



Universidad Nacional del Nordeste  
Facultad de Ciencias Exactas y  
Naturales y Agrimensura



# PROYECTO FINAL

“Sistema Inalámbrico de Registro de  
Parámetros de la Marcha Humana”

- **Alumno:** Vera, Lautaro Juan Bautista
- **Profesor Titular:** Mg. Lombardero, Oscar Guillermo
- **Año:** 2017

## Agradecimientos

Durante el camino transitado para desarrollar el trabajo que se presenta a continuación, existieron diversos obstáculos, los cuales no se hubieran podido sortear sin el apoyo incondicional de aquellas personas que me acompañaron siempre en el proyecto.

Del punto de vista académico, el rol que el Mg. Oscar Guillermo Lombardero cumplió como mi Director en este trabajo de investigación dentro del GRIER (Grupo de Ingeniería en Rehabilitación) fue fundamental para un correcto enfoque, y así el trabajo tenga un mayor impacto en el área de Rehabilitación y, por lo tanto, se agradece mucho su tutela.

Parte de los inconvenientes surgieron durante la etapa constructiva del proyecto, que por ser inalámbrico, debía ser pequeño. Esto exigía tener cierta técnica y experiencia en implementación de circuitos, por lo que para el dispositivo final, consulte al Lic. Eduardo Ricciardi, aportando en dicha etapa, por lo cual estoy muy agradecido.

Finalmente, tengo que expresar mi gratitud hacia mi familia, especialmente a mi padre, Lic. Gustavo Vera, y a mi madre, Susana Saade. Ambos fueron los pilares en los que me sostuve para afrontar tanto frustraciones personales como académicas, y si este proyecto llegó a buen puerto, es en gran parte gracias a ellos.

## Resumen

Con el auge de los Sistemas Microelectromecánicos (MEMS), la relación capacidad/tamaño de los sensores tuvo un crecimiento exponencial, tomando un papel principal en la investigación de medicina y rehabilitación, como por ejemplo en el registro del movimiento asociado al cuerpo humano.

En el presente trabajo se desarrolló un dispositivo inalámbrico capaz de capturar el movimiento asociado a la marcha humana, y determinar ciertos parámetros asociados a ella, como el ángulo de la rodilla.

El trabajo consta de dos etapas: la primera consistió en caracterizar el comportamiento de un acelerómetro y un giróscopo, ambos de 3 ejes, integrados en la IMU (Inertial Measurement Unit) MPU-6050 de InvenSense™, para determinar si son capaces de actuar como indicadores de posición para este propósito; y la segunda se basó, comprobada la hipótesis de la primer etapa, en desarrollar el dispositivo final con sensores de posición en el muslo y la pierna para registrar la marcha.

En la etapa de caracterización se desarrolló un prototipo que consistía de una sola IMU, solidario a la placa, ya que el propósito era realizar ensayos dinámicos y estáticos de linealidad, exactitud y precisión.

En la etapa final, se utilizaron dos IMUs que, mediante cables planos, se extendieron hacia los miembros inferiores para que la determinación de la marcha y en especial el ángulo de la rodilla, dependa solamente de la posición de los miembros y no de factores externos.

Finalmente, estos parámetros son de utilidad para el profesional de la salud especializado en Rehabilitación. En particular, el ángulo de la rodilla durante la marcha, permite obtener la curva del ciclo del ángulo de flexión y extensión de rodilla durante la marcha para un paciente dado, y contrastando con la curva teórica y sus bandas de confianza, se pueden diagnosticar patologías asociadas a la rodilla.

# Índice

Agradecimientos .....	1
Resumen.....	2
Índice .....	0
Introducción .....	1
Objetivos .....	1
Fundamentos Teóricos.....	2
Acelerómetro .....	2
Tipos de Acelerómetros .....	3
Giróscopo .....	5
Giróscopos Vibratorios de Efecto Coriolis (CVG).....	6
Ángulos de Navegación .....	8
Adquisición de Datos.....	12
Sensores o Transductores .....	14
Acondicionamiento de la señal .....	14
Amplificadores operacionales.....	14
Amplificador de instrumentación .....	15
Los aisladores .....	15
Los Multiplexores .....	15
Sample & Hold.....	16
Conversor Analógico Digital A/D .....	16
Conversor Digital Analógico D/A.....	17
Micropresesadores .....	17
Protocolos de Comunicaciones .....	18
Protocolo I <sup>2</sup> C.....	19
Protocolo SPI.....	26
La Marcha Humana .....	29
Materiales y Métodos .....	39
Materiales Empleados.....	39
Errores inherentes.....	45
Circuitos Implementados .....	<b>¡Error! Marcador no definido.</b>
Técnicas Empleadas .....	46
Procesamiento de Datos .....	49
Ensayos.....	<b>¡Error! Marcador no definido.</b>

Ensayos.....	52
Resultados .....	59
Conclusión .....	70
Bibliografía .....	71

# Introducción

## Objetivos

Los estudios de biomecánica suponen un complemento fundamental para el profesional de la salud a la hora de evaluar cuantitativamente la capacidad funcional de los miembros involucrados en el movimiento del cuerpo humano.

Para poder precisar un diagnóstico correcto y elaborar un tratamiento óptimo, se debe recurrir a un enfoque interdisciplinario, que involucre tanto las ciencias médicas como los desarrollos particulares de ingeniería.

Cualquier cuerpo en movimiento emite señales, las cuales pueden ser captadas por diversos sensores para poder registrar la dinámica de dicho movimiento.

El objetivo de este trabajo es establecer si es posible utilizar sensores iniciales, más precisamente un acelerómetro y un giróscopo (MPU-6050), como indicadores de posición, caracterizarlos, estimar el alcance de los resultados y una vez hecho esto, desarrollar bajo dicho alcance un dispositivo inalámbrico capaz de registrar el movimiento asociado a la marcha humana, para posteriormente determinar parámetros cinemáticos involucrados, particularmente el ciclo del ángulo de flexión/extensión de rodilla.

A priori se conoce por el principio de funcionamiento de dichos sensores, que su capacidad está limitada a describir la posición de un cuerpo respecto a un estado inicial de reposo. Es decir, se pueden precisar inclinaciones, rotaciones pero no se puede precisar si el cuerpo se desplazó de un punto a otro. Esto no supone un problema, ya que este tipo de sensores apuntan a ensayos de biomecánica que se enfocan en rotaciones e inclinaciones que se producen dentro de un movimiento aproximadamente uniforme, y resulta indistinto en qué punto de la trayectoria se encuentra el cuerpo.

El dispositivo final dispone de una tarjeta microSD para almacenar los datos capturados inalámbricamente, para luego introducir la tarjeta en un ordenador y mediante una GUI (Graphic User Interface) de MATLAB™ obtener

las curvas de la marcha humana del sujeto bajo estudio. Cabe destacar que el proceso de obtención del ciclo del ángulo de flexión/extensión de la rodilla durante la marcha no se realiza en dicha GUI, sino en un código de MATLAB™ desarrollado en un archivo .m de manera separada.

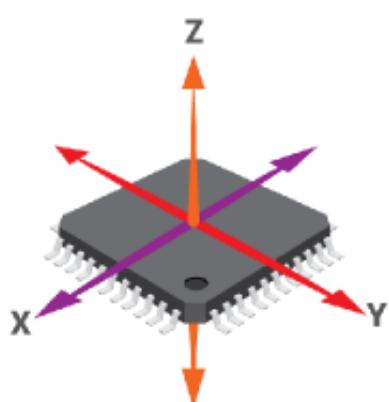
## Fundamentos Teóricos

### Acelerómetro

Según se describe en [1], un acelerómetro es un dispositivo que mide aceleración. La mayoría de ellos trabajan de manera indirecta: llevan una determinada cantidad de masa conocida, denominada *masa sísmica*, unida mecánicamente con el objeto que está siendo medido, de manera que cualquier aceleración que sufra el objeto medido, la masa sísmica debe experimentar la misma aceleración. Entonces el acelerómetro detectará la fuerza ejercida sobre la masa sísmica. El valor de la fuerza medida está relacionado con el valor de aceleración por la segunda ley de Newton:

$$a = \frac{F}{m_s}$$

Donde  $F$  es la fuerza medida por un transductor de fuerza y  $m_s$  una cantidad conocida fija de masa.



Por lo tanto, el transductor puede estar calibrado para leer las unidades de aceleración. Por ejemplo si el valor conocido de la masa sísmica es 0.5 kg, y si el transductor de fuerza detecta una fuerza de 2.0 newtons que está siendo ejercida sobre la masa sísmica, el transductor será calibrado para leer  $4.0 \text{ m/s}^2$ , en lugar de 2.0 newtons.

Figura 1

La unión mecánica de la masa sísmica con el objeto medido está dada por un vínculo elástico de baja deflexión, lo cual se puede concebir como un resorte muy rígido. El movimiento de la masa sísmica es restringido a una determinada dirección, por eso se dice que los

acelerómetros son direccionales, midiendo la aceleración en un eje de coordenadas. Los acelerómetros de 3 ejes repiten dicho sistema en cada eje de coordenadas  $x$ ,  $y$ ,  $z$  (figura 1).

### Tipos de Acelerómetros

Existen distintas clasificaciones de los acelerómetros, pero las más globales son según la salida que ofrecen y según su principio de funcionamiento.

Según su salida, se clasifican en analógicos o digitales.

Los acelerómetros analógicos tienen como salida una señal continua de tensión con un rango específico para cada acelerómetro, detallado en la hoja de datos. Estos valores de tensión generalmente no están calibrados en valores de aceleración en g. Además son más susceptibles al ruido eléctrico, y necesitan de conversores analógico/digital para su procesamiento, quedando determinada la exactitud de la medida por la resolución que posean dichos conversores.

Los acelerómetros digitales tienen como salida una señal discreta digital binaria, dada en valores crudos proporcionales a la aceleración. Tienen mayor

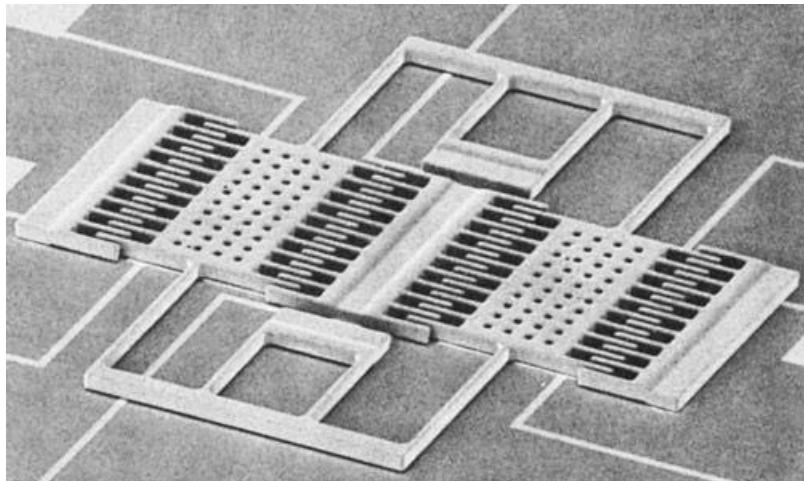


Figura 2

inmunidad al ruido, ya que realizan la conversión digital internamente en el mismo encapsulado, lo más rápido posible, reduciendo así la influencia de señales espurias. Como desventaja, se puede decir que su exactitud

está determinada por la resolución del conversor que se utiliza. Además se necesita conocer y proveer la interfaz de comunicación digital necesaria, correspondiente al protocolo de comunicación involucrado.

Por otro lado, los acelerómetros también pueden ser clasificados por el

principio físico de funcionamiento que los gobierna. Existen diversos tipos, piezoeléctricos, capacitivos, de efecto Hall, etc. Los más difundidos son los acelerómetros piezoeléctricos y los acelerómetros capacitivos, sobre todo porque el desarrollo de la tecnología MEMS (figura 2) permitió su implementación de manera compacta, pequeña, robusta a un bajo costo de producción. Los Sistemas Microelectromecánicos (Microelectromechanical Systems, MEMS) son

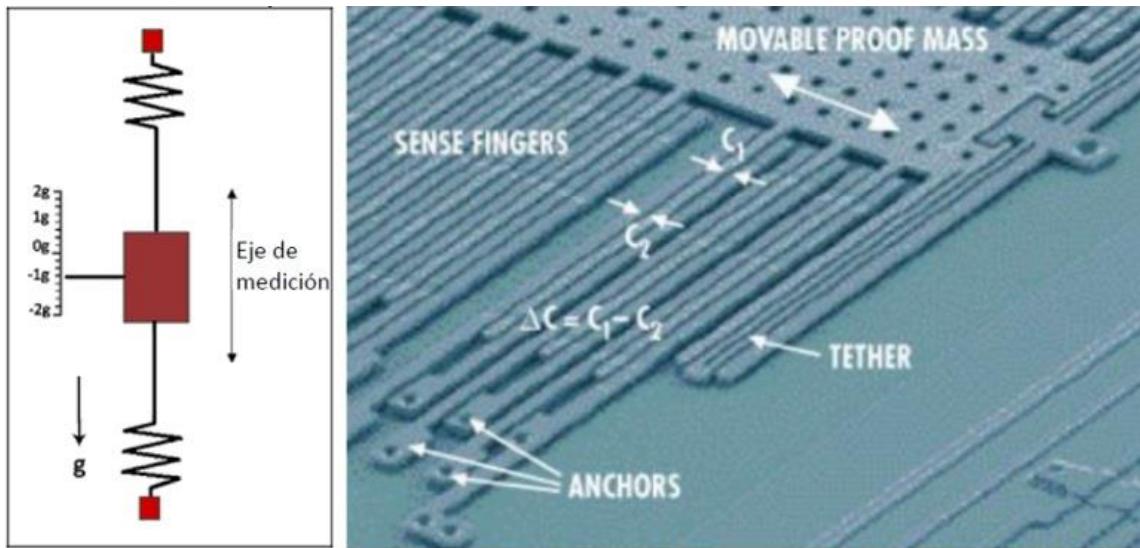


Figura 3

sistemas electromecánicos que van desde 1  $\mu\text{m}$  a 1mm de tamaño, consistiendo en circuitos integrados con partes tridimensionales e incluso piezas móviles.

Un acelerómetro piezoeléctrico se rige por el efecto del mismo nombre, el *efecto piezoeléctrico*, un fenómeno físico reversible por el cual fuerza o presión ejercida sobre ciertos cristales produce una tensión eléctrica. El fenómeno opuesto también se cumple, una tensión eléctrica deforma dichos cristales.

Generalmente se utilizan para medir vibraciones cuya frecuencia es inferior a 2 kHz, porque la frecuencia natural de oscilación del cristal es del orden de los 5 kHz. La masa sísmica está unida mecánicamente a un elemento piezoeléctrico, y a medida que el acelerómetro se mueve, la masa aplica mayor o menor presión sobre el cristal generando un voltaje alterno cuya amplitud es proporcional a las aceleraciones de las vibraciones.

Un acelerómetro capacitivo realiza la medición por variación de capacitancia (figura 3). En este caso la masa sísmica consiste en dos partes, una fija y otra móvil. La parte móvil está sujetada mediante muelles al objeto medido,

y cuando el objeto se mueve la aceleración asociada se transfiere a dicha parte móvil (figura 4). La parte móvil y la parte fija están polarizadas, de manera que en conjunto tienen una capacitancia asociada, el movimiento de la parte móvil tiene asociado una variación de la capacitancia, la cual es proporcional a la aceleración sufrida por el objeto.

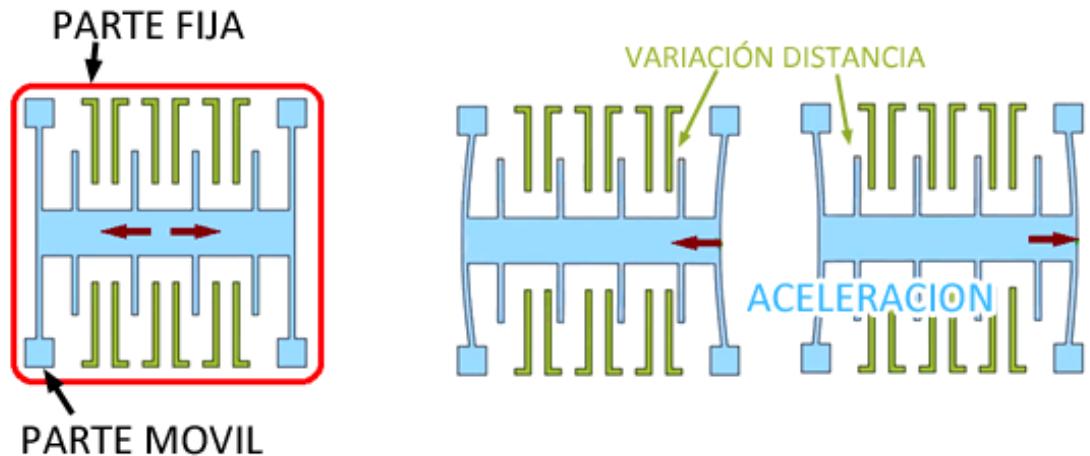


Figura 4

### Giróscopo

Un giróscopo o giroscopio es un dispositivo mecánico utilizado para medir la orientación en el espacio de un objeto (figura 5). Los giróscopos mecánicos se componen de un elemento rotatorio cuyo centro de masas se encuentra en el punto medio de su eje de rotación. Este elemento rotatorio (generalmente un disco) se encuentra suspendido por soportes Cardán, que son dos aros concéntricos cuyos ejes de rotación forman un ángulo recto. La suspensión Cardán permite mantener la orientación del eje de rotación del disco en el espacio, aun cuando su soporte se mueva, y así determinar la orientación del objeto asociado al giróscopo.



Figura 5

Existen distintos tipos de giróscopos (mecánicos, de anillo láser, de fibra óptica, de estructura vibrante). Tienen muchas aplicaciones, sobre todo en aeronavegación, cumpliendo el rol de indicadores de orientación durante el vuelo para, por ejemplo, corregir trayectorias.

En biomecánica, los sensores de movimiento que se utilizan generalmente son de tecnología MEMS ya que, como se dijo anteriormente, son compactos, pequeños y robustos, siendo menos invasivos y menos susceptibles a interferencias externas cuando se utilizan sobre un paciente.

#### Giróscopos Vibratorios de Efecto Coriolis (CVG)

Los giróscopos implementados con tecnología MEMS son los giróscopos de estructura vibrante o, como los define la IEEE, giróscopos vibratorios de *Efecto Coriolis* (Coriolis Vibratory Gyroscope, CVG). Estos giróscopos funcionan de manera similar a los *halterios* presentes en algunos insectos voladores. Sin estos halterios dichos insectos perderían estabilidad durante su vuelo. El principio físico subyacente del Efecto Coriolis es que un objeto vibrante tiende a continuar vibrando en el mismo plano incluso si este objeto rota.

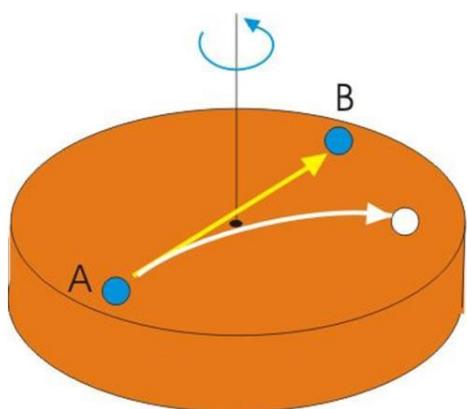


Figura 6

El Efecto Coriolis es el fenómeno que un observador fijo externo al sistema de rotación de un objeto, describe la aparición de una fuerza ficticia, similar a la *fuerza centrífuga*, denominada *fuerza de Coriolis* (figura 6). Un observador no inercial situado sobre una plataforma giratoria experimenta una fuerza aparente que le impide permanecer en reposo, la fuerza centrípeta. El observador percibe la fuerza centrífuga como resultado de la no inercialidad

Para permanecer en reposo, el observador aplica una fuerza, la *fuerza centrípeta*. El observador percibe la fuerza centrífuga como resultado de la no inercialidad

de su sistema de referencia. De la igual manera, la fuerza de Coriolis es la fuerza ficticia que el mismo observador no inercial percibiría que actúa sobre un objeto que se desplaza en línea recta sobre un radio perpendicular al eje de rotación. Para un observador fuera del sistema de rotación dicho objeto describiría una trayectoria recta, pero para el observador no inercial el objeto describiría una trayectoria curva. El causante de ese cambio de trayectoria es la fuerza de Coriolis, cuya expresión matemática es:

$$\vec{F}_c = -2m(\vec{\omega} \times \vec{v})$$

Donde  $F_c$  es la fuerza de Coriolis,  $m$  la masa del objeto,  $\omega$  la velocidad angular del sistema de rotación y  $v$  la velocidad lineal con la que se mueve el objeto sobre el radio.

Como sabemos la tierra es un cuerpo en rotación, y por lo tanto todo lo que este sobre ella actúa como un observador no inercial. Las masas de aire o de agua no son la excepción y son el ejemplo más notorio de manifestación del efecto Coriolis. El fenómeno se observa cuando dichas masas de aire o de agua se desplazan siguiendo meridianos terrestres, y su trayectoria y velocidad se ven modificadas por él. En efecto, los vientos o corrientes oceánicas que se desplazan siguiendo un meridiano se desvían acelerando en la dirección de giro (este) si van hacia los polos o al contrario (oeste) si van hacia el ecuador. Se puede añadir, que por consecuencia, en el Ecuador, no hay efecto de Coriolis.

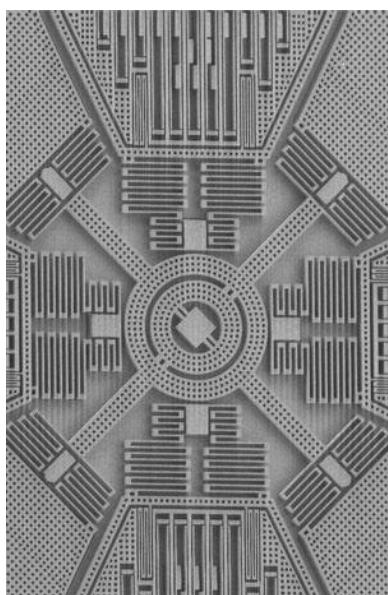


Figura 7

Finalmente, los CVG utilizan el efecto Coriolis para medir la velocidad angular de un objeto en rotación. A diferencia de los acelerómetros, estos dispositivos son diferenciales, es decir, no poseen una referencia absoluta sino que siempre mide ángulos relativos a una referencia arbitraria.

Para registrar el efecto de la fuerza Coriolis, un MEMS dispone de estructuras similares a las del acelerómetro (figura 7). Ciertas partes del sistema se someten a vibración por resonancia y el efecto

de la fuerza de Coriolis deforma la estructura, lo cuál puede ser medido por la variación de la capacitancia del sistema (figura 8). La salida que ofrecen este tipo de giróscopos pueden ser analógica o digital, siendo más frecuentes los digitales, ya que las señales crudas del giróscopos solo ofrecen velocidad angular, y mediciones como la posición o la orientación en el espacio requieren de un procesamiento posterior, que se facilita mediante una salida digital bajo un protocolo de comunicaciones como I<sup>2</sup>C o SPI. Son generalmente de 3 ejes, midiendo en cada instante la velocidad angular asociada a la rotación de cada eje x, y, z.

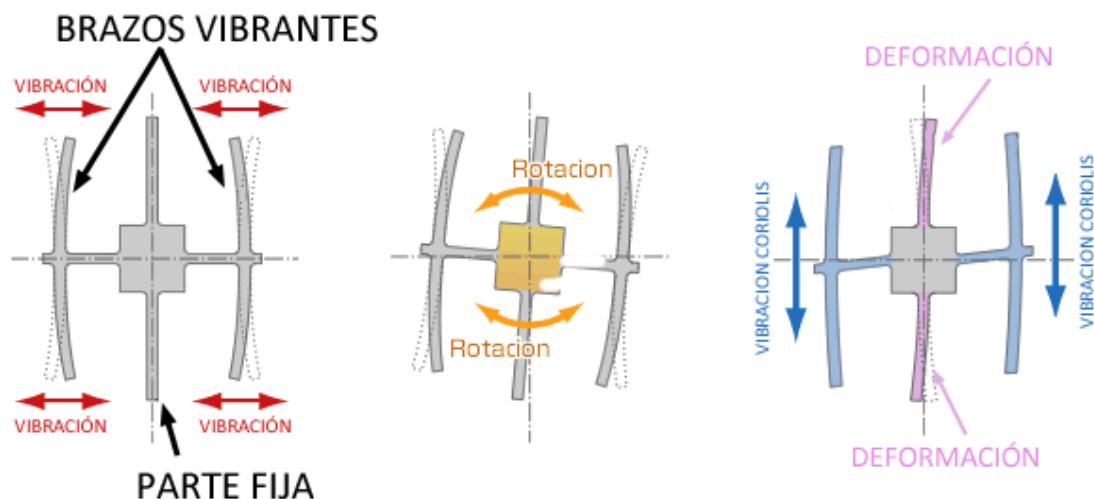


Figura 8

### Ángulos de Navegación

Los *ángulos de navegación*, también llamados en matemáticas *ángulos de Tait-Bryan* son un tipo de *ángulos de Euler*. Según la teoría matemática de las rotaciones[2], los ángulos de Euler constituyen un conjunto de tres coordenadas angulares que sirven para especificar la orientación de un sistema de referencia de ejes ortogonales, normalmente móvil, respecto a otro sistema de referencia de ejes ortogonales normalmente fijos.

Dados dos sistemas de coordenadas xyz y XYZ con origen común, es posible especificar la posición de un sistema en términos del otro usando tres ángulos  $\alpha$ ,  $\beta$  y  $\gamma$ .

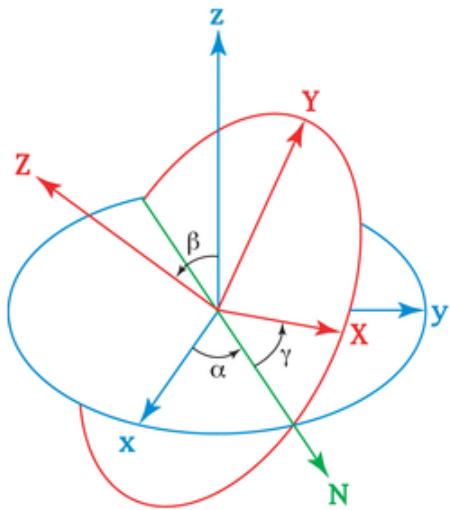


Figura 9

La definición matemática es estática y se basa en escoger dos planos, uno en el sistema de referencia y otro en el triedro rotado. En la figura 9, serían los planos xy y XY. Escogiendo otros planos se obtendrían distintas convenciones alternativas, las cuales se llaman de Tait-Bryan cuando los planos de referencia son no-homogéneos (por ejemplo xy y XY son homogéneos, mientras xy y XZ no lo son).

La intersección de los planos coordinados xy y XY escogidos se llama línea de nodos, y se usa para definir los tres ángulos:

- $\alpha$  es el ángulo entre el eje x y la línea de nodos.
- $\beta$  es el ángulo entre el eje z y el eje Z.
- $\gamma$  es el ángulo entre la línea de nodos y el eje X.

Los tres ángulos de Euler descritos son los valores de las tres rotaciones intrínsecas que describen el sistema. También se considera la notación:  $\alpha=\phi$ ,  $\gamma=\psi$  y  $\beta=\theta$ . Las rotaciones de Euler son los movimientos resultantes de variar uno de los ángulos de Euler dejando fijo los otros dos. Estos movimientos son:

- Precesión.
- Nutación.
- Rotación Intrínseca.

La orientación de un sistema de referencia se puede describir como la composición de estas tres rotaciones, y además, estas rotaciones tienen la

propiedad de ser conmutativas. Esta propiedad se observa intuitivamente en un soporte Cardán.

A partir de la relación entre los ángulos de Euler y el movimiento de los soportes Cardán, se puede probar que todo sistema de coordenadas puede ser descrito con los tres ángulos de Euler. Si llamamos  $[R]$  a la matriz de rotación tridimensional que representa la transformación de coordenadas desde el sistema fijo al sistema móvil, el teorema de Euler sobre rotaciones tridimensionales, afirma que existe una descomposición única en términos de los tres ángulos de Euler:

$$[R] = \begin{bmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & \sin \theta \\ 0 & -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} \cos \phi & \sin \phi & 0 \\ -\sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Tras cada rotación el sistema de referencia queda girado de la siguiente forma: el primer giro de ángulo  $\phi$  es al alrededor del eje  $Z_1$ , el segundo giro de ángulo  $\theta$  es al alrededor del eje  $X_2$  y el tercer giro de ángulo  $\psi$  es al alrededor del eje  $Z_3$ . La velocidad angular  $\Omega$  de un sólido rígido expresada en términos de los ángulos de Euler viene dada por:

$$[\Omega] = \begin{Bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{Bmatrix} = \begin{Bmatrix} \dot{\theta} \cos \psi + \dot{\phi} \sin \theta \sin \psi \\ -\dot{\theta} \sin \psi + \dot{\phi} \sin \theta \cos \psi \\ \dot{\psi} + \dot{\phi} \cos \theta \end{Bmatrix}$$

Finalmente, llegamos a los ángulos de navegación. Muchas veces en ingeniería aeronáutica se utiliza el nombre de ángulos de Euler para hablar de los ángulos que en geometría se conocen como ángulos de Tait-Bryan (por el matemático escocés Peter Guthrie Tait).

