Wire.write(0x1E);   // +/- 2000 dgrs/sec, 1KHz, 1E, 19
   Wire.endTransmission();

   Wire.beginTransmission(ITG3205_Address);
   Wire.write(0x17);
   Wire.write(0x00);
   Wire.endTransmission();
}

void ITG3205::itg3205ReadGyro()
{
    Wire.beginTransmission(ITG3205_Address);    // address of the accelerometer
    Wire.write(0x1B);                           // set read pointer to data
    Wire.endTransmission();
    Wire.requestFrom(ITG3205_Address,8);

    temp = Wire.read() << 8;
    temp |= Wire.read();
```

```
    x = Wire.read() << 8;
    x |= Wire.read();
    y = Wire.read() << 8;
    y |= Wire.read();
    z = Wire.read() << 8;
    z |= Wire.read();

    x = x - g_offx;
    y = y - g_offy;
    z = z - g_offz;
}

void ITG3205::itg3205CalGyro()
{
    int tmpx = 0;
    int tmpy = 0;
    int tmpz = 0;
    g_offx = 0;
    g_offy = 0;
    g_offz = 0;

    for (char i = 0;i<10;i++)
    {
        delay(10);
        itg3205ReadGyro();
        tmpx += x;
        tmpy += y;
        tmpz += z;
    }
    g_offx = tmpx/10;
    g_offy = tmpy/10;
    g_offz = tmpz/10;
}
float ITG3205::itg3205GyroX()
{   //float data = bma180GetgSense();
    return x / 14.375;
}
float ITG3205::itg3205GyroY()
{   //float data = bma180GetgSense();
    return y / 14.375;
}
float ITG3205::itg3205GyroZ()
{   //float data = bma180GetgSense();
    return z / 14.375;
}
float ITG3205::itg3205Temp()
{   return (35+((temp+13200) / 280));
}
```

## Código Arduino MMA7455

```
#include <Wire.h> //Include the Wire library
#include <MMA_7455.h> //Include the MMA_7455 library

MMA_7455 mySensor = MMA_7455();
int lectura;
char Ax=0;
char Ay=0;
char Az=0; //Aceleraciones
float t=0;
double t0 = 0; // Almacena el estado del reloj interno en milisegundo
double t1 = 0; // Almacena el estado del reloj interno en milisegundo
double t2=0;
//Pulsador
const int boton=6;
int estadoboton;
int contadorboton=0;
int estadoanterior=HIGH;  //Lectura previa pin
long tiempo_pulsacion = 0;  //La ultima vez que el pin de entrada cambio
long retardo_pulsacion = 50;  //Tiempo en el que no va a cambiar de valor la entraa

void setup()
{
  pinMode(boton,INPUT);
  Serial.begin(9600); //Abre el puerto serie con 9600 baudios de velocidad de com

  mySensor.initSensitivity(2); //Sensitividad 2,4,8g
 mySensor.calibrateOffset(0,0,0);
  //mySensor.calibrateOffset(-3,14,-8);


 }

 void loop()
 {
   Pulsador();  //Funcion pulsador

   if(contadorboton==1){      //Si se ha pulsado una vez comienza a transmitir
    t0=millis();              //Primer retardo sistema

    LeerAceleracion();        //Funcion aceleracion

    t1=millis();              //Segundo retardo sistema(despues de leer la aceleracion)
    t = t+((t1 - t0)/1000);   //Añadimos diferencia de retardos

    EnviarDatos();            //Funcion envia datos

    t2=millis();              //Tercer retardo sistema (tras enviar datos)

   t=t+((t2-t1)/1000);        //Añadimos retarddo del sistema
```

```
  }else{
    if(contadorboton==2){     //Detecta boton pulsado dos veces y para de transmitir
      contadorboton=0;
      t=0;                    //Inicializamos a cero el tiempo
      Serial.println("]");    //Enviamos caracter para que detecte el final de la transmision
    }
  }
}

void EnviarDatos(){             //Enviamos los datos por puerto serie
 Serial.print(Ax, DEC);         //1.Aceleracion X
 Serial.print(",");
 Serial.print(Ay, DEC);         //2.Aceleracion y
 Serial.print(",");
 Serial.print(Az,DEC);          //3.Aceleracion z
 Serial.print(",");
 Serial.print(t);               //4.tiempo
 Serial.println();
}

void LeerAceleracion(){          //Leemos aceleracion del sensor MMA7455
    Ax = mySensor.readAxis('x'); //Lee eje x (Lo mide en count/g; 63=1g)
    Ay = mySensor.readAxis('y'); //Lee eje y
    Az = mySensor.readAxis('z'); //Lee eje z

}

int Pulsador(){                  //Detectamos pulsación botón y eliminamos rebote

  int lectura=digitalRead(boton);

  if (lectura != estadoanterior) {  //Si el pulsador cambia (por ruido o pulsacion)
    tiempo_pulsacion=millis();      //Resetea el tiempo de rebote
  }
  if ((millis() - tiempo_pulsacion) > retardo_pulsacion) {  //Si la lectura ha estado mas tiempo que el
                                                            //que el retardo inicial del sistema
    if(lectura!=estadoboton){         //Acutalizamos estado si ha cambiado
      estadoboton=lectura;
      if (estadoboton==LOW){          //Si se ha pulsado activamos el contador de pulsaciones
        contadorboton++;
        }
    }
  }
  estadoanterior=lectura; //En el siguiente ciclo este será el estado anterior del pulsador
  return (contadorboton); //Devolvemos el valor del contador de pulsaciones
}
```