Estos ángulos se prefieren en aeronáutica porque le asignan un ángulo de inclinación cero a un avión en horizontal, a diferencia de los ángulos de Euler que le asignarían  $\pi/2$ . Estos ángulos también definen una rotación de forma única alrededor de cada uno de los ejes intrínsecos del objeto. Sin embargo, como ambas son formas de expresar la orientación de un cuerpo, existe una relación entre ellos; pudiéndose expresar unos en función de otros mediante una matriz de transformación.

Los ángulos de Tait-Bryan son tres coordenadas angulares que definen un triángulo rotado desde otro que se considera el sistema de referencia. Se definen matemáticamente de forma similar a los ángulos de Euler, pero en vez de usar como línea de nodos el corte entre dos planos homólogos (por ejemplo el XY es el homólogo del xy), se utilizan dos planos no homólogos (por ejemplo XY e yz).

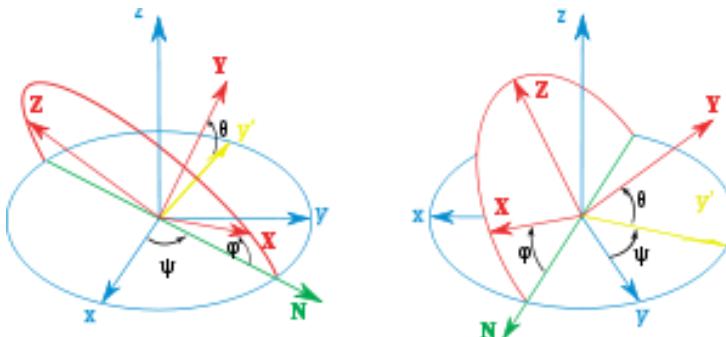


Figura 10

Por ejemplo, en la figura 10, que usa el convenio ZXY, se definen mediante los planos xy (plano perpendicular al eje "z" usado en la primera rotación) del sistema de referencia, y el plano ZX del sistema móvil (plano perpendicular al eje "Y" de la última rotación).

En aeronavegación estos ángulos adquieren gran importancia a la hora de determinar la orientación de las aeronaves. En este caso, los tres ángulos son el ángulo de guiñada ( $\psi$ ), el ángulo de elevación o cabeceo ( $\theta$ ) y ángulo de alabeo ( $\phi$ ). Estos tres ángulos son equivalentes a tres maniobras consecutivas. Dado un sistema de tres ejes fijos en el aeroplano, llamados eje de guiñada (yaw), de cabeceo (pitch) y de alabeo (roll), existen tres rotaciones principales, normalmente llamadas igual que el eje sobre el que se producen, que permiten alcanzar el sistema del aeroplano desde el sistema de referencia (figura 11). Tienen que venir dadas en ese orden y ser realizadas en ese orden, ya que el resultado final depende del orden en que se apliquen:

- *Cabeceo*: es una inclinación del morro del avión, o rotación respecto al eje ala-ala.
- *Alabeo*: rotación respecto de un eje morro-cola del avión.
- *Guiñada*: rotación intrínseca alrededor del eje vertical perpendicular al avión.

Son tres rotaciones intrínsecas, es decir, relativas al sistema móvil. Esto es útil por ejemplo cuando el piloto de un avión quiere describir una maniobra.

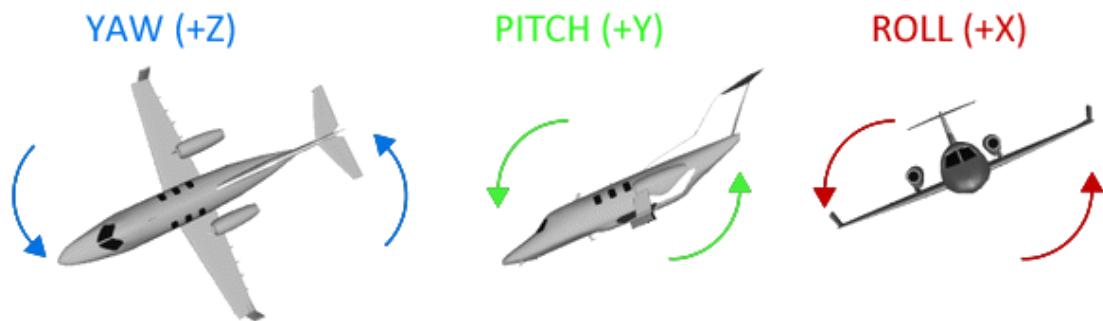


Figura 11

### **Adquisición de Datos**

La *Adquisición de Datos* o adquisición de señales consiste en la toma de muestras del mundo real (sistema analógico) para generar datos que puedan ser manipulados por un ordenador u otros dispositivos electrónicos (sistema digital). Consiste en tomar un conjunto de señales físicas, convertirlas en tensiones eléctricas y digitalizarlas de manera que se puedan ser procesadas por un ordenador (figura 12). Se requiere una etapa de acondicionamiento, que adecua la señal a niveles compatibles con el elemento que hace la transformación a señal digital. El elemento que hace dicha transformación es el módulo de digitalización o tarjeta de adquisición de datos (DAQ).

Un *Sistema de Adquisición de Datos* (DAQ) no es más que un equipo electrónico cuya función es el control o simplemente el registro de una o varias variables de un proceso o fenómeno físico cualquiera, y de forma general puede estar compuesto por los siguientes elementos:

- Sensores.
- Amplificadores operacionales.
- Amplificadores de instrumentación.
- Aisladores.
- Multiplexores analógicos.

- Multiplexores digitales.
- Circuitos Sample & Hold.
- Conversores A-D.
- Conversores D-A.
- Microprocesadores.
- Contadores.
- Filtros.
- Comparadores.
- Fuentes de potencia.

El DAQ debe tener una estructura y organización muy equilibrada que le permita su buen funcionamiento de ello depende de que el mismo rinda al máximo y sin ningún defecto.

## Sistema Digital de Adquisición de Datos

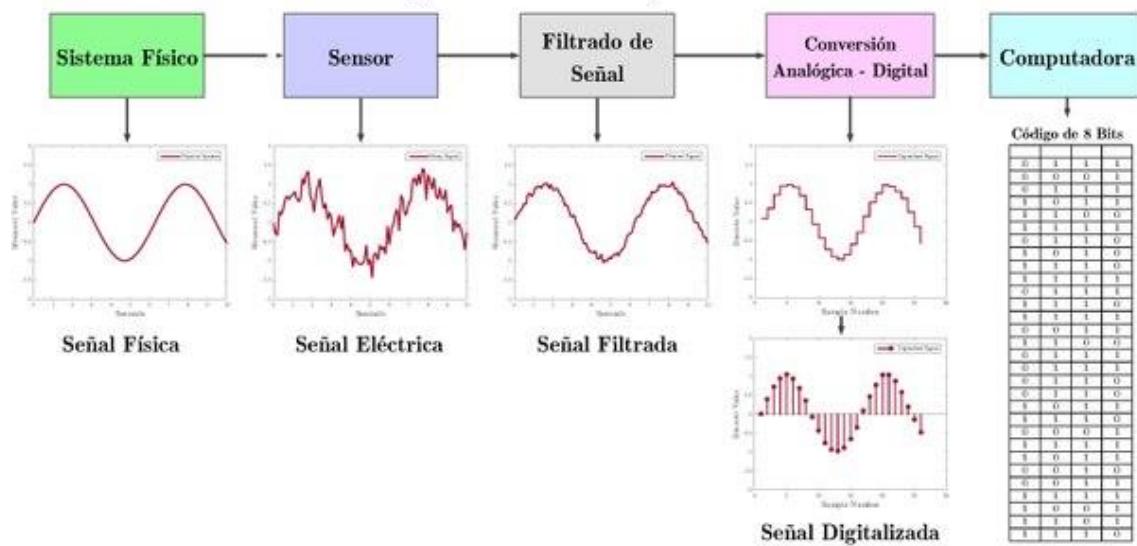


Figura 12

## Sensores o Transductores

Los sensores tienen un rol vital en todo DAQ ellos tienen la función de convertir la variable física que se desea registrar en una magnitud eléctrica (voltaje, corriente, resistencia, capacidad, Inductancia, etc.). Entre las magnitudes físicas más importantes a registrar tenemos: temperatura, humedad, presión, concentración, iluminación, flujo, posición, nivel, peso, etc. Diversas pueden ser las variables ambientales, industriales, biológicas, químicas, etc. que en un momento determinado podemos necesitar controlar, esto provoca que sean también numerosos los tipos de sensores así como su principio de funcionamiento, lo cual determina generalmente el costo de sensor que será necesario utilizar.

## Acondicionamiento de la señal

En todo DAQ o sistema donde sea usado un conversor A/D es muy importante el acondicionamiento previo de la señal que es suministrada al conversor, la esencia del acondicionamiento es hacer que el rango de variación real que experimentará la variable a medir se convierta en el rango máximo de voltaje de entrada que acepta el conversor A/D que se utiliza, o sea que el valor mínimo de la variable a medir imponga a la entrada del conversor el valor mínimo del voltaje que el acepta y el valor máximo de la variable a medir imponga el valor máximo de voltaje que el conversor admite. Paralelamente el acondicionamiento de la señal también implica la transformación de la señal entregada por el sensor de forma que siempre la magnitud final sea voltaje, además en el acondicionamiento se debe garantizar el filtrado de ruido no deseado en la variable medida.

La etapa acondicionadora está formada básicamente por amplificadores operacionales, comparadores de nivel y amplificadores de instrumentación.

## Amplificadores operacionales

En sus configuraciones básicas (inversora, no inversora, amplificadora, conversor de corriente a voltaje, etc.), son usados para garantizar que al conversor A/D le sea suministrado el rango máximo de voltaje y así el mismo

pueda dar el mayor número de combinaciones posibles.

### Amplificador de instrumentación

Puede alternadamente sustituir al amplificador operacional, siempre que la aplicación lo exija, pues los mismos tienen prestaciones superiores a los amplificadores operacionales normales, lo cual hace que sean más costosos. Entre las características de los amplificadores de instrumentación tenemos una impedancia de entrada infinita y una ganancia ajustable en ocasiones mediante una red resistiva de precisión externa o mediante resistores internos de precisión por interruptores o por software.

### Los aisladores

Son dispositivos de mucha importancia principalmente en sistemas médicos donde se requiere aislar completamente al paciente del equipo de medición con el fin de evitar que en caso de desperfectos del equipo los pacientes estén expuestos altos niveles de voltaje o corriente, también en equipos o instrumentos que manejen altas tensiones es necesario garantizar el aislamiento entre los instrumentos de medición y las fuentes de alta tensión. Entre los dispositivos más comunes son los opto-acopladores.

### Los Multiplexores

Los multiplexores ya sean analógicos o digitales son dispositivos que nos permiten multiplexar varias entradas en una única salida. Ellos nos permiten que para registrar varias señales diferentes podamos utilizar un único conversor A/D y con ello disminuir de forma considerable el costo e un DAQ. Generalmente los multiplexores se pueden dividir por el tipo de salida en simples y diferenciales o por el número de entradas en de 2, 4, 8 ó 16 entradas. El hecho de existir una gran variedad de multiplexores nos obliga a hacer una correcta selección según las exigencias de nuestro sistema, sobre la base de disminuir los costos del mismo. Los multiplexores diferenciales de mayor costo que los de salida simple, son usados normalmente cuando son utilizadas para multiplexar señales de naturaleza diferentes por ejemplo: temperatura, presión, concentración, etc. Los amplificadores de salida simple se recomiendan cuando se multiplexan señales

de naturaleza semejante: por ejemplo cuando registramos la temperatura en diferentes puntos. En esencia la diferencia entre los multiplexores de salida simple y diferencial está en que para los últimos, la señal de referencia (tierra) es también multiplexada lo cual no ocurre para los multiplexores de salida simple. En la medida que aumenta el número de entradas de un Multiplexor también aumenta su costo y el número de terminales de control que el mismo necesita, por lo cual es también muy necesario utilizar en una aplicación un Multiplexor con el número de entradas que se requiera.

### Sample & Hold

Es un dispositivo electrónico con dos posibilidades de trabajo: el modo *Sample* y el modo *Hold* (figura 13):

- Modo Sample: La señal pasa a la salida del dispositivo tal y como está en la entrada del mismo.
- Modo Hold: La salida se mantiene en el nivel de voltaje que existía en la entrada en el momento que la señal Hold fue activada.

El sample & hold debe ser utilizado cuando la señal de voltaje que entra a un conversor A/D varia en un nivel suficiente como para que el conversor cambie 1/2 bit menos significativo en un tiempo menor que el que el conversor necesita para hacer la conversión (figura 14).

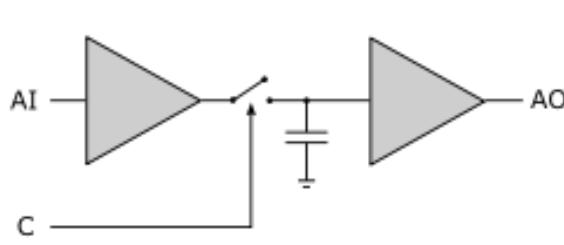


Figura 13

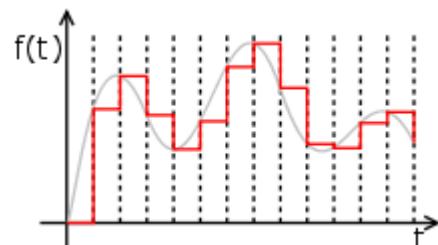


Figura 14

### Conversor Analógico Digital A/D

Es un dispositivo electrónico capaz de convertir una señal analógica, ya sea de tensión o corriente, en una señal digital mediante un cuantificador y codificándose en muchos casos en un código binario en particular. Donde un código es la representación unívoca de los elementos, en este caso, cada valor

numérico binario hace corresponder a un solo valor de tensión o corriente.

En la cuantificación de la señal se produce pérdida de la información que no puede ser recuperada en el proceso inverso, es decir, en la conversión de señal digital a analógica y esto es debido a que se truncan los valores entre 2 niveles de cuantificación, mientras mayor cantidad de bits mayor resolución y por lo tanto menor perdida de información.

La *resolución* es el nivel de voltaje que es capaz de discriminar un conversor A/D. O sea el nivel de voltaje para el cual el conversor cambia en un bit menos significativo (LSB). La resolución ( $R$ ) depende del voltaje a plena escala y del número de bits del conversor.

### Conversor Digital Analógico D/A

Es un dispositivo que convierte un código digital en una señal eléctrica correspondiente (voltaje o corriente). Su función dentro de un DAQ es de control: proporcionar un nivel de voltaje o corriente deseada a un elemento que permitirá corregir la variable medida hasta el valor deseado. Este tipo de dispositivo también se puede utilizar como generador de señales.

### Microprocesadores

Los microprocesadores se encargan del almacenamiento y procesamiento de los datos, son dispositivos que se encargan de todas las funciones de procesamiento de la señal. Es el circuito integrado central más complejo de un DAQ.

Es el encargado de ejecutar instrucciones programas en lenguaje de bajo nivel, realizando operaciones aritméticas y lógicas simples, tales como sumar, restar, multiplicar, dividir, lógicas binarias y accesos a memoria.

Puede contener una o más unidades centrales de procesamiento (CPU) constituidas, esencialmente, por registros, una unidad de control, una unidad aritmético lógica (ALU) y una unidad de cálculo en coma flotante.

## **Protocolos de Comunicaciones**

Lo fundamental de la comunicación de datos es resolver el problema de llevar la información de un punto A hacia un punto B sin errores, utilizando redes con la codificación correspondiente para su trasmisión. Para esto utilizamos canales de comunicación que establecen la unión entre los puntos A y B. En dichos puntos estarán los equipos transmisores y receptores de datos y sus convertidores encargados de la codificación y decodificación. Los sistemas de comunicación no responden ni reaccionan ante el contenido de la información. Un componente importante en el sistema de comunicación es el protocolo de comunicación.

Un *protocolo de comunicaciones* es un sistema de reglas que permiten que dos o más entidades de un sistema de comunicación se comuniquen entre ellas para transmitir información por medio de cualquier tipo de variación de una magnitud física. Se trata de las reglas o el estándar que define la sintaxis, semántica y sincronización de la comunicación, así como también los posibles métodos de recuperación de errores. Los protocolos pueden ser implementados por hardware, por software, o por una combinación de ambos.

También se define como un conjunto de normas que permite la comunicación entre ordenadores, estableciendo la forma de identificación de estos en la red, la forma de transmisión de los datos y la forma en que la información debe procesarse.

Los sistemas de comunicación utilizan formatos bien definidos (protocolo) para intercambiar mensajes. Cada mensaje tiene un significado exacto destinado a obtener una respuesta de un rango de posibles respuestas predeterminadas para esa situación en particular. Normalmente, el comportamiento especificado es independiente de cómo se va a implementar. Los protocolos de comunicación tienen que estar acordados por las partes involucradas. Para llegar a dicho acuerdo, un protocolo puede ser desarrollado dentro de estándar técnico. Un lenguaje de programación describe el mismo para los cálculos, por lo que existe una estrecha analogía entre los protocolos y los lenguajes de programación: “los protocolos son a las comunicaciones como los lenguajes de

programación son a los cómputos".

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

La comunicación *I<sup>2</sup>C* (Inter-Integrated Circuit)[3]es un protocolo de comunicación serial desarrollado por Phillips Semiconductors allá por la década de los 80'. Básicamente se creó para poder comunicar varios circuitos integrados al mismo tiempo dentro de los televisores.

El protocolo *I<sup>2</sup>C* toma e integra lo mejor de los protocolos SPI y UART. Con el protocolo *I<sup>2</sup>C* podemos tener a varios maestros controlando uno o múltiples esclavos. Esto puede ser de gran ayuda cuando se van a utilizar varios microcontroladores para almacenar un registro de datos hacia una sola memoria o cuando se va a mostrar información en una sola pantalla.

El protocolo *I<sup>2</sup>C* utiliza sólo dos vías o cables de comunicación, así como también lo hace el protocolo UART.

Este bus se basa en tres señales:

- *SDA (System Data)* por la cual viajan los datos entre los dispositivos.
- *SCL (System Clock)* por la cual transitan los pulsos de reloj que sincronizan el sistema.
- *GND (Masa)*, interconectada entre todos los dispositivos "enganchados" al bus.

*I<sup>2</sup>C* es un protocolo de comunicación serial. Como podemos observar, el protocolo *I<sup>2</sup>C* envía información a través de una sola vía de comunicación. La información es enviada bit por bit de forma coordinada. Además, *I<sup>2</sup>C* es un protocolo síncrono. Al igual el protocolo SPI, el protocolo *I<sup>2</sup>C* trabaja de forma síncrona. Esto quiere decir que el envío de bits por la vía de comunicación SDA está sincronizado por una señal de reloj que comparten tanto el maestro como el esclavo a través de la vía SCL.

En la siguiente tabla podemos observar la ficha técnica del protocolo I<sup>2</sup>C:

<b>Número de vías o cables</b>	2
<b>Velocidad máxima</b>	Modo estándar (Sm) = 100kbps
	Modo rápido (Fm) = 400kbps
	Modo High Speed (Fm+) = 3.4Mbps
	Modo Ultra Fast (Hs-mode) = 5Mbps
<b>Síncrono o Asíncrono</b>	Síncrono
<b>Paralelo o Serial</b>	Serial
<b>Número máximo de Maestros</b>	Ilimitado
<b>Número máximo de Esclavos</b>	1008

Con el protocolo I<sup>2</sup>C la información viaja en mensajes. Los mensajes van divididos en *tramas* de datos. Cada mensaje lleva una trama con una dirección la cual transporta la dirección binaria del esclavo al que va dirigido el mensaje, y una o más tramas que llevan la información del mensaje. También el mensaje contiene condiciones de inicio y paro, lectura y escritura de bits, y los bits ACK y NACK. Todo esto va entre cada sección de datos.

Para que pueda quedar en la figura 15 se ilustra un mensaje enviado a través del protocolo I<sup>2</sup>C:

## Protocolo I<sup>2</sup>C:



Figura 15

A continuación se detalla el significado de cada sección del mensaje:

- Condición de Inicio – Start: La vía SDA cambia de un nivel de voltaje Alto a un nivel de voltaje Bajo, antes de que el canal SCL cambie de Alto a nivel Bajo.
- Condición de Paro – Stop: La vía SDA ahora cambia de un nivel de voltaje Bajo a Alto, después de que la vía SCL cambia de Bajo a Alto.
- Trama de Dirección – Address Frame: Es una secuencia única que va de los 7 a los 10 bits. Esta sección (Frame) se envía a cada Esclavo, y va a identificar al Esclavo con el que el Maestro se quiere comunicar.
- Bit para Lectura/Escritura A – Read/Write Bit A: Es un bit de información enviado a los Esclavos. Por medio de este bit el Maestro indica si le va enviar información al Esclavo (Nivel Bajo de voltaje = Escritura), o si el Maestro quiere solicitarle información al Esclavo (Nivel Alto de Voltaje = Lectura).
- Bit ACK/NACK: Despues de cada sección (Frame) de información enviada en un mensaje, podemos notar que lleva un bit acknowledge/no-acknowledge (reconocido/no-reconocido). Esto ayuda a identificar si la información fue enviada correctamente. En seguida de que se envía un Frame, si este fue recibido con éxito, se retorna un bit ACK al remitente. Si la información no fue recibida con éxito, se retorna un bit NACK.

El protocolo I<sup>2</sup>C no cuenta con una línea de selección de esclavos (Como lo hace el protocolo SPI), así que se debe establecer una forma de notificación al Esclavo, para que éste se prepare para recibir información del Maestro. Mediante la sección de Dirección (Address) en el mensaje es como se notifica al Esclavo, es por eso que va en seguida del Bit de Inicio (Start).

Así que esta es la secuencia que se sigue para el direccionamiento de esclavos.

1. El Maestro envía la dirección del Esclavo con el que quiere comunicarse, esta dirección se envía a todos los Esclavos que estén conectados.
2. Cada Esclavo recibe la dirección, y la compara con su propia dirección.
3. Si la dirección coincide con el Esclavo, en seguida el Esclavo envía un bit ACK con nivel de voltaje Bajo, de regreso al Maestro.
4. Si la dirección no coincide con el Esclavo, simplemente no se hace nada, y la vía SDA permanecerá en nivel de voltaje Alto.

Después de la dirección en el mensaje, se envía un bit, notificando al Esclavo si el Maestro quiere escribir información o leer información de él. Si el Maestro quiere enviarle información al Esclavo (Escribir), entonces manda un Bajo. Si el Maestro quiere solicitar información al Esclavo (Leer), el bit Read/Write será de nivel Alto.

Después de que el Maestro recibe el bit ACK del Esclavo, la primera sección o Frame de información está lista para ser enviada al Esclavo.

El Frame para transportar información siempre es de 8 bits y se envía primero el bit más significativo. Después de enviar un Frame de información, inmediatamente se envía un bit ACK/NACK, para comprobar que el Frame de información efectivamente se ha llegado a su destino y se ha recibido satisfactoriamente. Independientemente de quién envía el Frame de información, si el Esclavo o el Maestro, siempre se debe enviar en seguida el bit ACK/NACK, antes de que se envíe el segundo Frame de información.

Una vez que se envían ambos Frames de información, el Maestro puede activar una condición de paro (Stop), y con esto se va a detener la transmisión. La condición de paro (Stop) es un cambio de voltaje de nivel Bajo a nivel Alto, en la vía SDA, y se activa cuando en la vía SCL se pasa de un nivel Bajo a un nivel

Alto (La vía SCL, posteriormente, permanecerá en nivel Alto).

Para mayor detalle, se describen los pasos del protocolo en la ejecución:

1.- El Maestro envía la condición de Inicio (Start) a cada Esclavo que esté conectado en la vía SDA, cambia el nivel de voltaje a Bajo, y deja la vía SCL en estado Alto (figura 16).

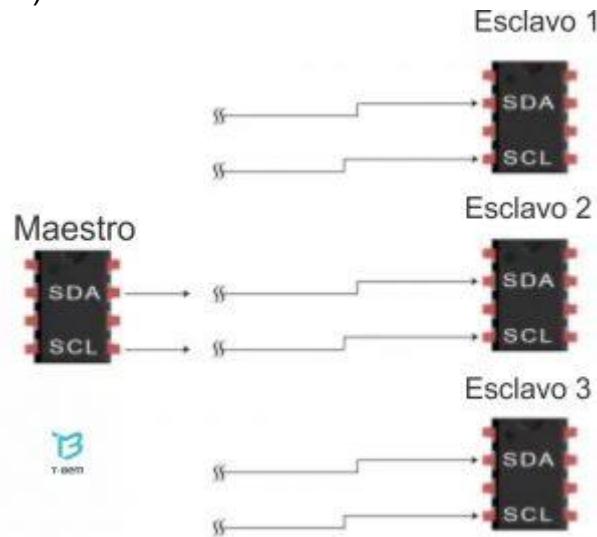


Figura 16

2.- El Maestro envía la dirección de 7 a 10 bits, a cada uno de los Esclavos para identificar al Esclavo con el que se quiere comunicar (figura 17).

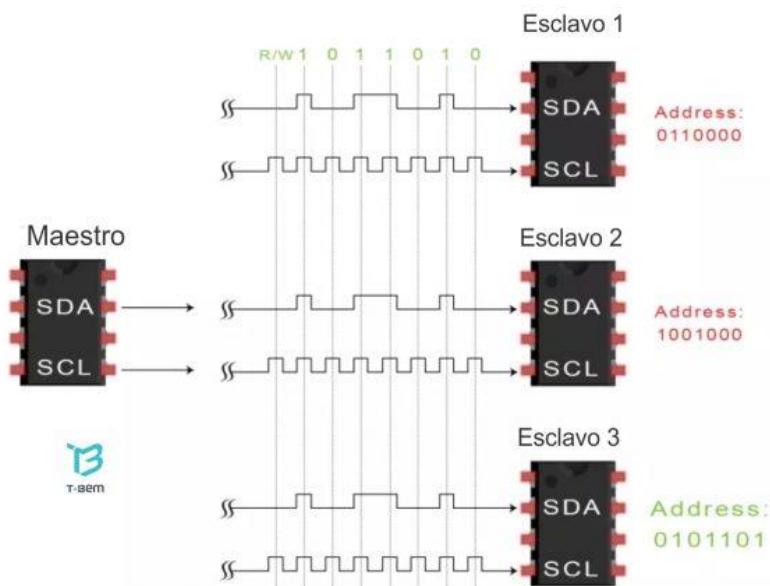


Figura 17

3.- Cada Esclavo recibe la dirección y la compara con su propia dirección. Si la dirección coincide, el Esclavo envía un bit ACK, y pone la vía SDA en nivel Bajo de voltaje. Si la dirección no es la misma, entonces los Esclavos no hacen nada y dejan la vía SDA en el mismo nivel de voltaje Alto (figura 18).

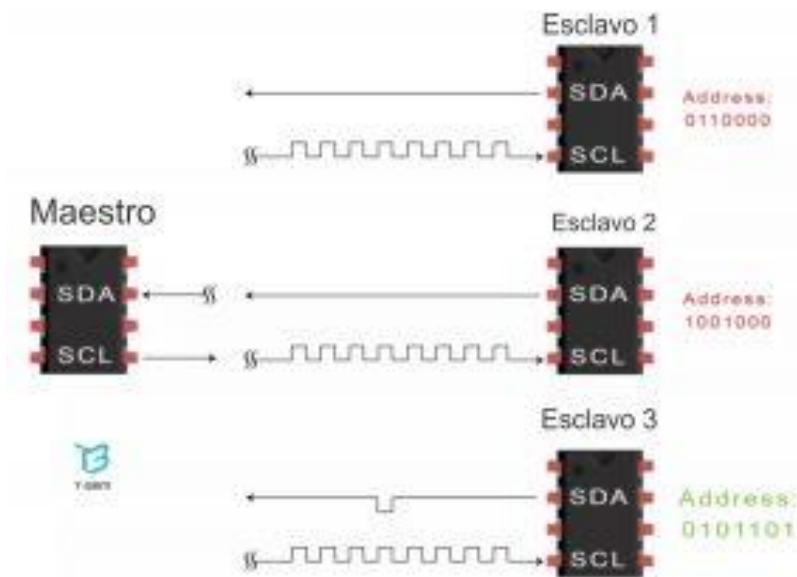


Figura 18

4.- El Maestro envía o recibe los Frames de información (figura 19).

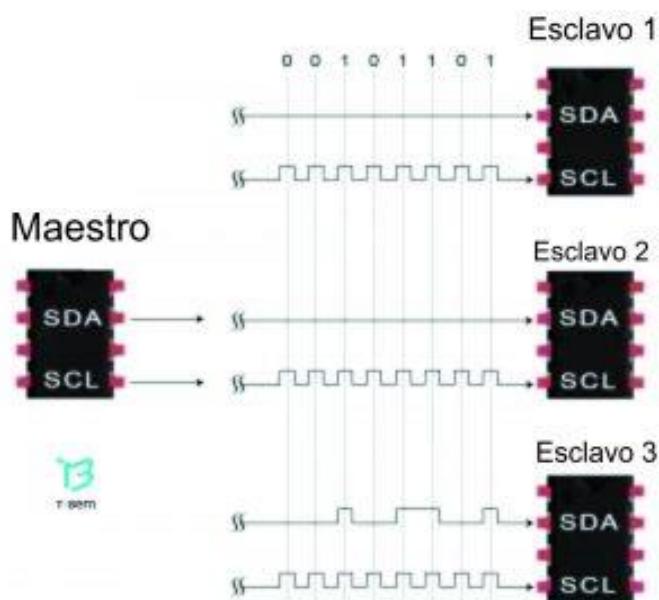


Figura 19

5.- Después de que cada Frame de información fue enviado, el dispositivo que recibe (ya sea Esclavo o Maestro) va a enviar un bit ACK al remitente, para notificarle que la información se recibió exitosamente (figura 20).

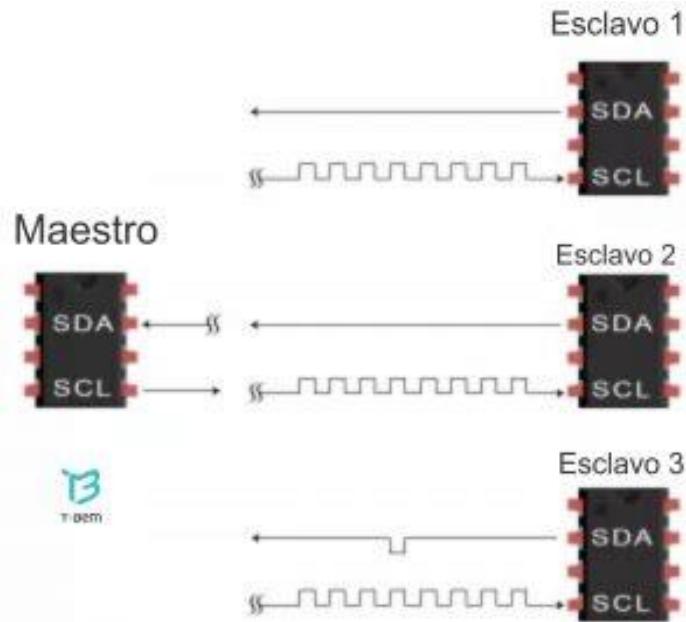


Figura 20

6.- Para concluir la transmisión de información, el Maestro envía al Esclavo la condición de paro (Stop) con un nivel Alto en la vía SDA, cuando cambia el estado de SCL a Alto (figura 21).

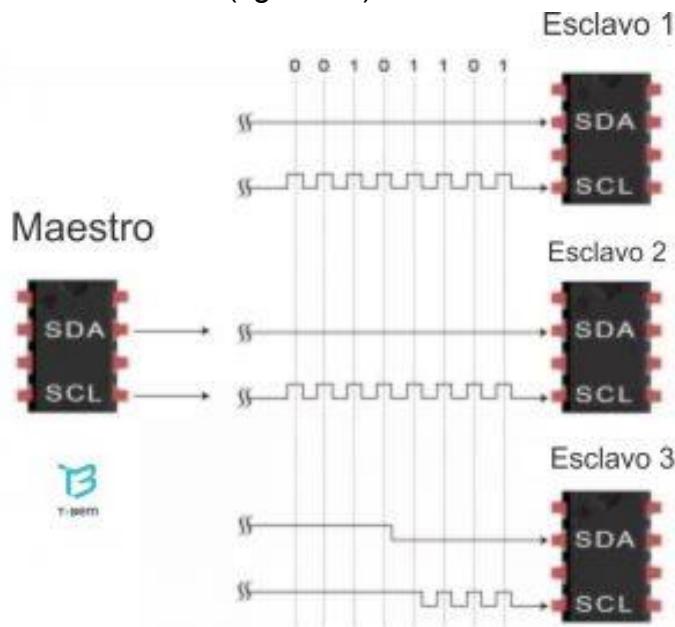


Figura 21

## Protocolo SPI

El bus *SPI* (Serial Peripheral Interface) [4]es un protocolo de comunicación serial que nace casi a principios de 1980 cuando Motorola™ lo comienza a introducir y desarrollar en el primer microcontrolador derivado de la misma arquitectura del microcontrolador 68000. SPI se ha convertido en uno de los más populares protocolos para trabajar con comunicación serial debido a su velocidad de transmisión, simplicidad, funcionamiento y también gracias a que muchos dispositivos en el mercado como pantallas LCD, sensores, microcontroladores pueden trabajar con él.

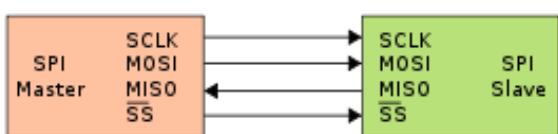


Figura 22

El SPI es un protocolo síncrono que trabaja en modo full duplex para recibir y transmitir información, permitiendo que dos dispositivos pueden comunicarse

entre sí al mismo tiempo utilizando canales diferentes o líneas diferentes en el mismo cable. Al ser un protocolo síncrono el sistema cuenta con una línea adicional a la de datos encarga de llevar el proceso de sincronismo (figura 22).

Dentro de este protocolo se define un maestro que será aquel dispositivo encargado de transmitir información a sus esclavos. Los esclavos serán aquellos dispositivos que se encarguen de recibir y enviar información al maestro. El maestro también puede recibir información de sus esclavos, cabe destacar. Para que este proceso se haga realidad es necesario la existencia de dos registros de desplazamiento, uno para el maestro y uno para el esclavo respectivamente. Los registros de desplazamiento se encargan de almacenar los bits de manera paralela para realizar una conversión paralela a serial para la transmisión de información.

Existen cuatro líneas lógicas encargadas de realizar todo el proceso:

- *MOSI* (Master Out Slave In): Línea utilizada para llevar los bits que provienen del maestro hacia el esclavo.
- *MISO* (Master In Slave Out): Línea utilizada para llevar los bits que

provienen del esclavo hacia el maestro.

- **CLK (Clock):** Línea proveniente del maestro encarga de enviar la señal de reloj para sincronizar los dispositivos.
- **SS (Slave Select):** Línea encargada de seleccionar y a su vez, habilitar un esclavo.

En la figura 23 se presenta una imagen donde se tienen todas estas líneas con sus respectivos registros de desplazamiento y su dirección de flujo:

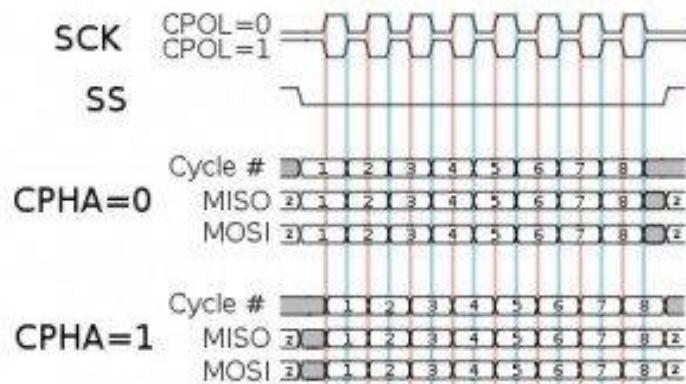


Figura 23

Existen cuatro modos en el cual se puede enviar información dependiendo de dos parámetros basados en la señal de reloj. El primer de ellos es la polaridad y el segundo es la fase. Al tener dos parámetros donde cada uno puede tomar dos estados se tendrá entonces cuatro modos distintos de poder llevar a cabo el proceso de transmisión y envío de información.

- Modo 0: CPOL = 0 y CPHA = 0. Modo en el cual el estado del reloj permanece en estado lógico bajo y la información se envía en cada transición de bajo a alto, es decir alto activo.
- Modo 1: CPOL = 0 y CPHA = 1. Modo en el cual el estado del reloj permanece en estado lógico bajo y la información se envía en cada transición de alto a bajo, es decir bajo activo.
- Modo 2: CPOL = 1 y CPHA = 0. Modo en el cual el estado del reloj permanece en estado lógico alto y la información se envía en cada transición de alto a bajo, es decir bajo activo.

transición de bajo a alto, es decir alto activo.

- Modo 3: CPOL = 1 y CPHA = 1. Modo en el cual el estado del reloj permanece en estado lógico alto y la información se envía en cada transición de alto a bajo, es decir bajo activo.

Esta puede brindarle una idea más intuitiva de los modos explicados anteriormente.

La configuración de modos es independiente para cada esclavo con esto quiere decir que cada esclavo puede tener una configuración de CPOL y CPHA distinta a la de otros esclavos, inclusive una frecuencia de trabajo distinta y entonces para esto, el maestro deberá adaptarse a la configuración de cada esclavo. Por esta razón es recomendable que el sistema trate de trabajar con los mismo parámetros de ser posible porque si no, resultaría poco práctico.

En este protocolo se define únicamente un maestro y varios esclavos. La manera en la cual estos dispositivos se conectan pueden ser de dos tipos: encadenado o paralelo. El de tipo encadenado las entradas del MOSI de cada esclavo va conectada con el MOSI del maestro para el primer caso o de su esclavo anterior para el resto. Además, se utiliza un único modo de selección de esclavo proveniente del maestro en forma paralela hacia cada esclavo.

Por otro lado, en el tipo paralelo se utiliza un único MOSI proveniente del maestro en forma paralela hacia cada esclavo. Además, se adiciona una línea de selección de esclavo proveniente del maestro por cada esclavo que exista en el sistema.

La transmisión de información puede darse de muchas maneras dependiendo del fabricante, en muchos casos la línea SS habilita un esclavo cuando ésta se pone en estado lógico cero pero eso puede cambiar. La transmisión de bits se puede dar comenzando con el LSB o con MSB dependiendo también del fabricante, es muchos casos se comienza por el bit más significativo. Un bit es transmitido cada ciclo de reloj.

Existe una serie de ventajas que ofrece este protocolo, entre ellas está la velocidad de transmisión ya que es configurable a través de software y

dependerá también de los dispositivos utilizados en el sistema. Con respecto a otros protocolos seriales que trabajan a modo half duplex, el SPI tiene velocidades de transmisión mayores, debido a que éste trabaja en modo full duplex. Otros parámetros configurables a través de software son la frecuencia del reloj, la configuración de fase (CPHA) y polaridad (CPOL). Si solo existe un esclavo, puede colocarse la línea SS fija si el esclavo lo permite. No se limitan a trabajar con palabras de ocho bits. Es ampliamente utilizado cuando se necesita comunicar con equipos a distancias cortas.

### ***La Marcha Humana***

La marcha humana [5] es un modo de locomoción bípeda con actividad alternada de los miembros inferiores, que se caracteriza por una sucesión de doble apoyo y de apoyo unipodal, es decir que durante la marcha el apoyo no deja nunca el suelo, mientras que en la carrera, como en el salto, existen fases aéreas, en las que el cuerpo queda suspendido durante un instante. También se puede definir como un desequilibrio permanente hacia delante.

Desde una óptica dinámica, la marcha es una sucesión de impulsos y frenados, en los que el motor o el impulso se sitúan a nivel del miembro inferior posterior y el frenado en el anterior.

Más que el desarrollo de un reflejo innato, la marcha es una actividad aprendida. Durante los primeros años de su infancia el niño experimenta con su sistema neuromuscular y esquelético, hasta llegar a integrar esta actividad a nivel involuntario. Hasta los 7 u 8 años no se alcanza la marcha característica que una persona muestra en la edad adulta. Aunque algunas variables dependientes del crecimiento, como la longitud del paso, continúan evolucionando hasta alcanzar los valores típicos del adulto alrededor de los 15 años. Pese al carácter individual de este proceso, las semejanzas entre sujetos distintos son tales que puede hablarse de un patrón característico de marcha humana normal, patrón que varía con diferentes circunstancias como el tipo de terreno, la velocidad, la pendiente,... y sobre todo bajo determinadas condiciones patológicas. Nos centraremos en este tema en la marcha humana normal, sobre suelo llano, en línea recta y a velocidad espontáneamente adoptada.

## El Ciclo de Marcha

Para su mejor descripción conviene dividir la marcha en fases, ya que su análisis cinemático comienza por la inspección visual de cada región anatómica, en cada una de las fases del ciclo de la marcha, mientras el individuo camina. Por ello, definiremos el ciclo de marcha y sus fases.

El ciclo de marcha es la secuencia de acontecimientos que tienen lugar desde el contacto de un talón con el suelo, hasta el siguiente contacto del mismo talón con el suelo. Durante un ciclo de marcha completo, cada miembro inferior considerado pasa por dos fases:

- A) *Fase de apoyo*: en la cual el pie de referencia está en contacto con el suelo.
- B) *Fase de oscilación*: en la que el pie de referencia está suspendido en el aire.

La fase de apoyo constituye alrededor del 60% del ciclo y la fase de oscilación representa el 40% restante. Las fases del ciclo de marcha, para facilitar su estudio suelen dividirse, todavía, en componentes más pequeños o subfases, según la siguiente secuencia: El ciclo se inicia con el impacto de talón en el suelo; al 15% el antepié también contacta con el suelo, por lo que esta subfase se denomina "pie plano sobre el suelo" o media; al 40% del ciclo, el talón comienza a elevarse del suelo (subfase de despegue de talón o final), al 50%, despegá el antepié, que culmina al 60% del ciclo con el despegue de los dedos, lo que indica también el comienzo de la fase de oscilación. La atribución de percentiles en esta fase es algo imprecisa, pero en la primera parte, se realiza el avance del miembro oscilante hasta alcanzar el miembro contralateral, y la extensión de rodilla completa el avance del miembro inferior. Al cumplirse el 100% del ciclo, se produce de nuevo el impacto de talón, con el mismo pie. Autores como Perry dividen la fase de apoyo en 4 subfases (inicial, media, final y preoscilación) y la de oscilación en 3 (inicial, media y final). El ciclo de marcha con sus porcentajes de duración sucede exactamente igual para el miembro contralateral, lo que revela, considerando los dos miembros inferiores, la existencia de dos períodos de apoyo bipodal o doble apoyo, que se caracterizan

porque los dos pies contactan con el suelo: uno está iniciando el contacto de talón mientras que el otro, próximo a la fase de despegue, se apoya por la cabeza del primer metatarsiano y el pulpejo del *hallux* (dedo gordo). Estos periodos tienen un porcentaje de duración de alrededor de un 10%, cada uno, y, también hay durante un ciclo de marcha dos periodos de apoyo monopodal durante los cuales tan sólo un miembro inferior contacta con el suelo y sobre él recae el peso del cuerpo.

Los cuatro periodos en que se divide el ciclo de marcha son, por tanto:

1. *Primer periodo de doble apoyo*: Que comienza cuando el pie tomado como referencia toma contacto con el suelo por el talón, frenando la aceleración del cuerpo hacia delante y culmina con el despegue del miembro contralateral.
2. *Primer apoyo unipodal o periodo portante*: En el cual el peso del cuerpo recae en la extremidad tomada como referencia, mientras el miembro contralateral está oscilando.
3. *Segundo doble apoyo*: El pie considerado se apoya solo por el antepié en el suelo y está en situación posterior acelerando el cuerpo hacia delante, es el miembro propulsor o miembro activo dinámico.
4. *Segundo apoyo unipodal o periodo oscilante*: El pie que en el tiempo anterior solo se apoyaba por el antepié en el suelo, ha despegado e inicia su periodo oscilante.

Para una mayor descripción del ciclo de marcha pueden realizarse medidas de algunos parámetros generales descriptivos como la longitud, anchura y ángulo del paso, la cadencia y la velocidad de marcha.

Cada ciclo de marcha comprende dos pasos, siendo el paso la actividad entre el apoyo de un talón y el apoyo sucesivo del talón contralateral.

La *longitud del paso* corresponde a la distancia que separa el apoyo inicial de un pie del apoyo inicial del pie contralateral. Su media es de 75 cm.

La *anchura del paso* es la distancia entre los puntos medios de ambos talones y su media es de unos 10 cm. en terreno llano.

El *ángulo del paso* es el que forma el eje longitudinal del pie con la línea de dirección de la progresión; normalmente mide 15°.

La *cadencia* es el número de pasos ejecutados en la unidad de tiempo. Generalmente se mide en pasos por minuto. La cadencia espontánea o libre en adultos oscila de 100 a 120 ppm.

La *velocidad de marcha* es la distancia recorrida en la unidad de tiempo y también se obtiene evidentemente multiplicando la longitud del paso por su cadencia. Se expresa en m/min o Km/hora. La velocidad espontánea en adultos oscila de 75 a 80 m/min., es decir, de 4,5 a 4,8 Km/h.

### Mecanismos de Optimización de Energía

La marcha ocasiona un gasto energético y cada persona tiende a adoptar el tipo de marcha más eficiente para su estructura particular, con el menor gasto energético posible. Mediante la medición del consumo de oxígeno se puede determinar indirectamente el gasto energético.

Si nos desplazáramos sobre ruedas, el centro de gravedad seguiría una trayectoria rectilínea, se produciría un deslizamiento continuo y nuestra locomoción requeriría muy poca energía. Sin embargo, nuestro aparato locomotor imprime al centro de gravedad del cuerpo un movimiento que no es rectilíneo sino que describe unos desplazamientos, verticales y horizontales, que conducen a un mayor gasto metabólico; no obstante, el cuerpo humano ha desarrollado diversos mecanismos que mejoran el rendimiento de la marcha, a través de transferencias de energía y de la reducción del desplazamiento del centro de gravedad.

## Cinética de la Marcha

La *Cinética de la Marcha* estudia las fuerzas que intervienen desarrollando el movimiento de la misma.

Como hemos visto, en la provisión de la energía necesaria para la marcha va a ser muy importante la acción de la gravedad, colaborando en la transferencia entre energía potencial y cinética. El cuerpo humano durante la marcha utiliza al máximo la fuerza de gravedad y de reacción, la inercia y la mínima fuerza del músculo. Estas son, junto con la fricción o rozamiento, las principales fuerzas que influyen en la marcha.

*La fuerza de gravedad:* Como hemos señalado, la marcha se caracteriza por la traslación del centro de gravedad del cuerpo hacia adelante, llegando momentáneamente más allá del borde anterior de la base de sustentación, lo que origina una pérdida transitoria del equilibrio y la acción de la gravedad tiende a hacer caer el cuerpo hacia adelante y abajo, incrementando la velocidad y transformando la energía potencial en cinética, en este punto el pie que oscilaba se sitúa en el suelo, recuperando el equilibrio, al ofrecer una base de sustentación mucho más amplia y evitando, así, la caída del cuerpo.

*La fuerza de reacción* que ejerce el suelo sobre el individuo, a través de los pies y que es de igual magnitud que el impulso hacia abajo del pie durante la marcha, pero en sentido contrario. En el momento del choque de talón producimos una fuerza de frenado, mientras que en el momento del despegue se produce una fuerza de empuje hacia delante. En el impulso y en el frenado la dirección de la fuerza de reacción es diagonal y puede resolverse en dos componentes. El componente vertical sirve para contrarrestar la tensión hacia abajo de la fuerza de gravedad, mientras que el componente horizontal sirve en el choque de talón, para frenar el movimiento hacia adelante y en el despegue de antepié para generar la propulsión. Si a la superficie de apoyo le falta solidez como en el caso del barro, nieve húmeda y arena, aquella ofrece una resistencia muy pequeña de contrapresión, que trae como consecuencia un hundimiento o deslizamiento, que aumenta el gasto energético y disminuye la eficacia de la marcha.

La marcha requiere además una fricción adecuada entre el pie y el piso para no resbalar. La fuerza de fricción o de rozamiento dependerá del tipo de materiales en contacto y de las fuerzas que ejercen presiones entre ellos. Para que la marcha sea eficaz, la fricción debe ser suficiente para equilibrar el componente horizontal de las fuerzas de impulso y de frenado. Si es insuficiente, el pie se deslizará. Este hecho se comprueba al observar a un individuo cuando camina sobre hielo o en un suelo encerado, en los que la fricción es mínima y debe reducir la longitud del paso, para disminuir la componente horizontal y evitar así resbalar.

La *inercia*, entendida como la incapacidad del cuerpo o de sus segmentos para cambiar su estado de reposo o de movimiento sin la intervención de alguna fuerza, debe ser vencida en cada paso y cuanto mayor sea el peso del cuerpo mayor será la inercia que se ha de vencer. En cuanto al conocimiento de las diferentes fuerzas musculares que actúan en la marcha, su valoración es un problema extremadamente complejo y más aún cuando se consideran los aspectos fisiológicos y mecánicos. El momento de fuerza de un músculo depende de la longitud efectiva del brazo de palanca con el que actúa, de su sección transversal fisiológica, de la velocidad de contracción y de la longitud previa del mismo, siendo máxima cuando el músculo está elongado (aproximadamente al 120% de su longitud en reposo). La mayor demanda muscular de la marcha ocurre, precisamente, en el instante en que el músculo presenta una mayor longitud. Aunque la cuantificación absoluta de las fuerzas musculares que intervienen en la marcha sigue siendo un problema sin determinar, existen estudios exhaustivos, que explican la participación relativa de cada grupo muscular. Debemos recordar, que como sucede en otras destrezas motoras, también durante la marcha humana tienen lugar tres tipos de contracción muscular: Contracciones de tipo concéntrico, en las que ambas inserciones se aproximan produciendo movimiento en la dirección de la tracción muscular. Contracciones de tipo excéntrico, en las que las inserciones se separan frenando un movimiento que se produce en sentido contrario por la acción de otras fuerzas musculares o externas, como la gravedad o la inercia y que serán mayores que la fuerza generada por su propia tensión del músculo. Y también contracciones de tipo isométrico, en la que no hay variación de la

longitud del músculo, permaneciendo constante y que se producen para equilibrar fuerzas opuestas y mantener la estabilidad.

### Acciones Articulares en la Marcha

A continuación abordaremos la descripción de las acciones articulares en los periodos de marcha, definidos anteriormente. Tomando siempre como referencia la extremidad inferior derecha.

El *primer doble apoyo* se inicia cuando el pie tomado como referencia contacta con el talón en el suelo.

En el *plano sagital*, en este momento, el tobillo se halla en posición neutra de flexo/extensión, a continuación el tobillo se extiende por la caída del antepié controlada por los músculos del compartimento anterior de la pierna.

La *rodilla* alcanza al comienzo de este periodo su máxima extensión en la marcha, pero aún mantiene unos 5º de flexión, ya que, durante la marcha normal, la articulación de la rodilla nunca está en extensión total.

La *cadera* está en flexión de 30º y la fuerza de reacción del suelo, origina un momento flexor, debido a su alineación muy anterior, siendo contrarrestado por los músculos extensores de cadera (que se contraen para frenar esta flexión).

En el *primer apoyo unipodal o periodo portante*, el miembro inferior de referencia soporta el peso del cuerpo.

En el plano sagital la articulación del tobillo se flexiona de forma pasiva, por la inclinación hacia delante de la tibia. Como hemos señalado, la rodilla, en este periodo, debe estar ligeramente flexionada de 15 a 20º para evitar una ascensión brusca del centro de gravedad.

La cadera en este periodo realiza una extensión progresiva, pasando de una flexión inicial de unos 30º a una extensión de unos 10º al final del periodo portante.

En el *plano frontal*, durante el período portante las acciones musculares estabilizadoras son imprescindibles y a nivel de la cadera, en este periodo, hay

una caída de la pelvis de unos  $5^{\circ}$  hacia el lado oscilante siendo necesaria la contracción potente de los abductores, para evitar un mayor descenso.

En el *plano transversal* la pelvis se desplaza hacia adelante rotando sobre la cabeza femoral portante, con un giro de unos  $4^{\circ}$  alrededor del eje vertical, alcanzando su posición neutra al pasar un miembro frente a otro.

En el *segundo doble apoyo* el pie tomado como referencia está en situación posterior, próxima a la fase de despegue y las cabezas de los metatarsianos actúan como punto de apoyo para la rotación del miembro, en lo que se ha denominado rodillo de antepié.

En el plano sagital, este periodo se caracteriza por la extensión (de unos  $15^{\circ}$ ) de la tibiotarsiana por acción del tríceps sural y de los flexores de los dedos que se contraen con potencia elevando el talón del suelo. La rodilla y la cadera al principio se encuentran en extensión, siendo en este momento cuando la cadera alcanza su máxima extensión durante la marcha, de alrededor de unos  $10^{\circ}$ , pero al final del periodo comienza la actividad de los *flexores* impulsando el miembro hacia adelante y produciendo, de forma pasiva, una flexión de rodilla.

En el *segundo apoyo unilateral* el pie, tomado como referencia, ha despegado e inicia su periodo oscilante. El peso del cuerpo, por tanto, recae en la extremidad contralateral. Como se ha descrito, este periodo está dividido en dos fases, separadas por el momento del cruce de ambos miembros inferiores. En ambas fases, las principales acciones musculares que se desarrollan tienen lugar: en el plano sagital.

En la fase inicial de la oscilación se produce la flexión en masa de todo el miembro inferior. La rodilla aumenta su flexión alcanzando unos  $65^{\circ}$  en la mitad de la fase de oscilación, que corresponde al máximo valor de la flexión en todo el ciclo de marcha, a cadencia alta no se necesita acción muscular alguna para flexionar la rodilla ya que sigue un movimiento pendular. También la cadera alcanza su máxima flexión en el ciclo, alrededor de  $35^{\circ}$ , hacia la mitad del periodo oscilante y se debe principalmente al psoas ilíaco ayudado por los aductores y el sartorio.

En la fase final de la oscilación, los objetivos son desacelerar la pierna y posicionar correctamente el pie para establecer contacto con el suelo. Es necesaria una posición neutra del tobillo mantenida por los flexores. La rodilla debe pasar de una flexión necesaria para la oscilación a una postura de extensión al final de esta fase, que se va a completar por la acción de los vastos y el crural, mientras la contracción antagonista de los isquiotibioperoneos impide una extensión de rodilla demasiado violenta, al tiempo que desaceleran la flexión de cadera.

### Importancia de la Rodilla

La rodilla es la articulación más importante de las extremidades inferiores. La misma experimenta numerosos traumatismos, por lo que, los médicos especialistas necesitan tener un conocimiento muy profundo de la mencionada articulación ya que es la articulación más expuesta y menos protegida contra las lesiones mecánicas. Además, se distingue como la articulación más grande del ser humano que une tres huesos: fémur (en el extremo inferior), tibia (en el extremo superior) y la rótula, constituyéndose la articulación de mayor importancia para la marcha humana.

La *gonartrosis*, por ejemplo, es la enfermedad más común de una gran variedad de lesiones de la rodilla, siendo el ángulo de rotación la principal característica a estudiar. La misma provoca dolor durante el caminar o movimiento y, además, causa limitación progresiva de la movilidad de la articulación.

Después de realizar cirugías y verificar la reparación del problema, el proceso de rehabilitación depende de la intensidad de dolor del paciente lo que genera una calificación. En otras palabras, la evaluación de pacientes con gonartrosis depende de una valoración que relaciona el dolor y capacidad funcional. Esta medición es realizada entre el médico y el paciente.

Por lo expuesto, se hace necesario obtener una solución que permita medir y expresar el ángulo de la rodilla, en el plano sagital, para que el especialista pueda determinar el grado de afección haciendo uso de una interfaz computacional.

Del análisis de la marcha humana descrito en secciones anteriores, para un determinado ciclo de marcha, la curva cinemática correspondiente al ángulo de la rodilla es la que se presenta en la figura 24.

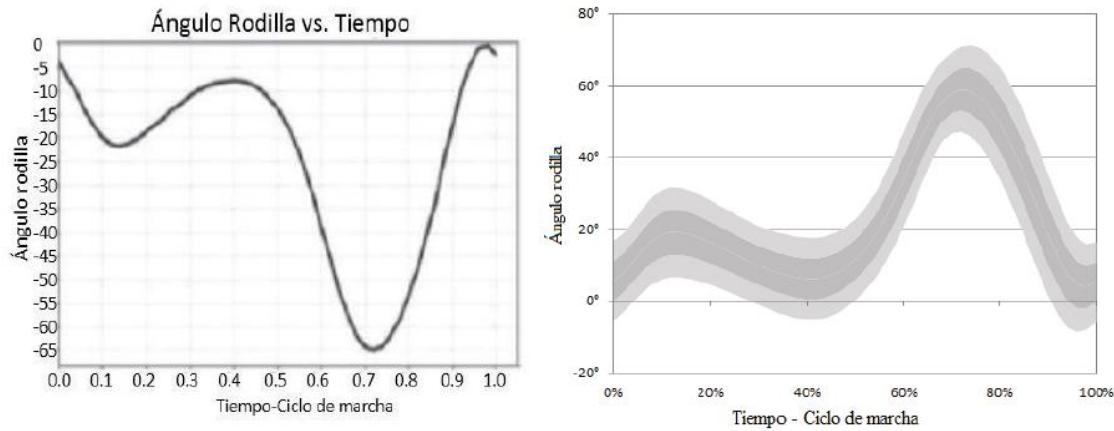


Figura 24

## Materiales y Métodos

### Materiales Empleados

Para mayor versatilidad, los fabricantes colocan los sensores dentro de placas de adaptación (*breakout board*), que contienen varios circuitos de apoyo. La placa de adaptación en el caso del MPU-6050 [6] es la GY-521, y es la que se utilizó en el proyecto.



Figura 25

El modulo GY-521 (figura 25) ofrece salida digital mediante el protocolo I<sup>2</sup>C, conversores analógico/digital de 16 bits de resolución por cada canal tanto del acelerómetro como del giróscopo (capta los ejes x, y, z al mismo tiempo), y más de 100 registros [7]. Parte de estos registros almacenan las mediciones digitalizadas por los conversores en un instante determinado, y se

conservan para una comunicación I<sup>2</sup>C. Otra parte de los registros establecen la configuración de los sensores, como sensibilidad, frecuencia de corte del filtro pasa bajos interno, etc. Ambos, acelerómetro y giróscopo tienen 4 sensibilidades seleccionables: ±2g, ±4g, ±8g, ±16g y ±250dps, ±500dps, ±1000dps, ±2000dps. Estas son las características principales, en la hoja de datos se exponen con mayor detalle todas sus características.

El principio de funcionamiento del acelerómetro es capacitivo y mide la aceleración asociada con el fenómeno de peso que experimenta una masa de prueba dentro del marco de referencia del dispositivo. Este giróscopo es un giróscopo de estructura vibrante o CVG. Por si solos, no funcionan como indicadores de posición, por lo tanto es necesario establecer una interfaz con un microcontrolador para el procesamiento de los datos, y convertirlo en información útil.

Como microcontrolador se utilizó un ATmega328P [8]. Se eligió este microcontrolador por estar implementado en una placa Arduino Nano, lo que radicó en una gran ventaja a la hora de ensayar varios códigos de programa con

facilidad y rapidez. Además de la facilidad de quemar programas sobre el microcontrolador, Arduino tiene la ventaja de ser *open source*, y de usar un lenguaje de programación simplificado basado en lenguaje C, *Wiring*. La facilidad de aprendizaje y entrenamiento de este lenguaje, sumado a la enorme cantidad de información acerca del mismo por ser *open source*, fueron vitales para conseguir una programación sin limitaciones.

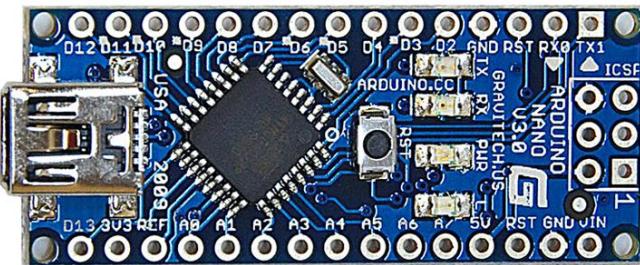


Figura 26

El Arduino Nano (figura 26) utiliza un regulador de tensión similar al 7805 para alimentar tanto al microcontrolador como a los circuitos de soporte, por lo tanto necesita un voltaje de

“dropout” mínimo de 2 volts aproximadamente para regular correctamente 5V. Por lo tanto cuando se energiza la placa por la entrada de este regulador, se necesitan como mínimo 7 volts. Para alcanzar este valor, se utiliza la fuente step-up DC-DC modelo CN6009 de alta eficiencia (figura 27-a).



Figura 27-a

Este módulo DC-DC está basado en el circuito integrado XL6009E1 el cual es un circuito conmutador de corriente de alta frecuencia (400 kHz), implementado en un convertidor DC-DC elevador (figura 27-b). Tiene una eficiencia de hasta 94%. El rango de trabajo de la tensión de entrada del módulo es de 3 volts a 32 volts, siendo recomendable de 5 volts a 32 volts. La salida opera de 5 volts a 32 volts.

Finalmente, como se trata de un dispositivo inalámbrico, la información recolectada durante los ensayos debe ser almacenada, aun en la ausencia de energía. Esta solución la ofrecen distintos tipos de memorias no volátiles. Se eligió utilizar una memoria *eeprom* AT24C04C [9] de Atmel™ para el prototipo de pruebas y, posteriormente, una memoria *flash* microSD para el dispositivo final.

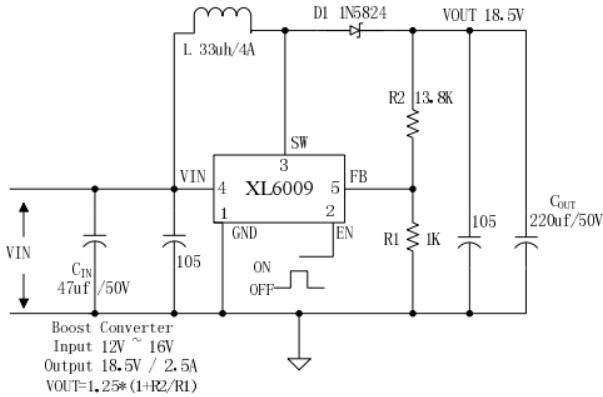


Figura 27-b

Hay que destacar que para mayor facilidad de operación de la memoria microSD se utilizó un lector genérico microSD (figura 28) que facilitó toda la interfaz de comunicación SPI, a diferencia de la memoria *eeprom* que era de interfaz I<sup>2</sup>C.

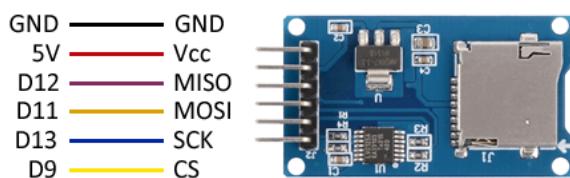


Figura 28

## ***Circuitos Implementados***

### **Prototipo**

Para el prototipo de pruebas, la placa Arduino Nano se implementó como una tarjeta de adquisición de datos. Por las características del ATmega328P se puede inducir que resultaría muy limitado a la hora de adquirir datos, pero esto no fue un problema, debido a la baja frecuencia involucrada en los ensayos, correspondientes a la baja frecuencia de la marcha humana bípeda.

Teniendo como parte central el Arduino Nano, se desarrolló un circuito que conectara vía I<sup>2</sup>C a una IMU MPU-6050 y, para conservar los datos, se utilizó aquí la memoria *eeprom* de Atmel™. Se eligió esta memoria por tener, como se dijo, comunicación I<sup>2</sup>C, desarrollando un único bus de comunicación digital que minimizó la cantidad de conexiónado y potenció la portabilidad, ya que los ensayos requerían un dispositivo inalámbrico.

Se destaca que en este caso todo el circuito se implementó en una misma placa de circuito impreso, sin partes móviles. Es decir, el sensor estaba solidario a todo el aparato, ya que lo único que se buscaba en primera instancia es la caracterización del sensor en sí.

Además, el procesamiento de los datos crudos del sensor fue ejecutado por el microcontrolador ATmega328P. Es decir, debido a los bajos requerimientos el microcontrolador tuvo la capacidad de actuar como DAQ y de a su vez procesar los datos, para luego almacenarlos.

En cuanto a su alimentación, el circuito se alimentó conectando cuatro baterías AAA en serie a la entrada del módulo DC-DC elevador, ajustándose el potenciómetro de dicho módulo para entregar una salida no regulada de aproximadamente 7 volts.

. Además, el circuito impreso correspondiente se desarrolló en una placa de 5 cm de alto por 7 cm de largo, las cuales son las mismas dimensiones de la caja contenedora que se utilizó. En la parte superior de la caja contenedora se presentó un panel con un interruptor de encendido/apagado y dos pulsadores, uno para la captura de datos, y otro para la descarga de datos. En el diagrama de la figura 29 se omite el conexionado de dicho panel.

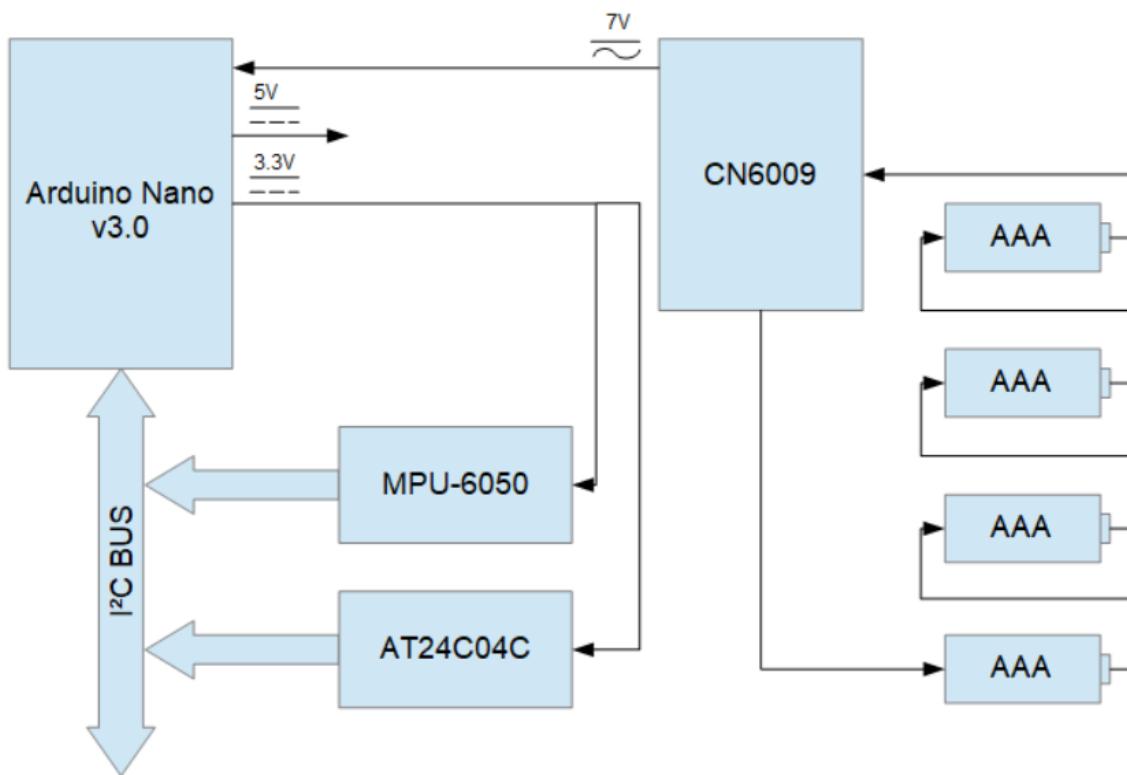


Figura 29

### Dispositivo Final

Con respecto al prototipo de pruebas, se hicieron varias modificaciones en el dispositivo final, ya que se para captar correctamente el ángulo de la rodilla, los requerimientos fueron mayores.

El compartimiento que contiene los circuitos implementados fue el mismo que se utilizó en el prototipo de pruebas, y la placa de circuito impreso fue de las mismas dimensiones.

En este caso, para captar el ángulo de la rodilla y tener mayor registro de la marcha humana, se utilizaron dos IMUs MPU-6050 por el motivo que se explicará más adelante. Estos sensores debían estar solidarios a sus extremidades inferiores correspondientes, uno con el muslo y otro con la pierna, mediante cintas velcro. Es decir, los sensores estaban fuera de la caja contenedora y conformaban la parte móvil del sistema. Su conexión se realizó mediante cables planos. La caja contenedora se acomodó en la cadera del paciente bajo estudio mediante un broche sujetador.

Otra modificación fue el reemplazo de la memoria eeprom por la memoria microSD debido a la necesidad de mayor almacenaje y facilitar además la portabilidad de la información para poder ser procesada en un ordenador.

Debido a que la escritura y la lectura de la memoria microSD era más compleja, el costo de procesamiento se vio afectado, limitando la capacidad de muestreo que se tenía anteriormente. Por lo tanto se procedió a utilizar a la placa Arduino Nano únicamente como placa de adquisición de datos, almacenando los datos crudos de los sensores para luego ser procesados correctamente en un ordenador utilizando la plataforma de MATLAB™.

La alimentación se realizó mediante baterías AA, de mayor capacidad que en el caso del prototipo, para otorgarle mayor autonomía al sistema.

## ***Errores inherentes***

En este apartado se describen los errores propios tanto del acelerómetro como del giróscopo dentro del módulo MPU-6050. Esta clase de imprecisiones dependen exclusivamente de la construcción del instrumento y del principio de funcionamiento que lo gobierna.

Se realizó un completo estudio previo de estos errores utilizando la información provista por los fabricantes, para posteriormente desarrollar un adecuado procesamiento de las señales.

El acelerómetro es altamente sensible al ruido eléctrico, y en menor medida al ruido mecánico. No está pensado para detectar vibraciones, pero puede detectar vibraciones no deseadas no tan pequeñas, y formaran parte del ruido intrínseco del mismo. Además, tiene un ancho de banda reducido (1kHz), funcionando mejor a frecuencias bajas. No tiene la capacidad de seguir eventos de mayor frecuencia (para ello se prefiere el giróscopo). A favor del acelerómetro, es muy preciso y cuenta con elevada exactitud.

El giróscopo, en cambio, cuenta con alta inmunidad al ruido, posee un mayor ancho de banda (8kHz) y sigue correctamente eventos de mayor frecuencia. No tiene tanta exactitud como el acelerómetro, y es propenso al error de deriva (*drift*).

Los valores de ancho de banda son teóricos y son dados por el fabricante en la hoja de datos. Los estudios de biomecánica involucran frecuencias relativamente bajas y a priori los eventos lentos pueden ser determinados con exactitud por el acelerómetro, y los eventos rápidos seguidos correctamente por el giróscopo.

## Técnicas Empleadas

### Acelerómetro y Giróscopo como Medidores de Inclinación

Como se mencionó anteriormente, tanto el acelerómetro como el giróscopo no acusan la posición del sistema por sí mismos. Para funcionar como sensores de posición necesitan un procesamiento de los datos de aceleración y velocidad que miden cada cierto tiempo [10].

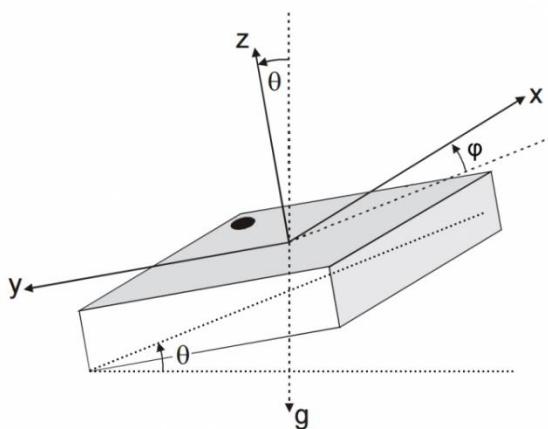


Figura 30

En este proyecto se utilizó como convención los ángulos de navegación (*yaw*, *pitch*, *roll*) para describir la orientación del sistema en tres dimensiones (figura 30). La rotación alrededor del eje z (yaw) queda desafectada de la sensibilidad del

acelerómetro, y solo puede ser detectada por el giróscopo, y por lo tanto las medidas de la misma con el MPU-6050 no son muy precisas y se descartó su procesamiento. Además, en la marcha humana la mayor cantidad de movimiento está concentrada en el plano sagital (pitch) y en el plano frontal (roll).

En el caso del acelerómetro, los ángulos de inclinación se obtienen mediante fórmulas trigonométricas que transforman las mediciones de aceleración de los ejes x, y, z en los ángulos yaw, pitch, roll. Existen diversas fórmulas trigonométricas que cumplen este propósito. Para el presente trabajo se utilizó la formula publicada en [11]:

$$\tan(\theta) = \frac{-a_x}{\sqrt{a_y^2 + a_z^2}}$$

$$\tan(\varphi) = \frac{a_y}{a_z}$$

Donde  $\theta$  es el ángulo pitch,  $\varphi$  el ángulo roll y  $a_x, a_y, a_z$  la aceleración medida en cada eje.

A diferencia del acelerómetro, el giróscopo mide velocidad angular y por lo tanto para obtener el ángulo barrido correspondiente se debe realizar una integración. Distintos métodos de integración numérica son aplicables, como la Regla de los Trapecios o la Regla de Simpson. Es importante que la integración tenga un bajo costo de procesamiento, ya que permite mayores tasas de muestreo, lo que se traduce en una mayor exactitud. Por esta razón, se recurrió a la Regla del Rectángulo:

$$\theta[nT_s] = \omega_y \cdot T_s + \theta[nT_s - 1]$$

$$\varphi[nT_s] = \omega_x \cdot T_s + \varphi[nT_s - 1]$$

Donde  $\omega_y$  es la velocidad angular medida en el eje  $y$ ,  $\omega_x$  la velocidad angular medida en el eje  $x$ , y  $T_s$  el tiempo de muestreo.

Estas fueron las técnicas empleadas para poder utilizar dichos sensores como indicadores de posición. En el caso del prototipo esto se implementó, como se dijo, mediante el Arduino Nano. En el dispositivo final, la plataforma de MATLAB™ se encargó de este trabajo.

### Adquisición de Datos

Una vez establecidas los métodos que permiten al acelerómetro y al giróscopo acusar ángulos de posición, es necesario programar la rutina de adquisición de datos en el microcontrolador.

Como se describió anteriormente, el dispositivo consistía de tres módulos: microcontrolador, memoria eeprom o microSD y los sensores correspondientes. Los tres módulos estaban conectados al bus I<sup>2</sup>C en el caso del prototipo. En el dispositivo final, el microcontrolador opera un bus I<sup>2</sup>C para la comunicación con los sensores y un bus SPI para la comunicación con la memoria SD. En ambos casos, la transferencia de datos fue arbitrada por el microcontrolador, que actuó de *maestro*, mientras que la memoria y los sensores actuaron como *esclavos*.

Como las frecuencias a estudiar eran relativamente bajas, la velocidad de transmisión de datos del bus no supuso un discriminante, y se omitieron los cálculos para alcanzar la máxima velocidad de los buses, tanto I<sup>2</sup>C como SPI. Básicamente la velocidad del bus aumenta cuando se aumenta la corriente del mismo, mediante resistencias pull-up más bajas. Como esto tenía asociado un mayor consumo, y no eran necesarias altas velocidades, se trabajó con la velocidad nominal.

Mediante programación y el protocolo I<sup>2</sup>C, se establecieron las configuraciones iniciales en los sensores, escribiendo en los registros del MPU-6050 reservados para ello. Se definió una sensibilidad de  $\pm 8g$  para el acelerómetro y de  $\pm 500\text{dps}$  para el giroscopio, ya que la fuerza aceleradora predominante era la gravedad, y no se necesitaba una alta sensibilidad.

La parte más sofisticada de adquisición de datos era realizada por el módulo MPU-6050. Aquí el fabricante ya implementó el filtro pasabajo correspondiente y la conversión analógica/digital necesaria, por lo que la señal cruda que se obtiene es de bastante calidad, aunque aún con ruido eléctrico.

Como se detalla en el Anexo B, mediante las rutinas correspondientes de I<sup>2</sup>C, se realizaron las llamadas correspondientes a los registros almacenadores de dichas señales crudas, y el microcontrolador se encargó de formar las variables y organizarlas para su almacenaje. Todo esto debe ocurrir en un lapso de tiempo constante y lo suficientemente corto como para cumplir con el Teorema de Muestreo, y por lo tanto se estableció  $T_s$  en 36 ms.

Este tiempo de 36 ms fue el mínimo que se pudo alcanzar teniendo en cuenta las condiciones más exigentes, es decir, las del dispositivo final, que involucraban la escritura física de una memoria flash, su comunicación SPI con el microcontrolador, y la comunicación I<sup>2</sup>C de dos sensores como esclavos.

En el prototipo esto no fue un problema, ya que la memoria eeprom solo necesitaba 3ms para realizar una escritura física.

## Calibración

Como toda unidad de medición, la IMU MPU-6050 actuando como indicador de posición también necesita calibración.

La etapa de calibración es indispensable para que cualquier medición sea coherente.

Una vez inicializados los sensores, se debe determinar correctamente la posición estática inicial, para que las inclinaciones que se midan tengan certeza. Como esta posición de inclinación nula puede variar de un ensayo a otro, se debe calibrar el dispositivo cada vez que se realiza una prueba.

La calibración tanto del acelerómetro como del giróscopo es el proceso de determinación del offset de cada eje.

El offset de cada eje es calculado mediante el promedio de 5000 muestras. En esta etapa la tasa de muestreo fue la máxima posible, siendo  $T_s$  del orden de los microsegundos. La obtención de dichas muestras se realizó mediante el algoritmo de adquisición de datos explicado anteriormente.

El algoritmo de calibración se ejecutó previamente a cualquier ensayo, es decir, previamente al registro de datos de los sensores. Determinado el offset en cada canal, se le aplicó la corrección de cero (offset) a cada muestra adquirida durante el ensayo y, finalmente, se procedió al procesamiento de los datos.

## **Procesamiento de Datos**

### Filtrado

Según se expuso anteriormente, el acelerómetro es susceptible al ruido mientras que el giróscopo mantiene una alta inmunidad frente a señales espurias. Cabe destacar que la IMU que implementa tanto al acelerómetro como al giróscopo ya cuenta con un prefiltrado de dichas señales.

Ambos, tanto el giróscopo como el acelerómetro cuentan con filtros limitadores de banda, de un 1 kHz para el acelerómetro y de 8 kHz para el giróscopo, colocados inmediatamente después de que se obtiene la señal de

tensión mediante convertidores capacitancia/tensión.

En el caso del acelerómetro, la IMU implementa además un filtro pasabajo, cuya frecuencia de corte es ajustable mediante registros.

Todo este proceso se realiza dentro del módulo de la IMU, y aun así sobrevive algo de ruido eléctrico las señales crudas del acelerómetro, por lo tanto fue necesario la implementación de un filtrado digital.

En el prototipo, este filtrado se realizó en el Arduino. Además se realizaron ensayos con un filtro FIR (promediador) y un IIR (primer orden), para evaluar el desempeño de cada uno y finalmente decidir cual se aplicaría mediante MATLAB™ en el dispositivo final. El giróscopo no precisa de un filtrado digital.

En el anexo B, en el código de programación, se expone la implementación del filtrado digital.

### Acondicionamiento

El acondicionamiento de las señales consistió en transformar los datos crudos sin significado en información útil.

Más precisamente, fue el proceso de transformar las aceleraciones y velocidades angulares obtenidas, en ángulos de navegación, mediante las técnicas mencionadas anteriormente. Hay que recordar que los ángulos determinados por medio del giróscopo involucran una integración numérica, y por lo tanto tienen asociado un error de deriva inherente. Es por ello que se hizo hincapié en que  $T_s$  sea constante, para que la integración no introduzca error adicional.

Los sensores se colocan en las extremidades de manera que detecten el movimiento en el plano sagital, por lo tanto, según la orientación de dichos sensores, los ángulos afectados son pitch y roll, manteniendo se yaw prácticamente constante y sin variación, por ser perpendicular a este plano. Es por esta razón que en la programación solo se realizaron cálculos para el ángulo pitch y roll.

### Almacenamiento

La memoria EEPROM utilizada en el prototipo es de 1024 Bytes. Se complementó esta memoria con la memoria EEPROM interna del Atmega328p, cuya capacidad es de 1KB. Utilizando ambas memorias se tuvo un espacio de 2KB aproximadamente, y siendo que cada dato ocupa 2 bytes, se pudo capturar hasta 1000 muestras para cada ensayo, lo cual era suficiente.

Las memorias microSD tienen capacidad del orden de los GigaBytes, y superan en muchos aspectos a una memoria eeprom. Su programación se realizó mediante librerías propias de Arduino que comandan los protocolos SPI, lo cual hacia más versátil su programación.

## Ensayos

### **Primer Parte: Caracterización**

Como banco de pruebas se utilizó un péndulo mecánico, colocando el prototipo solidario al brazo rígido de dicho péndulo (figura 31). Esto permitió realizar pruebas tanto dinámicas como estáticas, y así determinar completamente la precisión, la exactitud, la linealidad y los errores del instrumento, y así culminar con la primera etapa: la caracterización.

Se decidió solo medir el ángulo *roll*, ya que los resultados se extendían para el ángulo *pitch*, e incluso para cualquier tipo de ángulo.

El procedimiento de los ensayos dinámicos consistió en iniciar el registro de datos en la posición de inclinación nula, trasladar el péndulo a la posición de 90°, y soltarlo para que el efecto de la gravedad describa sobre el instrumento un movimiento oscilatorio amortiguado.

Las mediciones dinámicas realizadas con el acelerómetro siguiendo dicho procedimiento fueron:

- 1) Medición del valor crudo del vector de aceleración de la gravedad en el eje z.
- 2) Medición del ángulo *roll*, sin filtrar.
- 3) Medición del ángulo *roll*, con filtro FIR del tipo promediador con 10 coeficientes.
- 4) Medición del ángulo *roll*, con filtro IIR de primer orden.

En el caso del giroscopio, las pruebas realizadas con dicho procedimiento fueron:

- 1) Medición del ángulo *roll*.

El procedimiento de los ensayos estáticos consistió en iniciar la captura de datos también en la posición de inclinación nula, y luego trasladar a una velocidad constante el péndulo a distintas posiciones elegidas por el operador,

marcando generalmente dos o tres ángulos.

Las mediciones estáticas realizadas con el acelerómetro siguiendo dicho procedimiento fueron:

- 1) Medición del ángulo *roll*, con filtro FIR del tipo promediador con 10 coeficientes.
- 2) Medición del ángulo *roll*, con filtro IIR de primer orden.

En el caso del giróscopo, las pruebas realizadas con dicho procedimiento fueron:

- 1) Medición del ángulo *roll*.

Además, se realizaron dos pruebas adicionales, una dinámica y otra estática, combinando ambos, acelerómetro y giróscopo, mediante un filtro complementario.



Figura 31

Las mediciones obtenidas se contrastaron con una circunferencia graduada cuyo centro coincidía con el eje de rotación del péndulo (figura 32), y también con una aplicación Android de uso libre para teléfono móvil, que permitía utilizar a dicho teléfono como nivel de inclinación (figura 31) .

La aplicación Android hace uso de los acelerómetros incorporados en el *Smartphone* para determinar la inclinación del mismo. El móvil con la aplicación funcionando se colocó solidario al brazo del péndulo, de manera que describa el mismo movimiento que el dispositivo y sirva como patrón para validar las mediciones.

A continuación, se exponen las gráficas que arrojaron los ensayos detallados en el apartado anterior. En algunas pruebas se tomaron 500 muestras, y en otras 1000 muestras, según la información necesaria para una correcta visualización del fenómeno.

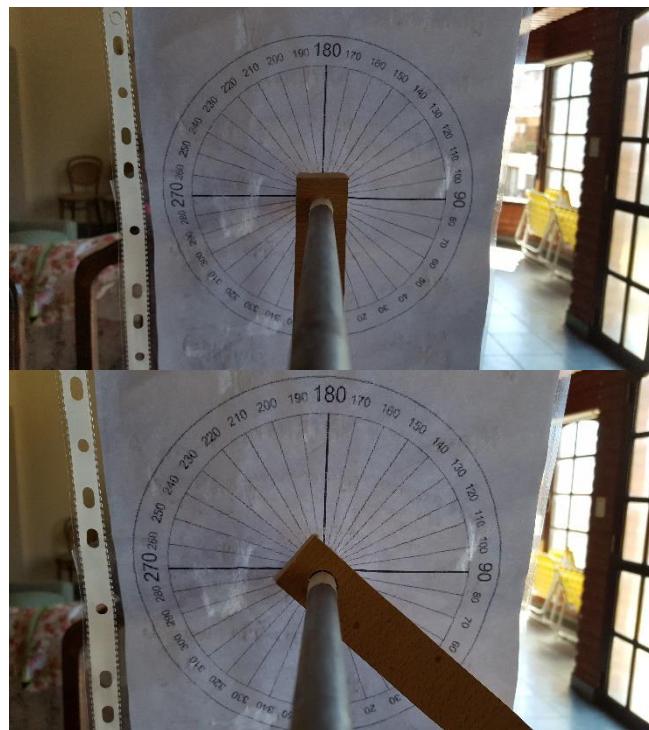
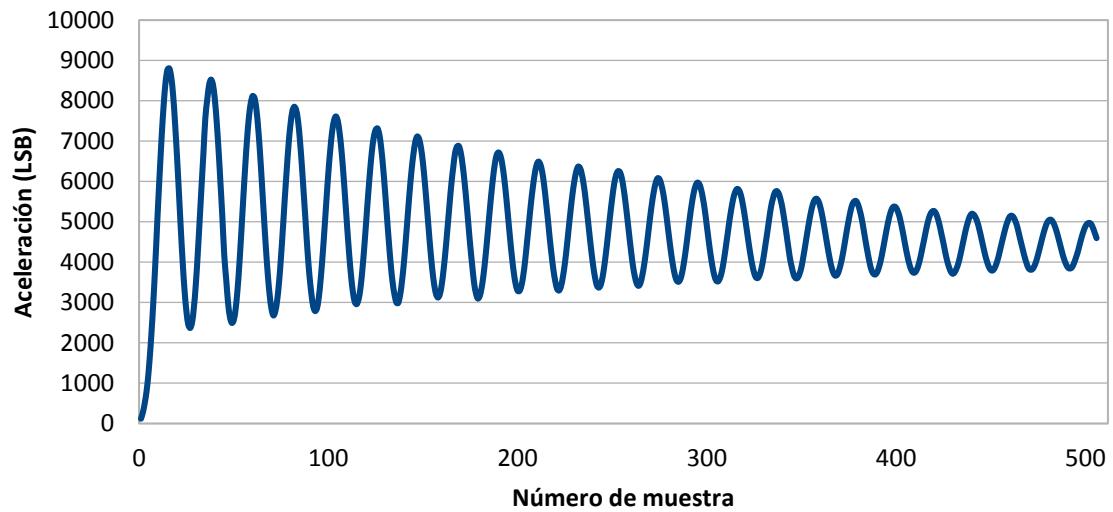


Figura 32

### Pruebas Dinámicas del Acelerómetro

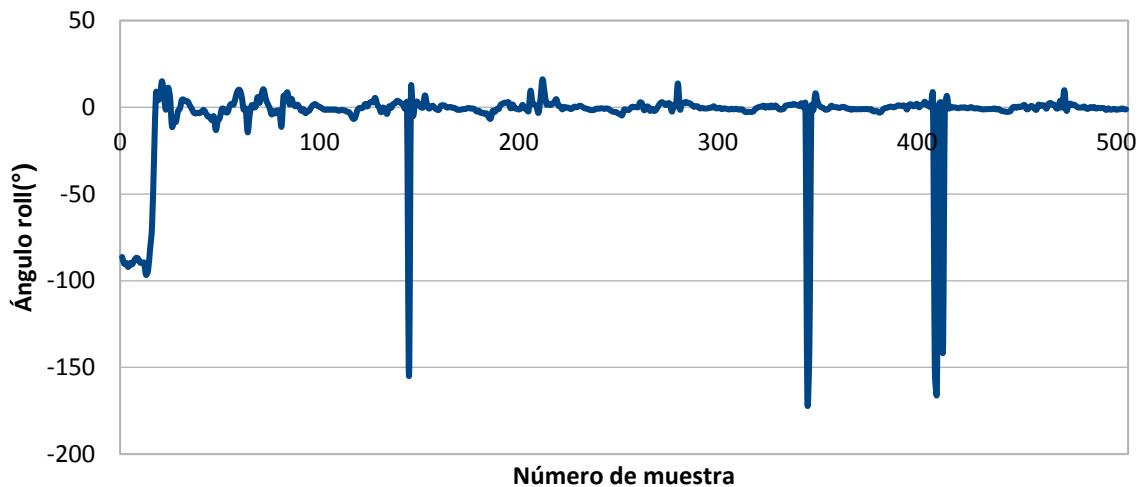
- 1) Medición del valor crudo del vector de aceleración de la gravedad en el eje z.

#### **Medición del vector de aceleración g en el eje z**



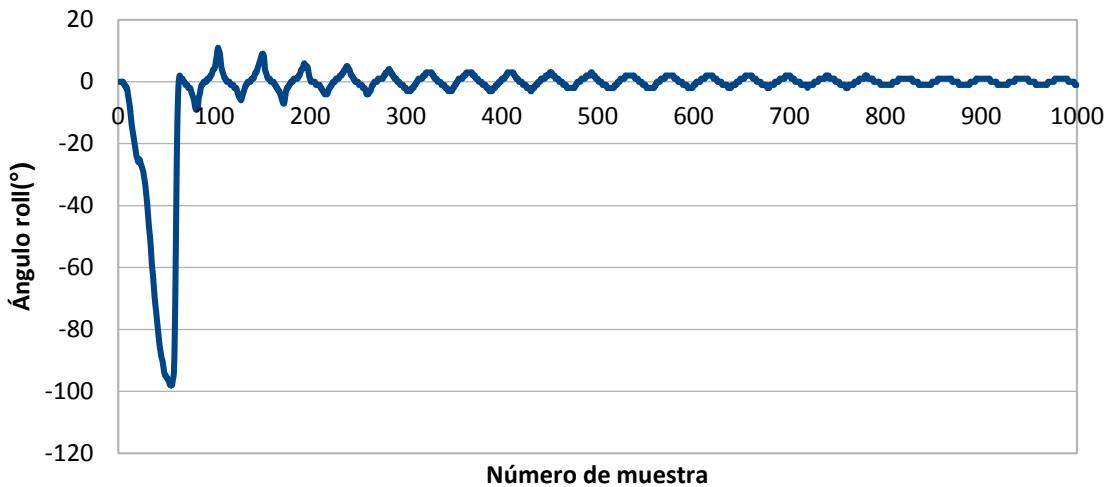
- 2) Medición del ángulo roll, sin filtrar.

#### **Medición del ángulo roll, sin filtrar**



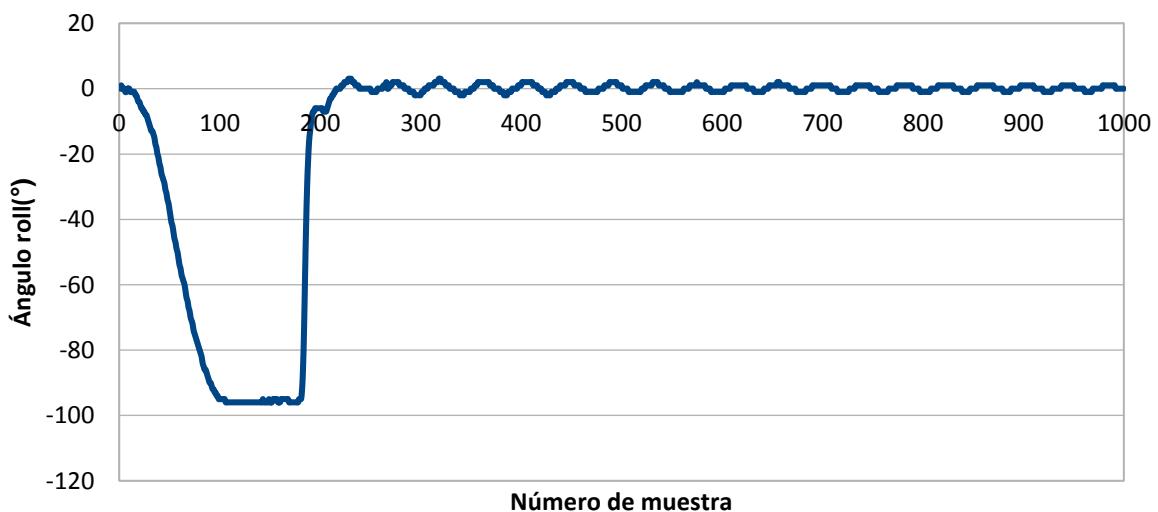
- 3) Medición del ángulo roll, con filtro FIR del tipo promediador con 10 coeficientes.

### **Medición del ángulo roll, con filtro FIR del tipo promediador con 10 coeficientes**



- 4) Medición del ángulo roll, con filtro IIR de primer orden.

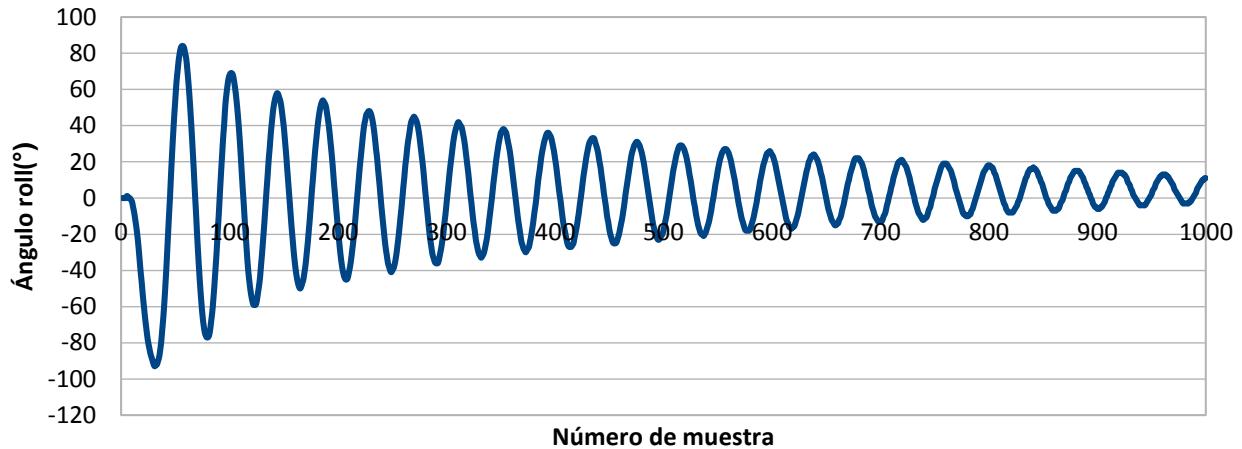
### **Medición del ángulo roll, con filtro IIR de primer orden**



## Pruebas Dinámicas del Giróscopo

- 1) Medición del ángulo *roll*.

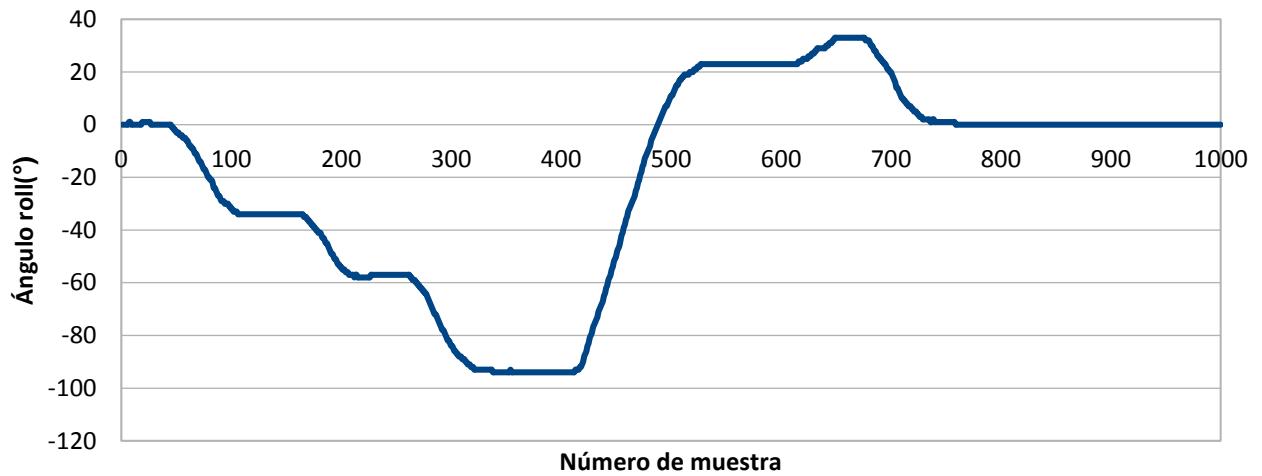
### **Medición del ángulo *roll***



## Pruebas Estáticas del Acelerómetro

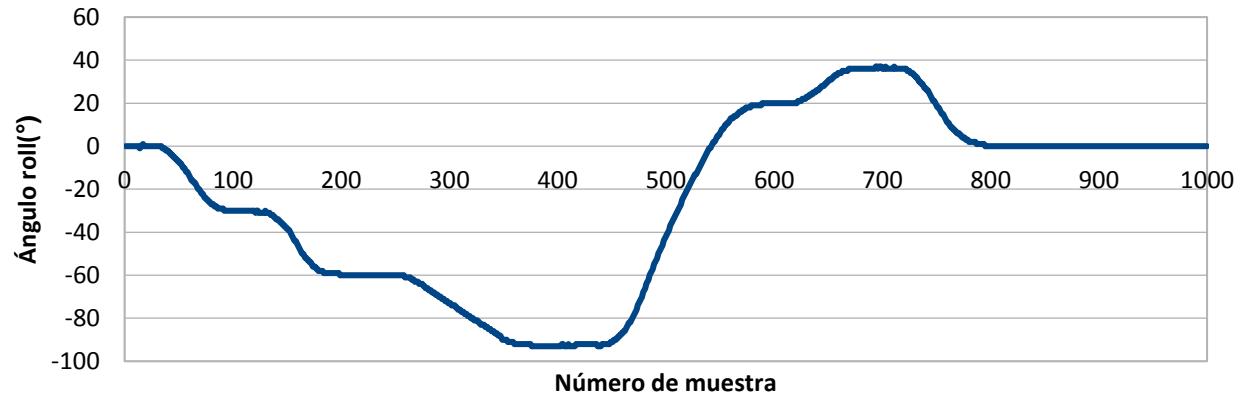
- 1) Medición del ángulo *roll*, con filtro FIR del tipo promediador con 10 coeficientes.

### **Medición del ángulo *roll*, con filtro FIR del tipo promediador con 10 coeficientes**



2) Medición del ángulo *roll*, con filtro IIR de primer orden.

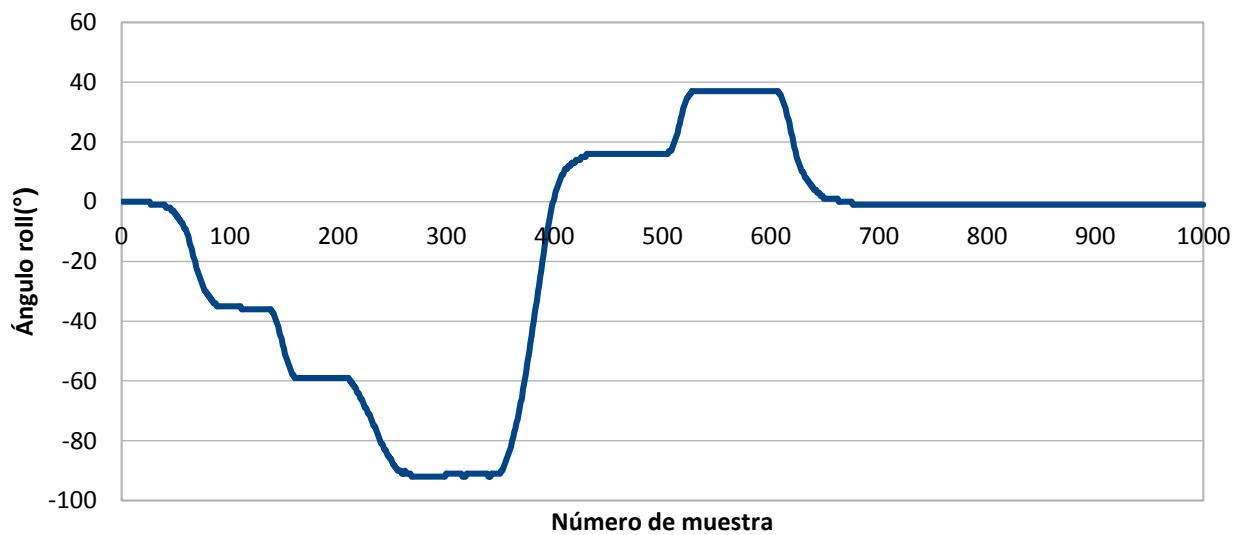
## Medición del ángulo roll, con filtro IIR de primer orden



### Pruebas Estáticas del Giróscopo

1) Medición del ángulo *roll*.

## Medición del ángulo *roll*



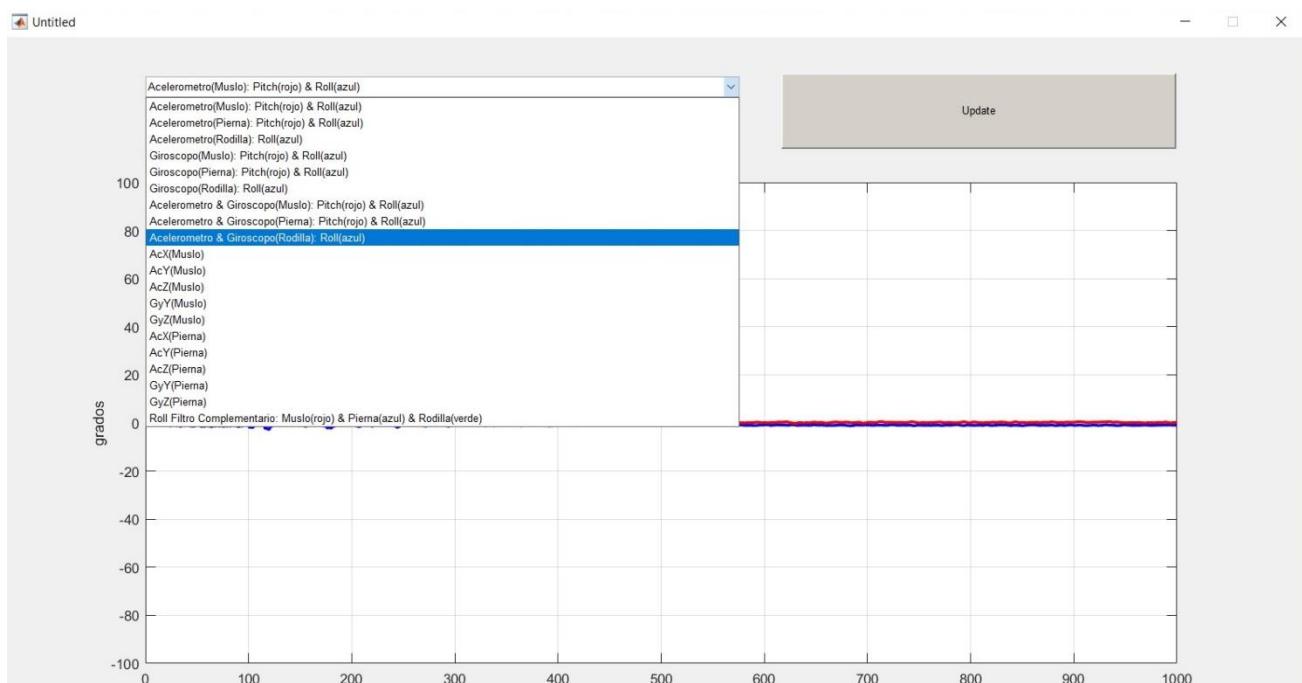
## **Segunda Parte: Registro de la Marcha Humana**

Una vez caracterizado completamente la IMU MPU-6050 y constatado su correcto funcionamiento como indicador de posición, se procedió al desarrollo del dispositivo final.

Las pruebas a realizar con el dispositivo final fueron directamente sobre el individuo bajo estudio. Para ello se colocaría el dispositivo en la cadera, extendiendo los sensores a las extremidades inferiores, uno sobre el muslo y otro sobre la pierna.

En esta disposición el sistema es capaz de capturar todas las señales de movimiento implicadas en la marcha, relativas al muslo y la pierna. El procesamiento de dichas señales se realizó en la plataforma de MATLAB™. Más precisamente, se desarrolló una *interfaz gráfica de usuario* (GUI) para la visualización de las señales tanto crudas como procesadas. Es decir la GUI ofrecía todos los datos: aceleraciones, velocidades, ángulos. El objetivo de la GUI fue brindar al usuario disponibilidad gráfica de todos los datos almacenados, únicamente conectando la memoria microSD y actualizando la interfaz.

En la figura 33 se ve una captura de la GUI, presentando todos los datos disponibles.



*Figura 33*

A modo de prueba, antes de ensayar sobre el individuo, se realizó un ensayo dinámico y estático como los anteriores, utilizando el péndulo, dejando fijo el sensor correspondiente al muslo, y solidario al brazo del péndulo el sensor correspondiente a la pierna. El objetivo de esta prueba fue constatar que se obtenían los mismos resultados que en el prototipo, y así tener la certeza que los sensores actuarían como indicadores de posición.

Primero se realizó el ensayo dinámico. En la figura 34 se observa el filtrado (naranja), de las señales del acelerómetro (azul) del eje x de la pierna. Más abajo, se observa lo mismo pero en el eje y de la pierna.

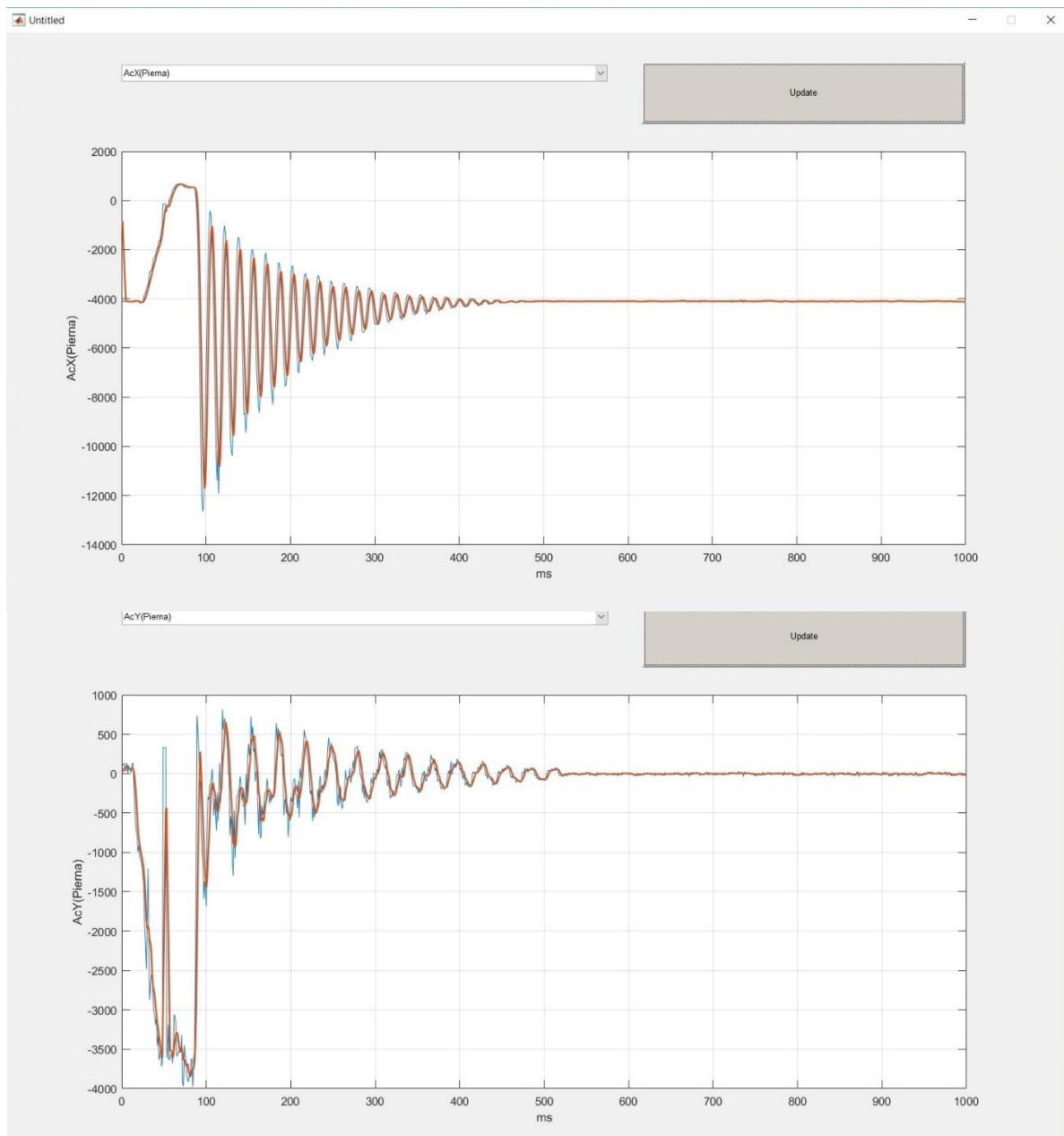


Figura 34

Finalmente, en la figura 35 se observa el ángulo roll del muslo (rojo), de la pierna (azul) y de la rodilla (verde) de la prueba dinámica.

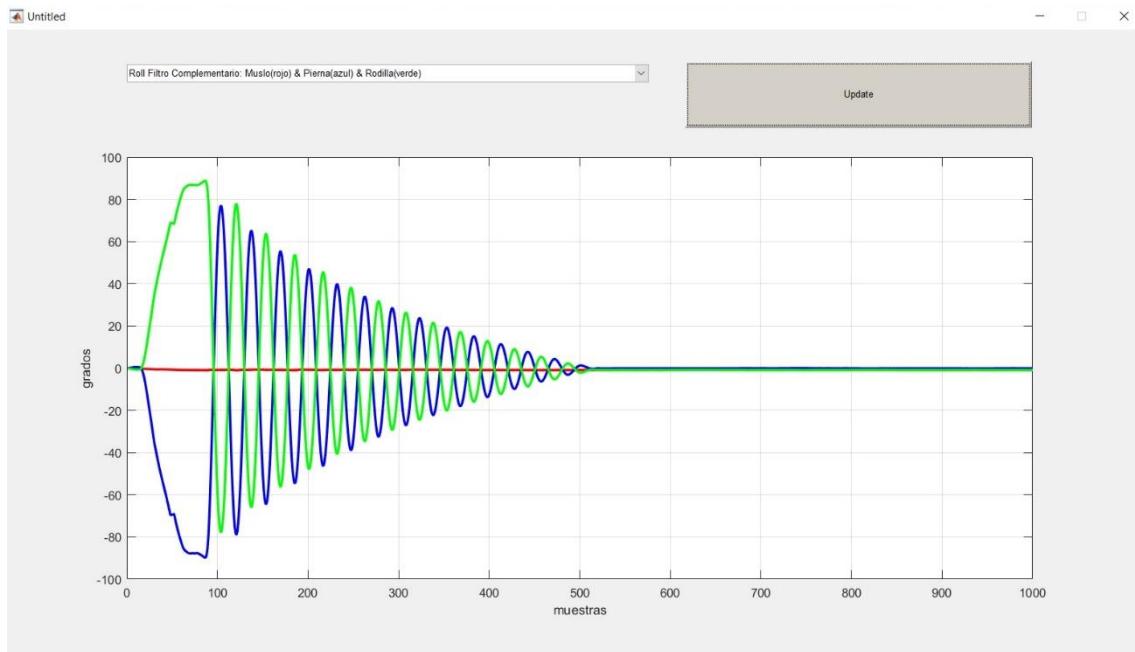


Figura 35

Luego se procedió con el ensayo estático. Aquí los efectos del filtrado se observaron de mejor manera (figura 36).

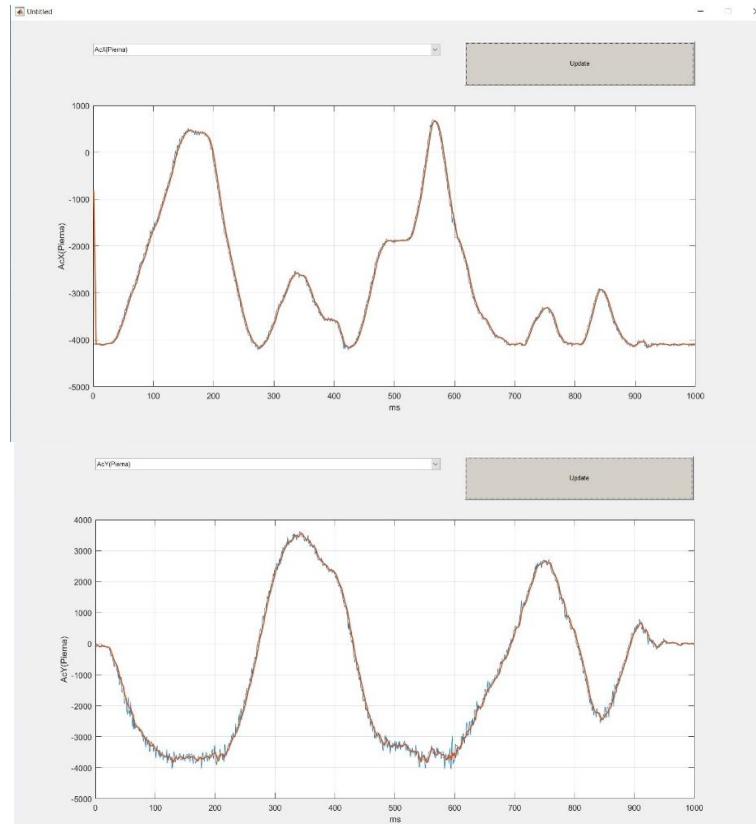


Figura 36

En la figura 37 se expone la captura de la GUI que muestra el ángulo roll durante el ensayo estático del muslo (rojo), de la pierna (azul) y de la rodilla (verde).

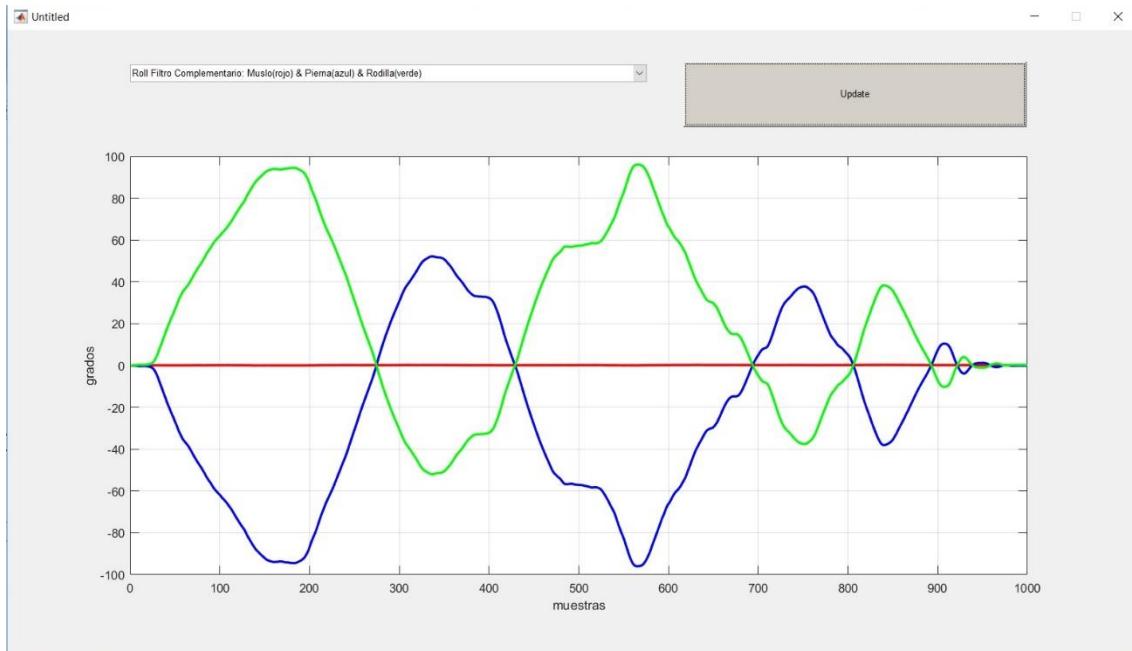


Figura 37

Finalmente abandonamos el péndulo para volcarnos directamente sobre los ensayos de marcha humana en un paciente. La disposición fue la descripta anteriormente, la cual se puede ver en la figura 38.



Figura 38

El individuo bajo estudio tiene el control del dispositivo. Colocado en su cadera, tiene a su alcance un panel con dos pulsadores A y B. El pulsador A al ser presionado por primera vez ejecuta la rutina de calibración, por lo cual el sujeto debe permanecer completamente erguido mientras el sistema detecta la posición de inclinación cero de los miembros. Un led indicador rojo permanece encendido mientras dura la calibración y se apaga cuando esta termina. Finalizada la calibración se presiona nuevamente el pulsador A, destellando una única vez el led indicador rojo, y dando inicio a la rutina de captura de movimiento. Es decir,

cuando el sujeto bajo estudio presiona por segunda vez el pulsador A, debe iniciar su marcha, de manera rectilínea. Con recorrer 15 metros es suficiente. Al finalizar, el paciente presiona el pulsador B, que enciende un led indicador verde y almacena de manera fija los datos en la microSD.

Luego de estos pasos se da por concluido el ensayo, teniendo ya la posibilidad de colocar la tarjeta microSD en el ordenador para observar los datos en la GUI.

Finalmente, se realizó dicho ensayo siendo el sujeto bajo estudio el autor de este trabajo. Como se dijo anteriormente se tenía como objetivo, aparte de capturar la marcha, detectar el ángulo de la rodilla.

El ángulo flexión/extensión de la rodilla se define y se calcula de la siguiente manera:

$$\text{Rodilla} = \text{Muslo} - \text{Pierna}$$

Es decir, el ángulo de la rodilla se define como el ángulo roll del muslo menos el ángulo roll de la pierna. Es por ello que en los ensayos dinámicos y estáticos previos, se observaba el ángulo de la rodilla como el ángulo de la pierna invertido, debido a que el muslo se encontraba fijo marcando ángulo cero. En la figura 39 se esquematiza el ángulo de la rodilla.

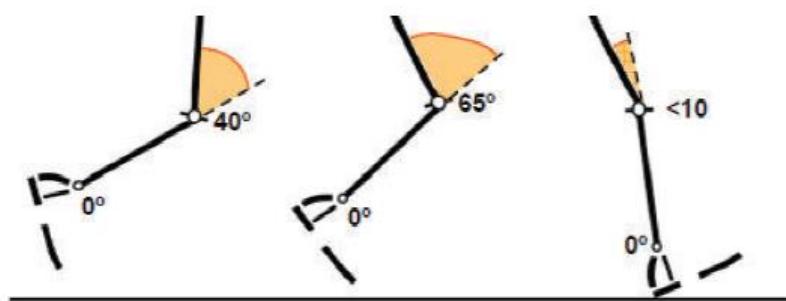


Figura 39

En la figura 40 se observa el ángulo de la rodilla obtenido durante la marcha. Se observa que hubo un tiempo en que hubo movimiento, correspondiente al A primera vista no se observa a simple vista el angulo de la rodilla, si se lo compara directamente con la curva teórica de flexión/extensión de la rodilla. Es decir, a partir de la marcha se debía obtener la curva correspondiente al paciente de flexión/extensión de rodilla.

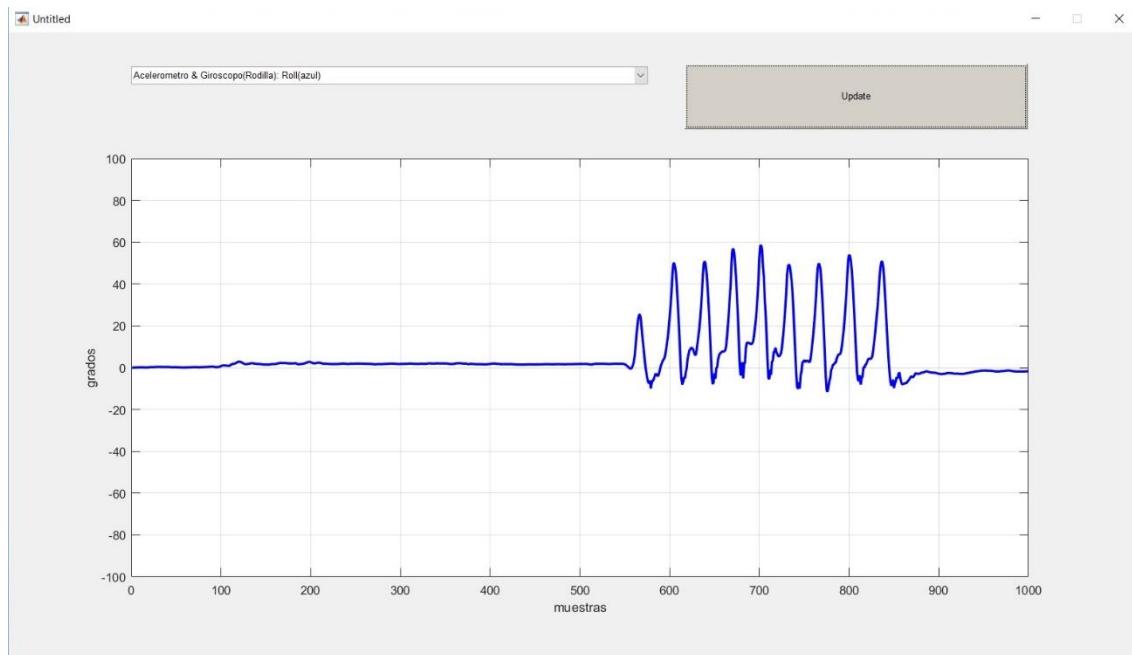


Figura 40

Para lograr esto, se desarrolló otro algoritmo en MATLAB™, independiente de la GUI que permitiera obtener la curva de la rodilla correspondiente al ciclo de marcha promedio del sujeto.

## Resultados

### **Primer Parte: Caracterización**

#### Pruebas Dinámicas del Acelerómetro

- 1) Medición del valor crudo del vector de aceleración de la gravedad en el eje z:

Este ensayo se realizó para comprobar la respuesta del acelerómetro a la gravedad. Los datos obtenidos describieron el movimiento oscilatorio amortiguado, con un valor medio de 4096 LSB aproximadamente, el cual es el valor crudo de aceleración correspondiente a 1g, para la sensibilidad seleccionada.

- 2) Medición del ángulo *roll*, sin filtrar:

El objetivo era observar el movimiento oscilatorio amortiguado, sin tratar la señal y observar el efecto del ruido en las muestras. Sólo se detectó en ángulo de inicio de -90°, y luego el acelerómetro no pudo seguir el movimiento. En esta prueba queda demostrado la necesidad de un filtrado, ya que el nivel de perturbaciones podía contribuir a una información errónea.

- 3) Medición del ángulo *roll*, con filtro FIR del tipo promediador con 10 coeficientes:

Con la aplicación de un filtro promediador de 10 coeficientes, las perturbaciones son prácticamente eliminadas. El acelerómetro sigue correctamente el movimiento durante la traslación paulatina desde 0° a -90°, pero pierde respuesta al comienzo del movimiento oscilatorio amortiguado aplicado por la gravedad.

- 4) Medición del ángulo *roll*, con filtro IIR de primer orden:

Aquí se comprobó que un filtro IIR de primer orden cuyo costo de programación es menor, cumple correctamente con los requisitos de una señal libre de ruido. En este caso la posición de -90° se mantuvo estacionaria un cierto tiempo para contrastarlo con el patrón. Se obtuvo

un error de  $<1^\circ$ . Luego de tres ensayos tratando de reproducir la dinámica del movimiento del péndulo, se comprueba la hipótesis de que el acelerómetro no es apto para seguir movimientos rápidos.

### Pruebas Dinámicas del Giróscopo

#### 1) Medición del ángulo *roll*:

Como se esperaba, el giróscopo reprodujo correctamente la dinámica de movimiento involucrada en el ensayo. Sin necesidad de filtrado presento una señal sin perturbaciones, destacando su inmunidad al ruido. Como también se predijo, a la larga existe un error de deriva (*drift*). El mismo se manifiesta como un valor medio distinto de  $0^\circ$  en las últimas muestras.

### Pruebas Estáticas del Acelerómetro

#### 1) Medición del ángulo *roll*, con filtro FIR del tipo promediador con 10 coeficientes:

Siguiendo el procedimiento, se marcaron distintos ángulos con el péndulo, y se contrastaron con la circunferencia graduada y el medidor de inclinación patrón del teléfono móvil. El error observado fue de  $<1^\circ$  en todas las mediciones.

#### 2) Medición del ángulo *roll*, con filtro IIR de primer orden:

Las mediciones no sufren cambios al utilizar el filtro IIR. El error observado también fue de  $<1^\circ$ .

### Pruebas Estáticas del Giróscopo

#### 1) Medición del ángulo *roll*:

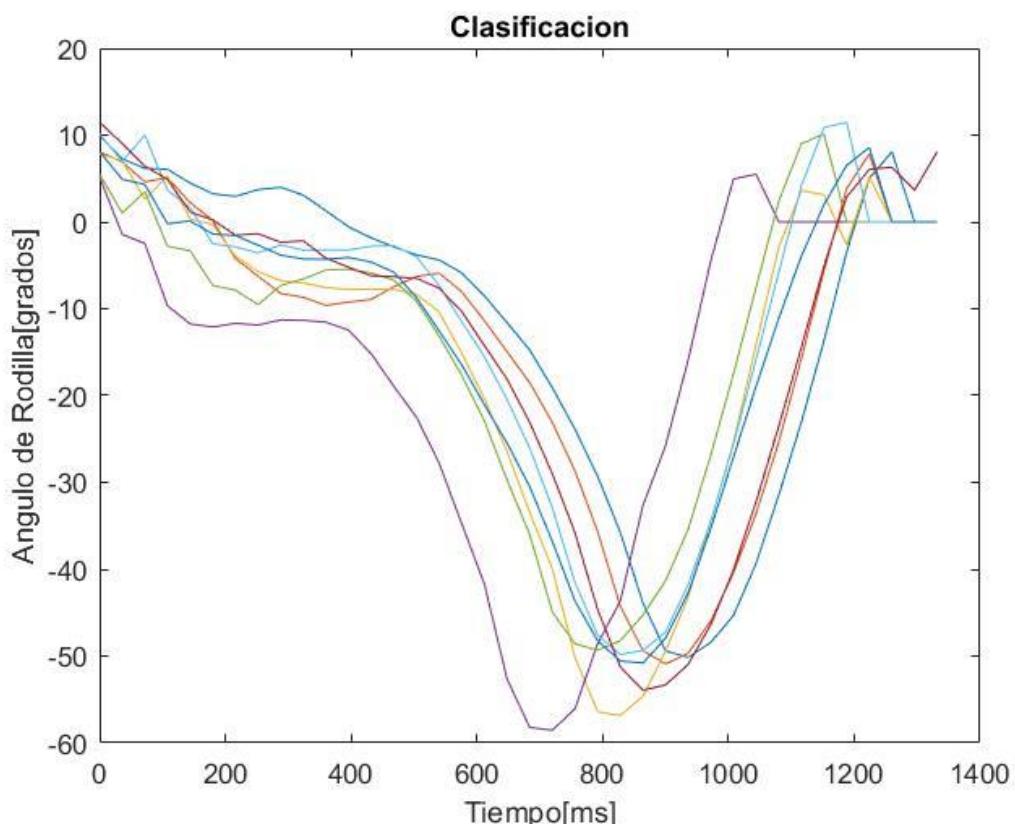
El giróscopo se desenvolvió correctamente en las mediciones estáticas. El error de cada medida fue de  $<1^\circ$ , y el error de deriva (*drift*) fue de  $-1^\circ$ , significativamente menor que en el caso dinámico, debido a que se pasó pocas veces por el punto de  $0^\circ$ .

Para compensar la debilidad de un sensor con la fortaleza de otro, es posible recurrir a un filtro complementario. Se realizó una prueba dinámica y una estática aplicando dicho filtro y combinando las salidas del acelerómetro y el giróscopo.

### **Segunda Parte: Registro de la Marcha Humana**

El algoritmo para obtener el ángulo de la rodilla correspondiente al ciclo de marcha consistía en tres procesos: *clasificación, normalización y ajuste*.

La clasificación consistía básicamente en obtener los intervalos de la curva de la rodilla correspondiente a los distintos ciclos de marcha. Esto se realizó mediante una técnica de detección de máximos locales teniendo en cuenta el período de la señal, obtenido con otra técnica previamente. Una vez obtenidos los distintos máximos correspondientes al inicio de un ciclo, se separaron los distintos ciclos, lo cual se puede observar en la figura 41.



*Figura 41*

Como se puede observar, los ciclos se desarrollan en intervalos de tiempo distintos, debido a que el ciclo de marcha es irregular, aunque siempre manteniéndose dentro de ciertos parámetros. Para corregir esto, se procede a la normalización, que es el proceso de independizar los ciclos del tiempo, y hacerlos función del porcentaje de ciclo de marcha. Este proceso se realizó mediante la interpolación cúbica de cada ciclo sobre un rango de 0 a 100. Este proceso se observa en la figura 42.

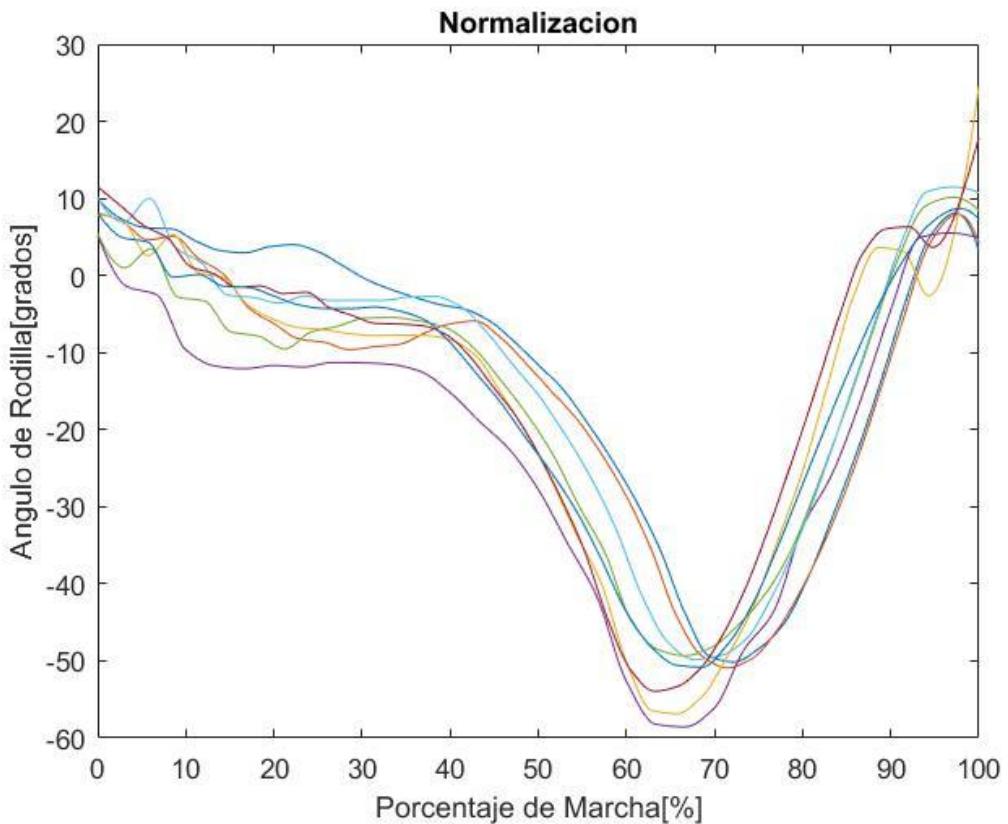


Figura 42

Finalmente, la curva cinemática del ángulo de la rodilla correspondiente al ciclo de marcha promedio se obtuvo aplicando el proceso de ajuste. El mismo consistía en promediar todas las curvas (cabe aclarar que no se podría promediar sin normalizar anteriormente), y a obtener una curva de ajuste de dicha curva promedio.

En la figura 43 se observa el resultado final del algoritmo. Se puede observar la gran similitud con la curva cinemática teórica de la rodilla.

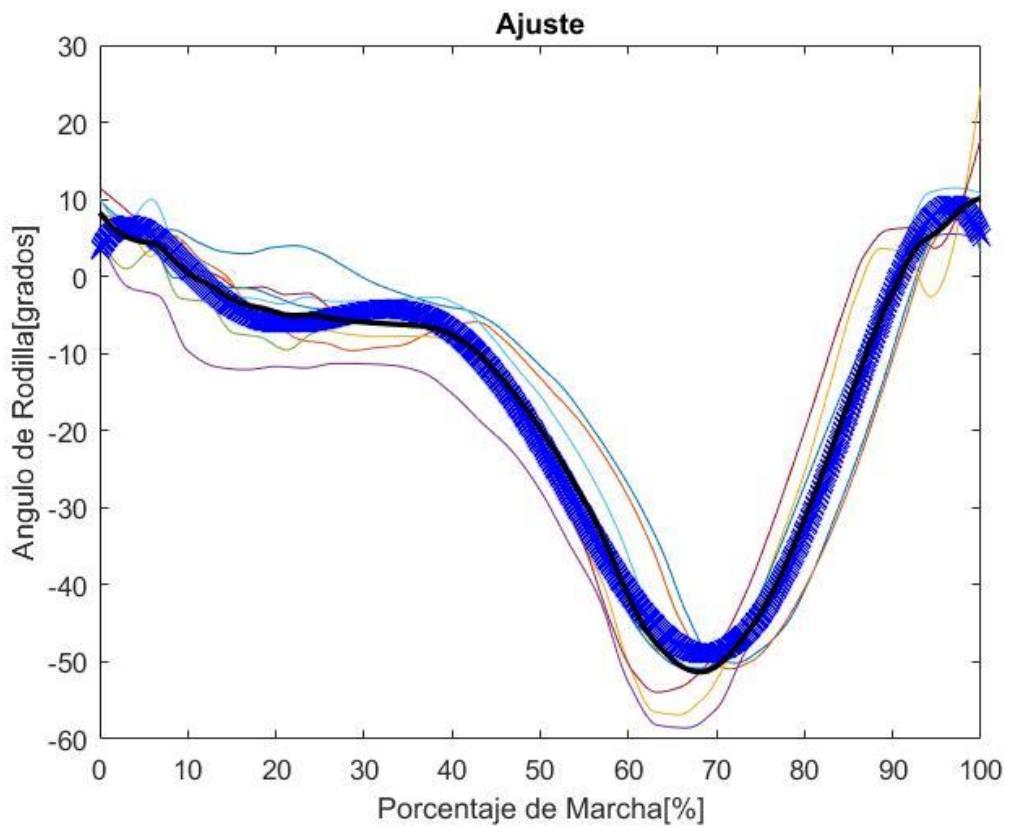


Figura 43

De aquí se deduce que el dispositivo puede obtener la curva cinemática de la rodilla correspondiente al ciclo de marcha para cualquier individuo, y así obtener un análisis cuantitativo de la capacidad de flexión/extensión de la rodilla.

## Conclusión

Mediante el estudio previo de los dispositivos involucrados y el posterior análisis detallado de los resultados arrojados por los ensayos programados, se logró la correcta caracterización tanto del acelerómetro como el giróscopo. Presentan característica lineal, repetitividad en la medida y error de  $>1^\circ$ .

Teniendo esto en cuenta, se comprueba la hipótesis postulada en el inicio de este trabajo. Ambos sensores pueden ser utilizados correctamente como indicadores de posición para medir la inclinación de los miembros inferiores y obtener el ciclo de la marcha humana.

El mayor potencial del acelerómetro y el giróscopo sale a la luz cuando se combinan mediante un filtro complementario, presentando una medida ausente de deriva y de errores en tiempos cortos.

Gracias a los resultados descritos anteriormente, el dispositivo final registró de manera precisa las señales emitidas por los sensores durante la marcha humana, y se reprodujo correctamente las curvas relacionadas con el ciclo de marcha, y más específicamente se obtuvo la curva cinemática de la rodilla, de gran correlación con la curva teórica.

Con este resultado, la aplicación inmediata de este trabajo es utilizar dicho dispositivo final como instrumento de apoyo para el especialista en rehabilitación, para realizar un seguimiento cuantitativo de la capacidad de flexión/extensión de la rodilla del paciente, y no depender exclusivamente de la valoración cualitativa propia del especialista. Un ejemplo es durante el tratamiento de pacientes con gonartrosis, donde el especialista evalúa el avance de la patología mediante una escala de dolor, es decir, flexiona la rodilla hasta cierto punto y según el dolor que siente el paciente, valora la mejoría en la rehabilitación. Este aparato, para este caso particular, evitaría esta práctica de manera frecuente, ya que mediante un ensayo de captura de la marcha humana permitiría al especialista observar la curva cinemática de la rodilla y observar si existió una mejoría, por contraste directo con la curva teórica esperada, evaluada dentro de una banda de confianza.

## Bibliografía

- [1] T. J. Maloney, Electrónica Industrial Moderna, México: Pearson, 2006.
- [2] H. Goldstein, Mecánica Clásica, Reverté, 2000.
- [3] M. Morales, «T-Bem,» 4 Febrero 2017. [En línea]. Available: <http://learn.teslabem.com/fundamentos-del-protocolo-i2c-aprende/>.
- [4] K. Navarro, «Panama Hitek,» 15 Octubre 2014. [En línea]. Available: <http://panamahitek.com/como-funciona-el-protocolo-spi/>.
- [5] C. Dra. Marco Sanz, «Cinesiología de la Marcha Humana Normal,» 2010.
- [6] InvenSense™, «MPU-6000 and MPU-6050 Product Specification Revision 3.4 Document Number: PS-MPU-6000A-00,» 2013.
- [7] InvenSense™, «MPU-6000 and MPU-6050 Register Map and Descriptions Revision 4.2 Document Number: RM-MPU-6000A-00,» 2013.
- [8] Atmel™, «8-bit AVR Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash ATmega48P/88P/168P/328P Datasheet Revision 8025FS,» 2008.
- [9] Atmel™, «SEEPROM AT24C04C/08C Datasheet Revision 8787F,» 2016.
- [10] B. Sensortec™, «BMA220 Datasheet Revision 1.10 Document Number: BST-BMA220-DS003-07,» 2011.
- [11] F. Semiconductors™, «“Tilt Sensing Using a Three-Axis Accelerometer” Revision 6 Document Number: AN3461 Application Note,» 2013.

## Anexo A

**MPU-6050**

Datasheet de Especificaciones



InvenSense Inc.  
1197 Borregas Ave, Sunnyvale, CA 94089 U.S.A.  
Tel: +1 (408) 988-7339 Fax: +1 (408) 988-8104  
Website: [www.invensense.com](http://www.invensense.com)

Document Number: PS-MPU-6000A-00  
Revision: 3.4  
Release Date: 08/19/2013

# MPU-6000 and MPU-6050 Product Specification Revision 3.4



## CONTENTS

1	REVISION HISTORY .....	.5
2	PURPOSE AND SCOPE .....	.6
3	PRODUCT OVERVIEW .....	.7
3.1	MPU-60X0 OVERVIEW .....	.7
4	APPLICATIONS .....	.9
5	FEATURES .....	.10
5.1	GYROSCOPE FEATURES .....	.10
5.2	ACCELEROMETER FEATURES .....	.10
5.3	ADDITIONAL FEATURES .....	.10
5.4	MOTION PROCESSING .....	.11
5.5	CLOCKING .....	.11
6	ELECTRICAL CHARACTERISTICS .....	.12
6.1	GYROSCOPE SPECIFICATIONS .....	.12
6.2	ACCELEROMETER SPECIFICATIONS .....	.13
6.3	ELECTRICAL AND OTHER COMMON SPECIFICATIONS .....	.14
6.4	ELECTRICAL SPECIFICATIONS, CONTINUED .....	.15
6.5	ELECTRICAL SPECIFICATIONS, CONTINUED .....	.16
6.6	ELECTRICAL SPECIFICATIONS, CONTINUED .....	.17
6.7	I <sup>2</sup> C TIMING CHARACTERIZATION .....	.18
6.8	SPI TIMING CHARACTERIZATION (MPU-6000 ONLY) .....	.19
6.9	ABSOLUTE MAXIMUM RATINGS .....	.20
7	APPLICATIONS INFORMATION .....	.21
7.1	PIN OUT AND SIGNAL DESCRIPTION .....	.21
7.2	TYPICAL OPERATING CIRCUIT .....	.22
7.3	BILL OF MATERIALS FOR EXTERNAL COMPONENTS .....	.22
7.4	RECOMMENDED POWER-ON PROCEDURE .....	.23
7.5	BLOCK DIAGRAM .....	.24
7.6	OVERVIEW .....	.24
7.7	THREE-AXIS MEMS GYROSCOPE WITH 16-BIT ADCS AND SIGNAL CONDITIONING .....	.25
7.8	THREE-AXIS MEMS ACCELEROMETER WITH 16-BIT ADCS AND SIGNAL CONDITIONING .....	.25
7.9	DIGITAL MOTION PROCESSOR .....	.25
7.10	PRIMARY I <sup>2</sup> C AND SPI SERIAL COMMUNICATIONS INTERFACES .....	.25
7.11	AUXILIARY I <sup>2</sup> C SERIAL INTERFACE .....	.26



## 6 Electrical Characteristics

### 6.1 Gyroscope Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V $\pm$ 5% or VDD, T<sub>A</sub> = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
<b>GYROSCOPE SENSITIVITY</b>						
Full-Scale Range	FS_SEL=0 FS_SEL=1 FS_SEL=2 FS_SEL=3		$\pm 250$ $\pm 500$ $\pm 1000$ $\pm 2000$		%/s %/s %/s %/s	
Gyroscope ADC Word Length			16		bits	
Sensitivity Scale Factor	FS_SEL=0 FS_SEL=1 FS_SEL=2 FS_SEL=3		131 65.5 32.8 16.4		LSB/(%) LSB/(%) LSB/(%) LSB/(%)	
Sensitivity Scale Factor Tolerance		-3	$\pm 2$	+3	%	
Sensitivity Scale Factor Variation Over Temperature					%	
Nonlinearity			0.2		%	
Cross-Axis Sensitivity			$\pm 2$		%	
<b>GYROSCOPE ZERO-RATE OUTPUT (ZRO)</b>						
Initial ZRO Tolerance	25°C		$\pm 20$		%/s	
ZRO Variation Over Temperature	-40°C to +85°C		$\pm 20$		%/s	
Power-Supply Sensitivity (1-10Hz)	Sine wave, 100mVpp; VDD=2.5V		0.2		%/s	
Power-Supply Sensitivity (10 - 250Hz)	Sine wave, 100mVpp; VDD=2.5V		0.2		%/s	
Power-Supply Sensitivity (250Hz - 100kHz)	Sine wave, 100mVpp; VDD=2.5V		4		%/s	
Linear Acceleration Sensitivity	Static		0.1		%/s/g	
<b>SELF-TEST RESPONSE</b>						
Relative	Change from factory trim	-14		14	%	1
<b>GYROSCOPE NOISE PERFORMANCE</b>						
Total RMS Noise	FS_SEL=0		0.05		%/s-rms	
Low-frequency RMS noise	DLPFCFG=2 (100Hz)		0.033		%/s-rms	
Rate Noise Spectral Density	Bandwidth 1Hz to 10Hz		0.005		%/s/√Hz	
	At 10Hz					
<b>GYROSCOPE MECHANICAL FREQUENCIES</b>						
X-Axis		30	33	36	kHz	
Y-Axis		27	30	33	kHz	
Z-Axis		24	27	30	kHz	
<b>LOW PASS FILTER RESPONSE</b>						
	Programmable Range	5		256	Hz	
<b>OUTPUT DATA RATE</b>						
	Programmable	4		8,000	Hz	
<b>GYROSCOPE START-UP TIME</b>						
ZRO Settling (from power-on)	DLPFCFG=0 to $\pm 1\%$ /s of Final		30		ms	

1. Please refer to the following document for further information on Self-Test: *MPU-6000/MPU-6050 Register Map and Descriptions*



## MPU-6000/MPU-6050 Product Specification

Document Number: PS-MPU-6000A-00  
Revision: 3.4  
Release Date: 08/19/2013

## 6.2 Accelerometer Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, TA = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
<b>ACCELEROMETER SENSITIVITY</b>						
Full-Scale Range	AFS_SEL=0 AFS_SEL=1 AFS_SEL=2 AFS_SEL=3		±2 ±4 ±8 ±16		g g g g	
ADC Word Length	Output in two's complement format		16		bits	
Sensitivity Scale Factor	AFS_SEL=0 AFS_SEL=1 AFS_SEL=2 AFS_SEL=3		16,384 8,192 4,096 2,048		LSB/g LSB/g LSB/g LSB/g	
Initial Calibration Tolerance			±3		%	
Sensitivity Change vs. Temperature	AFS_SEL=0, -40°C to +85°C		±0.02		%/°C	
Nonlinearity	Best Fit Straight Line		0.5		%	
Cross-Axis Sensitivity			±2		%	
<b>ZERO-G OUTPUT</b>						
Initial Calibration Tolerance	X and Y axes		±50		mg	1
	Z axis		±80		mg	
Zero-G Level Change vs. Temperature	X and Y axes, 0°C to +70°C		±35		mg	
	Z axis, 0°C to +70°C		±80		mg	
<b>SELF TEST RESPONSE</b>						
Relative	Change from factory trim	-14		14	%	2
<b>NOISE PERFORMANCE</b>						
Power Spectral Density	@10Hz, AFS_SEL=0 & ODR=1kHz		400		µg/√Hz	
<b>LOW PASS FILTER RESPONSE</b>						
Programmable Range		5		260	Hz	
<b>OUTPUT DATA RATE</b>						
Programmable Range		4		1,000	Hz	
<b>INTELLIGENCE FUNCTION INCREMENT</b>				32	mg/LSB	

1. Typical zero-g initial calibration tolerance value after MSL3 preconditioning
2. Please refer to the following document for further information on Self-Test: *MPU-6000/MPU-6050 Register Map and Descriptions*

Datasheet de Mapa de Registros



InvenSense Inc.  
1197 Borregas Ave, Sunnyvale, CA 94089 U.S.A.  
Tel: +1 (408) 988-7339 Fax: +1 (408) 988-8104  
Website: [www.invensense.com](http://www.invensense.com)

Document Number: RM-MPU-6000A-00  
Revision: 4.2  
Release Date: 08/19/2013

**MPU-6000 and MPU-6050  
Register Map and Descriptions  
Revision 4.2**



## MPU-6000/MPU-6050 Register Map and Descriptions

Document Number: RM-MPU-6000A-00  
Revision: 4.2  
Release Date: 08/19/2013

### 3 Register Map

The register map for the MPU-60X0 is listed below.

Addr (Hex)	Addr (Dec.)	Register Name	Serial IF	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
00	13	SELF_TEST_X	R/W	X_A_TEST[4-2]					XG_TEST[4-0]		
0E	14	SELF_TEST_Y	R/W	YA_TEST[4-2]				YG_TEST[4-0]			
0F	15	SELF_TEST_Z	R/W	Z_A_TEST[4-2]				ZG_TEST[4-0]			
10	16	SELF_TEST_A	R/W	RESERVED	X_A_TEST[1-0]	YA_TEST[1-0]		Z_A_TEST[1-0]			
19	25	SMPRT_DIV	R/W				SMPRT_DIV[7:0]				
1A	26	CONFIG	R/W	-	-	EXT_SYNC_SET[2:0]		DLFF_CFG[2:0]			
1B	27	GYRO_CONFIG	R/W	-	-	-	FS_SEL[1:0]	-	-	-	
1C	28	ACCEL_CONFIG	R/W	X_A_ST	YA_ST	Z_A_ST	AFS_SEL[1:0]				
23	35	FIFO_EN	R/W	TEMP_FIFO_EN	XG_FIFO_EN	YG_FIFO_EN	ZG_FIFO_EN	ACCEL_FIFO_EN	SLV2_FIFO_EN	SLV1_FIFO_EN	SLV0_FIFO_EN
24	36	I2C_MST_CTRL	R/W	MULT_MST_EN	WAIT_FOR_BS	SLV_3_FIFO_EN	I2C_MST_FIFO_EN	P_NSR		I2C_MST_CLK[3:0]	
25	37	I2C_SLV0_ADDR	R/W	I2C_SLV0_RW			I2C_SLV0_ADDR[6:0]				
26	38	I2C_SLV0_REG	R/W				I2C_SLV0_REG[7:0]				
27	39	I2C_SLV0_CTRL	R/W	I2C_SLV0_EN	I2C_SLV0_BYT_SW	I2C_SLV0_REG_DIS	I2C_SLV0_GRP		I2C_SLV0_LEN[3:0]		
28	40	I2C_SLV1_ADDR	R/W	I2C_SLV1_RW			I2C_SLV1_ADDR[6:0]				
29	41	I2C_SLV1_REG	R/W				I2C_SLV1_REG[7:0]				
2A	42	I2C_SLV1_CTRL	R/W	I2C_SLV1_EN	I2C_SLV1_BYT_SW	I2C_SLV1_REG_DIS	I2C_SLV1_GRP		I2C_SLV1_LEN[3:0]		
2B	43	I2C_SLV2_ADDR	R/W	I2C_SLV2_RW			I2C_SLV2_ADDR[6:0]				
2C	44	I2C_SLV2_REG	R/W				I2C_SLV2_REG[7:0]				
2D	45	I2C_SLV2_CTRL	R/W	I2C_SLV2_EN	I2C_SLV2_BYT_SW	I2C_SLV2_REG_DIS	I2C_SLV2_GRP		I2C_SLV2_LEN[3:0]		
2E	46	I2C_SLV3_ADDR	R/W	I2C_SLV3_RW			I2C_SLV3_ADDR[6:0]				
2F	47	I2C_SLV3_REG	R/W				I2C_SLV3_REG[7:0]				
30	48	I2C_SLV3_CTRL	R/W	I2C_SLV3_EN	I2C_SLV3_BYT_SW	I2C_SLV3_REG_DIS	I2C_SLV3_GRP		I2C_SLV3_LEN[3:0]		
31	49	I2C_SLV4_ADDR	R/W	I2C_SLV4_RW			I2C_SLV4_ADDR[6:0]				
32	50	I2C_SLV4_REG	R/W				I2C_SLV4_REG[7:0]				
33	51	I2C_SLV4_DO	R/W				I2C_SLV4_DO[7:0]				
34	52	I2C_SLV4_CTRL	R/W	I2C_SLV4_EN	I2C_SLV4_INT_EN	I2C_SLV4_REG_DIS		I2C_MST_DLY[4:0]			
35	53	I2C_SLV4_DI	R				I2C_SLV4_DI[7:0]				
36	54	I2C_MST_STATUS	R	PASS_THROUGH	I2C_SLV4_DONE	I2C_SLV4_ARB	I2C_SLV4_NACK	I2C_SLV3_NACK	I2C_SLV2_NACK	I2C_SLV1_NACK	I2C_SLV0_NACK
37	55	INT_PIN_CFG	R/W	INT_LEVEL	INT_OPEN	LATCH_INT_EN	INT_RD_CLEAR	FSYNC_INT_LEVEL	FSYNC_INT_EN	I2C_BYPASS_EN	-
38	56	INT_ENABLE	R/W	-	-	-	FIFO_OFLOW_EN	I2C_MST_INT_EN	-	-	DATA_RDY_EN
3A	58	INT_STATUS	R	-	-	-	FIFO_OFLOW_INT	I2C_MST_INT	-	-	DATA_RDY_INT



## MPU-6000/MPU-6050 Register Map and Descriptions

Document Number: RM-MPU-6000A-00  
Revision: 4.2  
Release Date: 08/19/2013

Addr (Hex)	Addr (Dec.)	Register Name	Serial IF	Bit7	Bits	Bits	Bit4	Bits	Bit2	Bit1	Bit0
3B	59	ACCEL_XOUT_H	R					ACCEL_XOUT[15:8]			
3C	60	ACCEL_XOUT_L	R					ACCEL_XOUT[7:0]			
3D	61	ACCEL_YOUT_H	R					ACCEL_YOUT[15:8]			
3E	62	ACCEL_YOUT_L	R					ACCEL_YOUT[7:0]			
3F	63	ACCEL_ZOUT_H	R					ACCEL_ZOUT[15:8]			
40	64	ACCEL_ZOUT_L	R					ACCEL_ZOUT[7:0]			
41	65	TEMP_OUT_H	R					TEMP_OUT[15:8]			
42	66	TEMP_OUT_L	R					TEMP_OUT[7:0]			
43	67	GYRO_XOUT_H	R					GYRO_XOUT[15:8]			
44	68	GYRO_XOUT_L	R					GYRO_XOUT[7:0]			
45	69	GYRO_YOUT_H	R					GYRO_YOUT[15:8]			
46	70	GYRO_YOUT_L	R					GYRO_YOUT[7:0]			
47	71	GYRO_ZOUT_H	R					GYRO_ZOUT[15:8]			
48	72	GYRO_ZOUT_L	R					GYRO_ZOUT[7:0]			
49	73	EXT_SENS_DATA_00	R					EXT_SENS_DATA_00[0:0]			
4A	74	EXT_SENS_DATA_01	R					EXT_SENS_DATA_01[7:0]			
4B	75	EXT_SENS_DATA_02	R					EXT_SENS_DATA_02[7:0]			
4C	76	EXT_SENS_DATA_03	R					EXT_SENS_DATA_03[7:0]			
4D	77	EXT_SENS_DATA_04	R					EXT_SENS_DATA_04[7:0]			
4E	78	EXT_SENS_DATA_05	R					EXT_SENS_DATA_05[7:0]			
4F	79	EXT_SENS_DATA_06	R					EXT_SENS_DATA_06[7:0]			
50	80	EXT_SENS_DATA_07	R					EXT_SENS_DATA_07[7:0]			
51	81	EXT_SENS_DATA_08	R					EXT_SENS_DATA_08[7:0]			
52	82	EXT_SENS_DATA_09	R					EXT_SENS_DATA_09[7:0]			
53	83	EXT_SENS_DATA_10	R					EXT_SENS_DATA_10[7:0]			
54	84	EXT_SENS_DATA_11	R					EXT_SENS_DATA_11[7:0]			
55	85	EXT_SENS_DATA_12	R					EXT_SENS_DATA_12[7:0]			
56	86	EXT_SENS_DATA_13	R					EXT_SENS_DATA_13[7:0]			
57	87	EXT_SENS_DATA_14	R					EXT_SENS_DATA_14[7:0]			
58	88	EXT_SENS_DATA_15	R					EXT_SENS_DATA_15[7:0]			
59	89	EXT_SENS_DATA_16	R					EXT_SENS_DATA_16[7:0]			
5A	90	EXT_SENS_DATA_17	R					EXT_SENS_DATA_17[7:0]			
5B	91	EXT_SENS_DATA_18	R					EXT_SENS_DATA_18[7:0]			
5C	92	EXT_SENS_DATA_19	R					EXT_SENS_DATA_19[7:0]			
5D	93	EXT_SENS_DATA_20	R					EXT_SENS_DATA_20[7:0]			
5E	94	EXT_SENS_DATA_21	R					EXT_SENS_DATA_21[7:0]			
5F	95	EXT_SENS_DATA_22	R					EXT_SENS_DATA_22[7:0]			
60	96	EXT_SENS_DATA_23	R					EXT_SENS_DATA_23[7:0]			
63	99	I2C_SLV0_DO	R/W					I2C_SLV0_DO[7:0]			
64	100	I2C_SLV1_DO	R/W					I2C_SLV1_DO[7:0]			
65	101	I2C_SLV2_DO	R/W					I2C_SLV2_DO[7:0]			
66	102	I2C_SLV3_DO	R/W					I2C_SLV3_DO[7:0]			

# ATmega328P

## Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 4/8/16/32K Bytes of In-System Self-Programmable Flash program memory (ATmega48P/88P/168P/328P)
  - 256/512/1K Bytes EEPROM (ATmega48P/88P/168P/328P)
  - 512/1K/2K Bytes Internal SRAM (ATmega48P/88P/168P/328P)
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C<sup>(1)</sup>
  - Optional Boot Code Section with Independent Lock Bits
  - In-System Programming by On-chip Boot Program
  - True Read-While-Write Operation
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package  
Temperature Measurement
  - 6-channel 10-bit ADC in PDIP Package  
Temperature Measurement
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Byte-oriented 2-wire Serial Interface (Philips I<sup>2</sup>C compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF
- Operating Voltage:
  - 1.8 - 5.5V for ATmega48P/88P/168PV
  - 2.7 - 5.5V for ATmega48P/88P/168P
  - 1.8 - 5.5V for ATmega328P
- Temperature Range:
  - -40°C to 85°C
- Speed Grade:
  - ATmega48P/88P/168PV: 0 - 4 MHz @ 1.8 - 5.5V, 0 - 10 MHz @ 2.7 - 5.5V
  - ATmega48P/88P/168P: 0 - 10 MHz @ 2.7 - 5.5V, 0 - 20 MHz @ 4.5 - 5.5V
  - ATmega328P: 0 - 4 MHz @ 1.8 - 5.5V, 0 - 10 MHz @ 2.7 - 5.5V, 0 - 20 MHz @ 4.5 - 5.5V
- Low Power Consumption at 1 MHz, 1.8V, 25°C for ATmega48P/88P/168P:
  - Active Mode: 0.3 mA
  - Power-down Mode: 0.1 µA
  - Power-save Mode: 0.8 µA (Including 32 kHz RTC)



8-bit **AVR®**  
Microcontroller  
with 4/8/16/32K  
Bytes In-System  
Programmable  
Flash

**ATmega48P/V**  
**ATmega88P/V**  
**ATmega168P/V**  
**ATmega328P**

**Preliminary**  
**Summary**

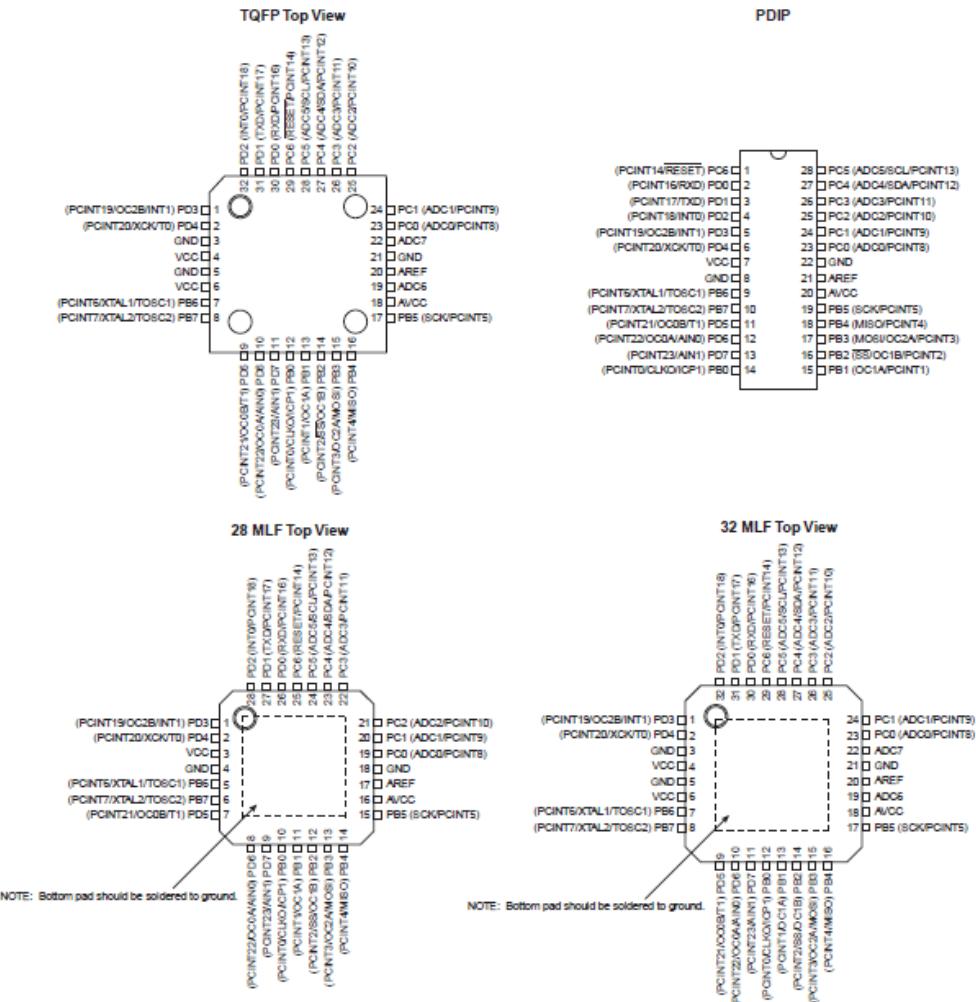
Rev. 8025FS-AVR-08/08





## 1. Pin Configurations

**Figure 1-1.** Pinout ATmega48P/88P/168P/328P



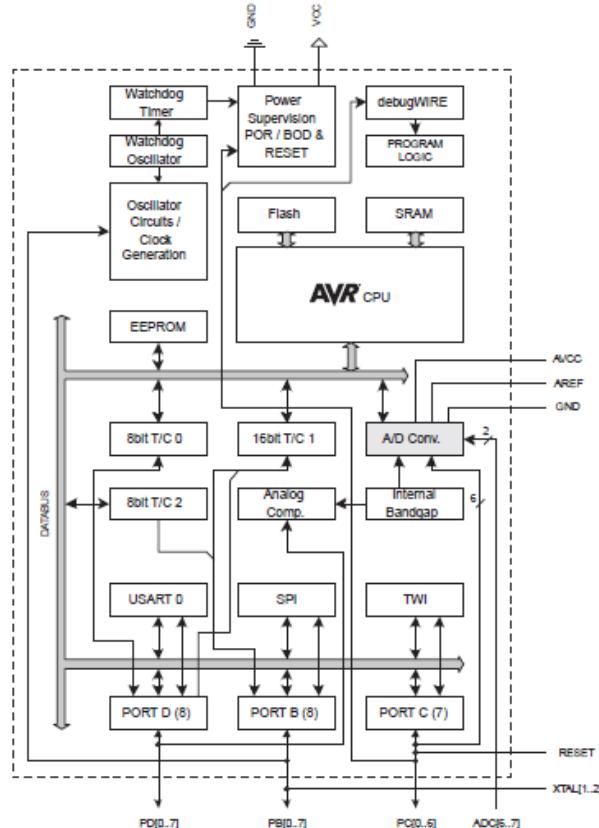
# ATmega48P/88P/168P/328P

## 2. Overview

The ATmega48P/88P/168P/328P is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega48P/88P/168P/328P achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

### 2.1 Block Diagram

Figure 2-1. Block Diagram



The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting

## AT24C04C



### AT24C04C and AT24C08C

I<sup>2</sup>C-Compatible, (2-wire) Serial EEPROM  
4-Kbit (512 x 8), 8-Kbit (1024 x 8)

#### DATASHEET

##### Standard Features

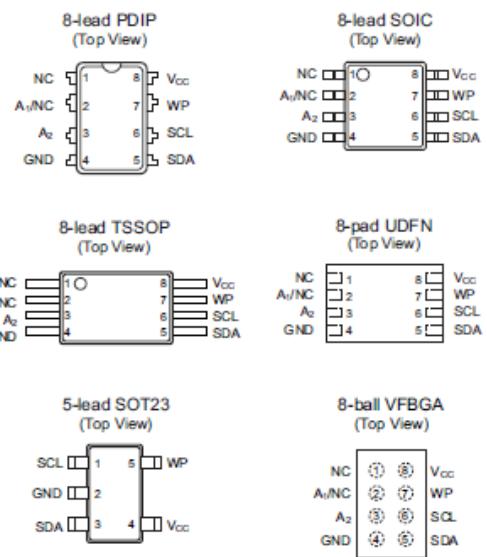
- Low-voltage and Standard-voltage Operation
  - V<sub>CC</sub> = 1.7V to 5.5V
- Internally Organized as 512 x 8 (4K), or 1024 x 8 (8K)
- I<sup>2</sup>C-compatible (2-wire) Serial Interface
- Schmitt Trigger, Filtered Inputs for Noise Suppression
- Bidirectional Data Transfer Protocol
- 1MHz (2.5V, 2.7V, 5V), 400kHz (1.7V) Compatibility
- Write Protect Pin for Hardware Data Protection
- 16-byte Page Write Mode
  - Partial Page Writes Allowed
- Self-timed Write Cycle (5ms Max)
- High-reliability
  - Endurance: 1,000,000 Write Cycles
  - Data Retention: 100 Years
- Green Package Options (Pb/Halide-free/RoHS Compliant)
  - 8-lead PDIP, 8-lead SOIC, 8-lead TSSOP, 8-pad UDFN, 5-lead SOT23, and 8-ball VFBGA
- Die Options: Wafer Form and Tape and Reel

##### Description

The Atmel® AT24C04C and AT24C08C provides 4,096/8,192 bits of Serial Electrically Erasable and Programmable Read-Only Memory (EEPROM) organized as 512/1024 words of 8 bits each. The device is optimized for use in many industrial and commercial applications where low-power and low-voltage operation are essential. AT24C04C/08C is available in space-saving 8-lead PDIP, 8-lead JEDEC SOIC, 8-lead TSSOP, 8-pad UDFN, 5-lead SOT23, and 8-ball VFBGA packages and is accessed via a 2-wire serial interface.

## 1. Pin Configurations and Pinouts

Pin Name	Function
NC	No Connect
A <sub>1</sub>	Address Input (4K Only)
A <sub>2</sub>	Address Input
SDA	Serial Data
SCL	Serial Clock Input
WP	Write Protect
GND	Ground
V <sub>CC</sub>	Power Supply



Notes:

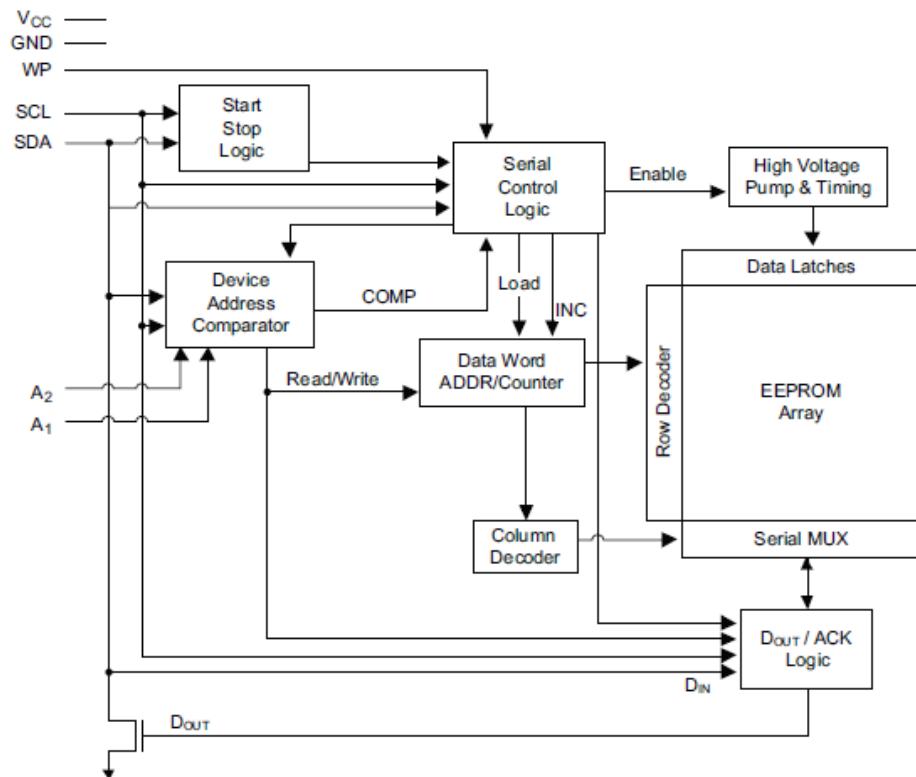
- For use of 5-lead SOT23, the software A<sub>2</sub> and A<sub>1</sub> bits in the device address word must be set to zero to properly communicate.
- Drawings are not to scale.

## 2. Absolute Maximum Ratings

Operating Temperature .....	-55°C to +125°C
Storage Temperature .....	-65°C to +150°C
Voltage on any pin with respect to ground .....	-1.0V to +7.0V
Maximum Operating Voltage .....	6.25V
DC Output Current .....	5.0mA

\*Notice: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

### 3. Block Diagram



## Anexo B

### **Programa de Captura de Datos en lenguaje Wiring de Arduino**

```
#include <SD.h>
#include <Wire.h>

//Direccion I2C de la IMU
#define MPU1 0x68
#define MPU2 0x69

//Ratios de conversion
#define A_R 4096
#define G_R 65.5

//MPU-6050
struct MPU
{
    int16_t Ax;
    int16_t Ay;
    int16_t Az;
    //int16_t Gx;
    int16_t Gy;
    int16_t Gz;
};
struct OFFSET
{
    float Ax;
    float Ay;
    float Az;
    float Gx;
    float Gy;
    float Gz;
};

OFFSET OffsetMuslo;
OFFSET OffsetPierna;

boolean flag = false;

const int button1 = 2;
const int button2 = 3;
const int CS = 4;
const int led1 = 5;
const int led2 = 6;
unsigned int lastInt1, lastInt2;
unsigned long timer1, timer2, timer3;

File DAT;
File CSV;

void setup()
{
    pinMode(button1, INPUT);
    pinMode(button2, INPUT);
    pinMode(CS, OUTPUT);
    pinMode(led1, OUTPUT);
    pinMode(led2, OUTPUT);
    digitalWrite(led1, LOW);
    digitalWrite(led2, LOW);
```

```

Wire.begin();
TWBR = 12;

Wire.beginTransmission(MPU1);
Wire.write(0x6B);
Wire.write(0);
Wire.endTransmission(true);
Wire.beginTransmission(MPU1);
Wire.write(0x1C);
Wire.write(0x10);
Wire.endTransmission(true);
Wire.beginTransmission(MPU1);
Wire.write(0x1B);
Wire.write(0x08);
Wire.endTransmission(true);

delay(10);

Wire.beginTransmission(MPU2);
Wire.write(0x6B);
Wire.write(0);
Wire.endTransmission(true);
Wire.beginTransmission(MPU2);
Wire.write(0x1C);
Wire.write(0x10);
Wire.endTransmission(true);
Wire.beginTransmission(MPU2);
Wire.write(0x1B);
Wire.write(0x08);
Wire.endTransmission(true);

Serial.begin(9600);

if(!SD.begin(4))
{
    Serial.println("No se pudo inicializar");
    return;
}

Serial.println("Inicializacion exitosa");

if(SD.exists("datalog.dat"))
{
    SD.remove("datalog.dat");
}
DAT = SD.open("datalog.dat", FILE_WRITE);

if(SD.exists("datalog.csv"))
{
    SD.remove("datalog.csv");
}
}

void loop()
{
    int buttonState1 = digitalRead(button1);
    if(buttonState1 == HIGH & flag == false)
    {
        if(millis() > lastInt1 + 150)
        {

```

```

//Serial.println("Calibrando Muslo y Pierna...");
digitalWrite(led1,HIGH);

calibracion();

//Serial.println("Calibracion terminada");

lastInt1 = millis();
flag = true;

digitalWrite(led1,LOW);

delay(500);
}

}

if(buttonState1 == HIGH && flag == true)
{
    if(millis() > lastInt1 + 150)
    {
        digitalWrite(led1,HIGH);
        delay(500);
        digitalWrite(led1,LOW);

        funcionWrite();
        lastInt1 = millis();
        flag = false;

        int buttonState3;
        while(1)
        {
            digitalWrite(led1,HIGH);
            delay(500);
            digitalWrite(led1,LOW);
            delay(500);
            buttonState3 = digitalRead(button2);
            if(buttonState3 == HIGH) break;
        }
    }
}

int buttonState2 = digitalRead(button2);
if(buttonState2 == HIGH)
{
    if(millis() > lastInt2 + 150)
    {
        digitalWrite(led2,HIGH);
        delay(500);
        digitalWrite(led2,LOW);

        funcionRead();
        lastInt2 = millis();

        while(1)
        {
            digitalWrite(led2,HIGH);
            delay(500);
            digitalWrite(led2,LOW);
            delay(500);
            buttonState1 = digitalRead(button1);
            if(buttonState1 == HIGH) break;
        }
    }
}

```

```

        }

    }

void calibracion()
{
    MPU Muslo;
    MPU Pierna;
    for(int i = 0; i < 5000; i++)
    {
        Wire.beginTransmission(MPU1);
        Wire.write(0x3B); //Pedir el registro 0x3B - corresponde al AcX
        Wire.endTransmission(false);
        Wire.requestFrom(MPU1,6,true);
        OffsetMuslo.Ax += Wire.read()<<8|Wire.read();
        OffsetMuslo.Ay += Wire.read()<<8|Wire.read();
        OffsetMuslo.Az += Wire.read()<<8|Wire.read();
        Wire.endTransmission(true);

        Wire.beginTransmission(MPU1);
        Wire.write(0x43);
        Wire.endTransmission(false);
        Wire.requestFrom(MPU1,6,true);
        OffsetMuslo.Gx += Wire.read()<<8|Wire.read();
        OffsetMuslo.Gy += Wire.read()<<8|Wire.read();
        OffsetMuslo.Gz += Wire.read()<<8|Wire.read();
        Wire.endTransmission(true);

        Wire.beginTransmission(MPU2);
        Wire.write(0x3B); //Pedir el registro 0x3B - corresponde al AcX
        Wire.endTransmission(false);
        Wire.requestFrom(MPU2,6,true);
        OffsetPierna.Ax += Wire.read()<<8|Wire.read();
        OffsetPierna.Ay += Wire.read()<<8|Wire.read();
        OffsetPierna.Az += Wire.read()<<8|Wire.read();
        Wire.endTransmission(true);

        Wire.beginTransmission(MPU2);
        Wire.write(0x43);
        Wire.endTransmission(false);
        Wire.requestFrom(MPU2,6,true);
        OffsetPierna.Gx += Wire.read()<<8|Wire.read();
        OffsetPierna.Gy += Wire.read()<<8|Wire.read();
        OffsetPierna.Gz += Wire.read()<<8|Wire.read();
        Wire.endTransmission(true);
    }
    OffsetMuslo.Ax /= 5000.0;
    OffsetMuslo.Ay /= 5000.0;
    OffsetMuslo.Az /= 5000.0;
    OffsetMuslo.Gx /= 5000.0;
    OffsetMuslo.Gy /= 5000.0;
    OffsetMuslo.Gz /= 5000.0;

    OffsetPierna.Ax /= 5000.0;
    OffsetPierna.Ay /= 5000.0;
    OffsetPierna.Az /= 5000.0;
    OffsetPierna.Gx /= 5000.0;
    OffsetPierna.Gy /= 5000.0;
    OffsetPierna.Gz /= 5000.0;

    Muslo.Ax = (int16_t)OffsetMuslo.Ax;
    Muslo.Ay = (int16_t)OffsetMuslo.Ay;
}

```

```

Muslo.Az = (int16_t)OffsetMuslo.Az;
//Muslo.Gx = (int16_t)OffsetMuslo.Gx;
Muslo.Gy = (int16_t)OffsetMuslo.Gy;
Muslo.Gz = (int16_t)OffsetMuslo.Gz;

Pierna.Ax = (int16_t)OffsetPierna.Ax;
Pierna.Ay = (int16_t)OffsetPierna.Ay;
Pierna.Az = (int16_t)OffsetPierna.Az;
//Pierna.Gx = (int16_t)OffsetPierna.Gx;
Pierna.Gy = (int16_t)OffsetPierna.Gy;
Pierna.Gz = (int16_t)OffsetPierna.Gz;

DAT.write((const uint8_t*)&Muslo, sizeof(Muslo));
DAT.write((const uint8_t*)&Pierna, sizeof(Pierna));
}

void funcionWrite()
{
    timer1 = 0;
    timer2 = 0;

    MPU Muslo;
    MPU Pierna;

    timer3 = millis();

    for(int j = 0 ;j < 1000; j++)
    {
        timer1 = micros();

        Wire.beginTransmission(MPU1);
        Wire.write(0x3B); //Pedir el registro 0x3B - corresponde al AcX
        Wire.endTransmission(false);
        Wire.requestFrom(MPU1,6,true);
        Muslo.Ax = Wire.read()<<8|Wire.read();
        Muslo.Ax -= (int16_t)OffsetMuslo.Ax; Muslo.Ax -= A_R;
        Muslo.Ay = Wire.read()<<8|Wire.read();
        Muslo.Ay -= (int16_t)OffsetMuslo.Ay;
        Muslo.Az = Wire.read()<<8|Wire.read();
        Muslo.Az -= (int16_t)OffsetMuslo.Az;
        Wire.endTransmission(true);

        Wire.beginTransmission(MPU1);
        Wire.write(0x45);
        Wire.endTransmission(false);
        Wire.requestFrom(MPU1,6,true);
        //Muslo.Gx = Wire.read()<<8|Wire.read();
        Muslo.Gx -= (int16_t)OffsetMuslo.Gx;
        Muslo.Gy = Wire.read()<<8|Wire.read();
        Muslo.Gy -= (int16_t)OffsetMuslo.Gy;
        Muslo.Gz = Wire.read()<<8|Wire.read();
        Muslo.Gz -= (int16_t)OffsetMuslo.Gz;
        Wire.endTransmission(true);

        Wire.beginTransmission(MPU2);
        Wire.write(0x3B); //Pedir el registro 0x3B - corresponde al AcX
        Wire.endTransmission(false);
        Wire.requestFrom(MPU2,6,true);
        Pierna.Ax = Wire.read()<<8|Wire.read();
        Pierna.Ax -= (int16_t)OffsetPierna.Ax; Pierna.Ax -= A_R;
        Pierna.Ay = Wire.read()<<8|Wire.read();

```

```

Pierna.Ay -= (int16_t)OffsetPierna.Ay;
Pierna.Az = Wire.read()<<8|Wire.read();
Pierna.Az -= (int16_t)OffsetPierna.Az;
Wire.endTransmission(true);

Wire.beginTransmission(MPU2);
Wire.write(0x45);
Wire.endTransmission(false);
Wire.requestFrom(MPU2, 6, true);
//Pierna.Gx = Wire.read()<<8|Wire.read();
//Pierna.Gx -= (int16_t)OffsetPierna.Gx;
Pierna.Gy = Wire.read()<<8|Wire.read();
Pierna.Gy -= (int16_t)OffsetPierna.Gy;
Pierna.Gz = Wire.read()<<8|Wire.read();
Pierna.Gz -= (int16_t)OffsetPierna.Gz;
Wire.endTransmission(true);

DAT.write((const uint8_t*)&Muslo, sizeof(Muslo));
DAT.write((const uint8_t*)&Pierna, sizeof(Pierna));

while(micros() - timer1 < 36000)
{
}

if(micros() - timer1 > 37000)
{
//Serial.print("Error de Downsampling en la muestra numero: ");
//Serial.println(j);
//Serial.println(micros() - timer1);
}
}

DAT.close();
Serial.println("Fin");
}

void funcionRead()
{
DAT = SD.open("datalog.dat", FILE_READ); //abrimos el archivo
CSV = SD.open("datalog.csv", FILE_WRITE); //abrimos el archivo

while (DAT.available())
{
    MPU Muslo;
    MPU Pierna;

    DAT.read((const uint8_t*)&Muslo, sizeof(Muslo));
    DAT.read((const uint8_t*)&Pierna, sizeof(Pierna));

    CSV.print(Muslo.Ax);
    CSV.print(",");
    CSV.print(Muslo.Ay);
    CSV.print(",");
    CSV.print(Muslo.Az);
    CSV.print(",");
    //CSV.print(Muslo.Gx);
    //CSV.print(",");
    CSV.print(Muslo.Gy);
    CSV.print(",");
    CSV.print(Muslo.Gz);
    CSV.print(",");
}
}

```

```
    CSV.print(Pierna.Ax);
    CSV.print(",");
    CSV.print(Pierna.Ay);
    CSV.print(",");
    CSV.print(Pierna.Az);
    CSV.print(",");
    //CSV.print(Pierna.Gx);
    //CSV.print(",");
    CSV.print(Pierna.Gy);
    CSV.print(",");
    CSV.println(Pierna.Gz);

}

DAT.close();
CSV.close();
}
```

## **Programa de la GUI en lenguaje MATLAB™**

```
function varargout = test_gui(varargin)
% TEST_GUI MATLAB code for test_gui.fig
%   TEST_GUI, by itself, creates a new TEST_GUI or raises the existing
%   singleton*.
%
%   H = TEST_GUI returns the handle to a new TEST_GUI or the handle
%   to
%   the existing singleton*.
%
%   TEST_GUI('CALLBACK', hObject, eventData, handles,...) calls the
%   local
%   function named CALLBACK in TEST_GUI.M with the given input ar-
%   guments.
%
%   TEST_GUI('Property','Value',...) creates a new TEST_GUI or rai-
%   ses the
%   existing singleton*. Starting from the left, property value
%   pairs are
%   applied to the GUI before test_gui_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property ap-
%   plication
%   stop. All inputs are passed to test_gui_OpeningFcn via varar-
%   gin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows
only one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help test_gui

% Last Modified by GUIDE v2.5 07-Nov-2017 19:00:42

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',          mfilename, ...
                   'gui_Singleton',    gui_Singleton, ...
                   'gui_OpeningFcn',   @test_gui_OpeningFcn, ...
                   'gui_OutputFcn',    @test_gui_OutputFcn, ...
                   'gui_LayoutFcn',    [], ...
                   'gui_Callback',     []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before test_gui is made visible.
function test_gui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
```

```

% hObject    handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to test_gui (see VARARGIN)

% Choose default command line output for test_gui
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% This sets up the initial plot - only do when we are invisible
% so window can get raised using test_gui.
if strcmp(get(hObject,'Visible'),'off')

end

% UIWAIT makes test_gui wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = test_gui_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
axes(handles.axes1);
cla;

M = csvread('F:\DATALOG.csv',1,0);

A = zeros(1000,6);

a = 1;
%b = [1/10 1/10 1/10 1/10 1/10 1/10 1/10 1/10 1/10];
%b = [1/8 1/8 1/8 1/8 1/8 1/8 1/8 1/8];
b = [1/5 1/5 1/5 1/5 1/5];
A(:,1) = filter(b,a,M(:,1));
A(:,2) = filter(b,a,M(:,2));
A(:,3) = filter(b,a,M(:,3));
A(:,4) = filter(b,a,M(:,6));
A(:,5) = filter(b,a,M(:,7));
A(:,6) = filter(b,a,M(:,8));

roll_Muslo_Ac = zeros(1000,1);
pitch_Muslo_Ac = zeros(1000,1);
roll_Pierna_Ac = zeros(1000,1);
pitch_Pierna_Ac = zeros(1000,1);
roll_Rodilla_Ac = zeros(1000,1);

```

```

roll_Muslo_Gy = zeros(1000,1);
pitch_Muslo_Gy = zeros(1000,1);
roll_Pierna_Gy = zeros(1000,1);
pitch_Pierna_Gy = zeros(1000,1);
roll_Rodilla_Gy = zeros(1000,1);

roll_Muslo = zeros(1000,1);
pitch_Muslo = zeros(1000,1);
roll_Pierna = zeros(1000,1);
pitch_Pierna = zeros(1000,1);
roll_Rodilla = zeros(1000,1);

for i = 1:1000

    roll_Muslo_Ac(i) = atan2((A(i,2)/4096),(-
(A(i,1))/4096))*57.295779;
    pitch_Muslo_Ac(i) = atan2(A(i,3)/4096,(sqrt((A(i,2)/4096)^2 +
(-(A(i,1)/4096))^2)))*57.295779;
    roll_Pierna_Ac(i) = atan2((A(i,5)/4096),(-
(A(i,4))/4096))*57.295779;
    pitch_Pierna_Ac(i) = atan2(A(i,6)/4096,(sqrt((A(i,5)/4096)^2 +
(-(A(i,4)/4096))^2)))*57.295779;
end

roll_Rodilla_Ac(:) = roll_Muslo_Ac(:) - roll_Pierna_Ac(:);

for i = 1:1000
    if i > 1
        roll_Muslo_Gy(i) = (M(i,5)/65.5)*0.036 + roll_Muslo_Gy(i-1);
        pitch_Muslo_Gy(i) = (M(i,4)/65.5)*0.036 + pitch_Muslo_Gy(i-1);
        roll_Pierna_Gy(i) = (M(i,10)/65.5)*0.036 + roll_Pierna_Gy(i-
1);
        pitch_Pierna_Gy(i) = (M(i,9)/65.5)*0.036 + pitch_Pierna_Gy(i-
1);
    else
        roll_Muslo_Gy(i) = (M(i,5)/65.5)*0.036;
        pitch_Muslo_Gy(i) = (M(i,4)/65.5)*0.036;
        roll_Pierna_Gy(i) = (M(i,10)/65.5)*0.036;
        pitch_Pierna_Gy(i) = (M(i,9)/65.5)*0.036;
    end
end

roll_Rodilla_Gy(:) = roll_Muslo_Gy(:) - roll_Pierna_Gy(:);

for i = 1:1000
    if i > 1
        roll_Muslo(i) = 0.98*((M(i,5)/65.5)*0.036 + roll_Muslo(i-1)) +
0.02*roll_Muslo_Ac(i);
        pitch_Muslo(i) = 0.98*((M(i,4)/65.5)*0.036 + pitch_Muslo(i-1)) +
0.02*pitch_Muslo_Ac(i);
        roll_Pierna(i) = 0.98*((M(i,10)/65.5)*0.036 + roll_Pierna(i-
1)) + 0.02*roll_Pierna_Ac(i);
        pitch_Pierna(i) = 0.98*((M(i,9)/65.5)*0.036 + pitch_Pierna(i-
1)) + 0.02*pitch_Pierna_Ac(i);
    else
        roll_Muslo(i) = 0.98*((M(i,5)/65.5)*0.036) +
0.02*roll_Muslo_Ac(i);
    end
end

```

```

        pitch_Muslo(i) = 0.98*((M(i,4)/65.5)*0.036) +
0.02*pitch_Muslo_Ac(i);
        roll_Pierna(i) = 0.98*((M(i,10)/65.5)*0.036) +
0.02*roll_Pierna_Ac(i);
        pitch_Pierna(i) = 0.98*((M(i,9)/65.5)*0.036) +
0.02*pitch_Pierna_Ac(i);
    end
end

roll_Rodilla(:) = roll_Muslo(:) - roll_Pierna(:);

popup_sel_index = get(handles.popupmenu1, 'Value');
switch popup_sel_index
    case 1
        plot(roll_Muslo_Ac(:), 'Linewidth', 2, 'Color', 'b');
        hold on
        plot(pitch_Muslo_Ac(:), 'Linewidth', 2, 'Color', 'r');
        axis([0 1000 -100 100]);
        xlabel('muestras');
        ylabel('grados');
        hold off
        grid on
    case 2
        plot(roll_Pierna_Ac(:), 'Linewidth', 2, 'Color', 'b');
        hold on
        plot(pitch_Pierna_Ac(:), 'Linewidth', 2, 'Color', 'r');
        axis([0 1000 -100 100]);
        xlabel('muestras');
        ylabel('grados');
        hold off
        grid on
    case 3
        plot(roll_Rodilla_Ac(:), 'Linewidth', 2, 'Color', 'b');
        axis([0 1000 -100 100]);
        xlabel('muestras');
        ylabel('grados');
        grid on
    case 4
        plot(roll_Muslo_Gy(:), 'Linewidth', 2, 'Color', 'b');
        hold on
        plot(pitch_Muslo_Gy(:), 'Linewidth', 2, 'Color', 'r');
        axis([0 1000 -100 100]);
        xlabel('muestras');
        ylabel('grados');
        hold off
        grid on
    case 5
        plot(roll_Pierna_Gy(:), 'Linewidth', 2, 'Color', 'b');
        hold on
        plot(pitch_Pierna_Gy(:), 'Linewidth', 2, 'Color', 'r');
        axis([0 1000 -100 100]);
        xlabel('muestras');
        ylabel('grados');
        hold off
        grid on
    case 6
        plot(roll_Rodilla_Gy(:), 'Linewidth', 2, 'Color', 'b');
        axis([0 1000 -100 100]);
        xlabel('muestras');
        ylabel('grados');
        grid on

```

```

case 7
plot(roll_Muslo(:), 'Linewidth', 2, 'Color', 'b');
hold on
plot(pitch_Muslo(:), 'Linewidth', 2, 'Color', 'r');
axis([0 1000 -100 100]);
xlabel('muestras');
ylabel('grados');
hold off
grid on
case 8
plot(roll_Pierna(:), 'Linewidth', 2, 'Color', 'b');
hold on
plot(pitch_Pierna(:), 'Linewidth', 2, 'Color', 'r');
axis([0 1000 -100 100]);
xlabel('muestras');
ylabel('grados');
hold off
grid on
case 9
plot(roll_Rodilla(:), 'Linewidth', 2, 'Color', 'b');
axis([0 1000 -100 100]);
xlabel('muestras');
ylabel('grados');
grid on
case 10
plot(M(:,1));
hold on
plot(A(:,1), 'Linewidth', 1.5);
xlabel('ms');
ylabel('AcX(Muslo)');
hold off
grid on
case 11
plot(M(:,2));
hold on
plot(A(:,2), 'Linewidth', 1.5);
xlabel('ms');
ylabel('AcY(Muslo)');
hold off
grid on
case 12
plot(M(:,3));
hold on
plot(A(:,3), 'Linewidth', 1.5);
xlabel('ms');
ylabel('AcZ(Muslo)');
hold off
grid on
case 13
plot(M(:,4));
xlabel('ms');
ylabel('GyY(Muslo)');
grid on
case 14
plot(M(:,5));
xlabel('ms');
ylabel('GyZ(Muslo)');
grid on
case 15
plot(M(:,6));
hold on

```

```

plot(A(:,4), 'Linewidth',1.5);
xlabel('ms');
ylabel('AcX(Pierna)');
hold off
grid on
case 16
plot(M(:,7));
hold on
plot(A(:,5), 'Linewidth',1.5);
xlabel('ms');
ylabel('AcY(Pierna)');
hold off
grid on
case 17
plot(M(:,8));
hold on
plot(A(:,6), 'Linewidth',1.5);
xlabel('ms');
ylabel('AcZ(Pierna)');
hold off
grid on
case 18
plot(M(:,9));
xlabel('ms');
ylabel('GyY(Pierna)');
grid on
case 19
plot(M(:,10));
xlabel('ms');
ylabel('GyZ(Pierna)');
grid on
case 20
plot(roll_Muslo(:, 'Linewidth',2,'Color','r');
hold on
plot(roll_Pierna(:, 'Linewidth',2,'Color','b');
hold on
plot(roll_Rodilla(:, 'Linewidth',2,'Color','g');
axis([0 1000 -100 100]);
xlabel('muestras');
ylabel('grados');
hold off
grid on
end

% -----
function FileMenu_Callback(hObject, eventdata, handles)
% hObject    handle to FileMenu (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% -----
function OpenMenuItem_Callback(hObject, eventdata, handles)
% hObject    handle to OpenMenuItem (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
file = uigetfile('*.fig');
if ~isequal(file, 0)
    open(file);
end

```

```

% -----
function PrintMenuItem_Callback(hObject, eventdata, handles)
% hObject    handle to PrintMenuItem (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
printdlg(handles.figure1)

% -----
function CloseMenuItem_Callback(hObject, eventdata, handles)
% hObject    handle to CloseMenuItem (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
selection = questdlg(['Close ' get(handles.figure1,'Name') '?'],...
                     ['Close ' get(handles.figure1,'Name') '...'],...
                     'Yes','No','Yes');
if strcmp(selection,'No')
    return;
end

delete(handles.figure1)

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu1 contents
% as cell array
%         contents{get(hObject,'Value')} returns selected item from po-
% pupmenu1

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
% called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaul-
tUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

set(hObject, 'String', {'Acelerometro(Muslo): Pitch(rojo) &
Roll(azul)', 'Acelerometro(Pierna): Pitch(rojo) & Roll(azul)', 'Acelero-
metro(Rodilla): Roll(azul)', 'Giroscopo(Muslo): Pitch(rojo) &
Roll(azul)', 'Giroscopo(Pierna): Pitch(rojo) & Roll(azul)', 'Giros-
copo(Rodilla): Roll(azul)', 'Acelerometro & Giroscopo(Muslo):
Pitch(rojo) & Roll(azul)', 'Acelerometro & Giroscopo(Pierna):
Pitch(rojo) & Roll(azul)', 'Acelerometro & Giroscopo(Rodilla):
Roll(azul)', 'AcX(Muslo)', 'AcY(Muslo)', 'AcZ(Muslo)', 'GyY(Muslo)', 'GyZ(M
uslo)', 'AcX(Pierna)', 'AcY(Pierna)', 'AcZ(Pierna)', 'GyY(Pierna)', 'GyZ(Pi
erna)'})

```

```
erna)', 'Roll Filtro Complementario: Muslo(rojo) & Pierna(azul) & Rodilla(verde)' });

% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
%             called

% Hint: place code in OpeningFcn to populate axes1
```

**Programa de Obtención de la Curva Cinemática de Rodilla en lenguaje  
MATLAB™**

```
M = csvread('F:\DATACONFIG.csv',1,0);

A = zeros(1000,6);

a = 1;
%b = [1/10 1/10 1/10 1/10 1/10 1/10 1/10 1/10 1/10 1/10];
%b = [1/8 1/8 1/8 1/8 1/8 1/8 1/8 1/8 1/8];
b = [1/5 1/5 1/5 1/5 1/5];
A(:,1) = filter(b,a,M(:,1));
A(:,2) = filter(b,a,M(:,2));
A(:,3) = filter(b,a,M(:,3));
A(:,4) = filter(b,a,M(:,6));
A(:,5) = filter(b,a,M(:,7));
A(:,6) = filter(b,a,M(:,8));

roll_Muslo_Ac = zeros(1000,1);
pitch_Muslo_Ac = zeros(1000,1);
roll_Pierna_Ac = zeros(1000,1);
pitch_Pierna_Ac = zeros(1000,1);
roll_Rodilla_Ac = zeros(1000,1);

roll_Muslo_Gy = zeros(1000,1);
pitch_Muslo_Gy = zeros(1000,1);
roll_Pierna_Gy = zeros(1000,1);
pitch_Pierna_Gy = zeros(1000,1);
roll_Rodilla_Gy = zeros(1000,1);

roll_Muslo = zeros(1000,1);
pitch_Muslo = zeros(1000,1);
roll_Pierna = zeros(1000,1);
pitch_Pierna = zeros(1000,1);
roll_Rodilla = zeros(1000,1);

for i = 1:1000

    roll_Muslo_Ac(i) = atan2((A(i,2)/4096),(-
(A(i,1))/4096))*57.295779;
    pitch_Muslo_Ac(i) = atan2(A(i,3)/4096,(sqrt((A(i,2)/4096)^2 +
(-(A(i,1)/4096))^2)))*57.295779;
    roll_Pierna_Ac(i) = atan2((A(i,5)/4096),(-
(A(i,4))/4096))*57.295779;
    pitch_Pierna_Ac(i) = atan2(A(i,6)/4096,(sqrt((A(i,5)/4096)^2 +
(-(A(i,4)/4096))^2)))*57.295779;
end

roll_Rodilla_Ac(:) = roll_Muslo_Ac(:) - roll_Pierna_Ac(:);

for i = 1:1000
    if i > 1
        roll_Muslo_Gy(i) = (M(i,5)/65.5)*0.036 + roll_Muslo_Gy(i-1);
        pitch_Muslo_Gy(i) = (M(i,4)/65.5)*0.036 + pitch_Muslo_Gy(i-1);
        roll_Pierna_Gy(i) = (M(i,10)/65.5)*0.036 + roll_Pierna_Gy(i-
1);
    end
end
```

```

        pitch_Pierna_Gy(i) = (M(i,9)/65.5)*0.036 + pitch_Pierna_Gy(i-
1);
    else
        roll_Muslo_Gy(i) = (M(i,5)/65.5)*0.036;
        pitch_Muslo_Gy(i) = (M(i,4)/65.5)*0.036;
        roll_Pierna_Gy(i) = (M(i,10)/65.5)*0.036;
        pitch_Pierna_Gy(i) = (M(i,9)/65.5)*0.036;
    end
end

roll_Rodilla_Gy(:) = roll_Muslo_Gy(:) - roll_Pierna_Gy(:);

for i = 1:1000
    if i > 1
        roll_Muslo(i) = 0.98*((M(i,5)/65.5)*0.036 + roll_Muslo(i-1)) +
0.02*roll_Muslo_Ac(i);
        pitch_Muslo(i) = 0.98*((M(i,4)/65.5)*0.036 + pitch_Muslo(i-1)) +
0.02*pitch_Muslo_Ac(i);
        roll_Pierna(i) = 0.98*((M(i,10)/65.5)*0.036 + roll_Pierna(i-
1)) + 0.02*roll_Pierna_Ac(i);
        pitch_Pierna(i) = 0.98*((M(i,9)/65.5)*0.036 + pitch_Pierna(i-
1)) + 0.02*pitch_Pierna_Ac(i);
    else
        roll_Muslo(i) = 0.98*((M(i,5)/65.5)*0.036) +
0.02*roll_Muslo_Ac(i);
        pitch_Muslo(i) = 0.98*((M(i,4)/65.5)*0.036) +
0.02*pitch_Muslo_Ac(i);
        roll_Pierna(i) = 0.98*((M(i,10)/65.5)*0.036) +
0.02*roll_Pierna_Ac(i);
        pitch_Pierna(i) = 0.98*((M(i,9)/65.5)*0.036) +
0.02*pitch_Pierna_Ac(i);
    end
end

roll_Rodilla(:) = roll_Muslo(:) - roll_Pierna(:);

[pks,locs] = findpeaks(-roll_Rodilla,'MinPeakHeight',5,'MinPeakDistan-
ce',25);

S1 = -roll_Rodilla(locs(1):locs(2));
S2 = -roll_Rodilla(locs(2):locs(3));
S3 = -roll_Rodilla(locs(3):locs(4));
S4 = -roll_Rodilla(locs(4):locs(5));
S5 = -roll_Rodilla(locs(5):locs(6));
S6 = -roll_Rodilla(locs(6):locs(7));
S7 = -roll_Rodilla(locs(7):locs(8));
S8 = -roll_Rodilla(locs(8):locs(9));

C1 = vertcat(-roll_Rodilla(locs(1):locs(2)),zeros(length(S7)-len-
gth(S1),1));
C2 = vertcat(-roll_Rodilla(locs(2):locs(3)),zeros(length(S7)-len-
gth(S2),1));
C3 = vertcat(-roll_Rodilla(locs(3):locs(4)),zeros(length(S7)-len-
gth(S3),1));
C4 = vertcat(-roll_Rodilla(locs(4):locs(5)),zeros(length(S7)-len-
gth(S4),1));
C5 = vertcat(-roll_Rodilla(locs(5):locs(6)),zeros(length(S7)-len-
gth(S5),1));
C6 = vertcat(-roll_Rodilla(locs(6):locs(7)),zeros(length(S7)-len-
gth(S6),1));

```

```

C7 = vertcat(-roll_Rodilla(locs(7):locs(8)),zeros(length(S7)-length(S7),1));
C8 = vertcat(-roll_Rodilla(locs(8):locs(9)),zeros(length(S7)-length(S8),1));

% S1 = -roll_Rodilla(579:614);
% S2 = -roll_Rodilla(614:648);
% S3 = -roll_Rodilla(648:682);
% S4 = -roll_Rodilla(682:711);
% S5 = -roll_Rodilla(711:743);
% S6 = -roll_Rodilla(743:776);
% S7 = -roll_Rodilla(776:813);
% S8 = -roll_Rodilla(813:847);

% C1 = vertcat(-roll_Rodilla(579:614),zeros(length(S7)-length(S1),1));
% C2 = vertcat(-roll_Rodilla(614:648),zeros(length(S7)-length(S2),1));
% C3 = vertcat(-roll_Rodilla(648:682),zeros(length(S7)-length(S3),1));
% C4 = vertcat(-roll_Rodilla(682:711),zeros(length(S7)-length(S4),1));
% C5 = vertcat(-roll_Rodilla(711:743),zeros(length(S7)-length(S5),1));
% C6 = vertcat(-roll_Rodilla(743:776),zeros(length(S7)-length(S6),1));
% C7 = vertcat(-roll_Rodilla(776:813),zeros(length(S7)-length(S7),1));
% C8 = vertcat(-roll_Rodilla(813:847),zeros(length(S7)-length(S8),1));

clasificacion = [C1,C2,C3,C4,C5,C6,C7,C8];
figure('Name','Clasificacion','NumberTitle','off')
plot((0:length(S7)-1)*36,clasificacion);
title('Clasificacion');
xlabel('Tiempo[ms]');
ylabel('Angulo de Rodilla[grados]');

t = 0:0.001:1;

I1 = interp1(((0:length(S1)-1)/length(S1)),S1,t,'pchip');
I2 = interp1(((0:length(S2)-1)/length(S2)),S2,t,'pchip');
I3 = interp1(((0:length(S3)-1)/length(S3)),S3,t,'pchip');
I4 = interp1(((0:length(S4)-1)/length(S4)),S4,t,'pchip');
I5 = interp1(((0:length(S5)-1)/length(S5)),S5,t,'pchip');
I6 = interp1(((0:length(S6)-1)/length(S6)),S6,t,'pchip');
I7 = interp1(((0:length(S7)-1)/length(S7)),S7,t,'pchip');
I8 = interp1(((0:length(S8)-1)/length(S8)),S8,t,'pchip');

normalizacion = vertcat(I1,I2,I3,I4,I5,I6,I7,I8);
figure('Name','Normalizacion','NumberTitle','off')
plot(t*100,normalizacion);
title('Normalizacion');
xlabel('Porcentaje de Marcha[%]');
ylabel('Angulo de Rodilla[grados]');

promedio = (I1 + I2 + I3 + I4 + I5 + I6 + I7 + I8)/8;
p = polyfit(t,promedio,7);
ajuste = polyval(p,t);
figure('Name','Ajuste','NumberTitle','off')
plot(t*100,normalizacion);
hold on
plot(t*100,ajuste,'x','MarkerSize',10,'Color','b');
hold on
plot(t*100,promedio,'LineWidth',2,'Color','black');
title('Ajuste');
xlabel('Porcentaje de Marcha[%]');
ylabel('Angulo de Rodilla[grados]');

```