

**UNIVERSIDAD POLITÉCNICA SALESIANA  
SEDE CUENCA**

**CARRERA DE INGENIERÍA ELECTRÓNICA**

**Tesis previa a la obtención del título de:**

**INGENIERO ELECTRÓNICO**

**TEMA:**

**DISEÑO, CONSTRUCCIÓN E IMPLEMENTACIÓN DE  
UN SISTEMA DE CAPTURA DE MOVIMIENTO PARA  
ANÁLISIS ERGONÓMICO DE RIESGO LABORAL DE  
EXTREMIDADES SUPERIORES.**

**AUTOR:**

**Juan Diego Bernal Iñiguez.**

**DIRECTOR:**

**Ing. Patricio Fernando Urgiles Ortiz.**

**Cuenca, Noviembre de 2014**

## CERTIFICACIÓN

En la facultad de Director del Proyecto de Tesis titulado “DISEÑO, CONSTRUCCIÓN E IMPLEMENTACIÓN DE UN SISTEMA DE CAPTURA DE MOVIMIENTO PARA ANÁLISIS ERGONÓMICO DE RIESGO LABORAL DE EXTREMIDADES SUPERIORES” desarrollado por el señor Juan Diego Bernal Iñiguez, certifico la aprobación del presente trabajo de tesis, una vez ejecutado la supervisión y revisión de su contenido.

Cuenca, Noviembre de 2014.

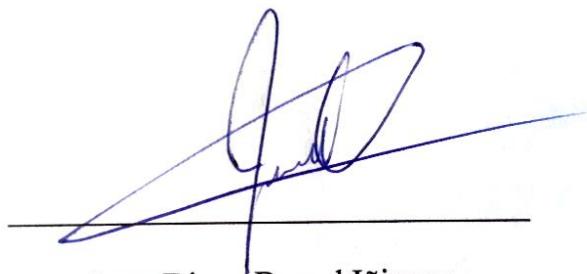


Ing. Patricio Fernando Urgiles Ortiz

## **RESPONSABILIDAD Y AUTORÍA**

El autor del Proyecto de Tesis titulado “DISEÑO, CONSTRUCCIÓN E IMPLEMENTACIÓN DE UN SISTEMA DE CAPTURA DE MOVIMIENTO PARA ANÁLISIS ERGONÓMICO DE RIESGO LABORAL DE EXTREMIDADES SUPERIORES”, Juan Diego Bernal Iñiguez, en virtud de los fundamentos teóricos y sus resultados, declaran de exclusiva responsabilidad y otorgan a la UNIVERSIDAD POLITÉCNICA SALESIANA la divulgación de este documento únicamente para propósitos académicos investigativos.

Cuenca, Noviembre de 2014.



Juan Diego Bernal Iñiguez.  
C.I.0104952833

# ÍNDICE

<b>PRÓLOGO</b>	<b>6</b>
<b>INTRODUCCIÓN</b>	<b>7</b>
<b>CAPÍTULO 1: MARCO TEÓRICO</b>	<b>8</b>
1.1 ERGONOMÍA	8
1.1.1 RIESGO LABORAL	
1.1.2 RIESGO ERGONÓMICO LABORAL	8
1.1.3 ANÁLISIS ERGONÓMICO DE PUESTO DE TRABAJO	9
1.1.4 ERGONOMÍA AMBIENTAL	9
1.1.5 ERGONOMÍA COGNITIVA	9
1.1.6 ERGONOMÍA ORGANIZATIVA	10
1.1.7 ERGONOMÍA GEOMÉTRICA	10
1.1.7.1 PROCESO DE ANÁLISIS ERGONÓMICO GEOMÉTRICO	10
1.1.7.2 MÉTODOS SISTEMÁTICOS DE ANÁLISIS ERGONÓMICO DE RIESGO DE UNA TAREA	12
1.1.7.3 MÉTODO RULA	12
1.1.7.4 PROCEDIMIENTO DE ANÁLISIS	14
1.1.7.5 EVALUACIÓN GLOBAL	15
1.2 SENsoRES INERcIALES	15
1.2.1 ACELERóMETRO	17
1.2.2 GIROSCOPIO	19
1.2.3 MAGNETóMETRO	20
<b>CAPÍTULO 2: CONSTRUCCIÓN DE HARDWARE</b>	<b>23</b>
2.1 DISEÑO DEL MÓDULO DETECTOR DE MOVIMIENTO	23
2.2 INTERCONEXIÓN DE PARTES	26
2.3 DISEÑO Y CONSTRUCCIÓN DE TARJETA ELECTRÓNICA	27
2.3.1 BATERÍA Y SISTEMA DE CARGA	28
2.3.2 DISEÑO DE CIRCUITO CARGADOR DE BATERÍA NI-MH	29
2.4 DISEÑO DE PREnda PARA USUARIO	30
<b>CAPItULO 3: DISEÑO DEL SOFTWARE</b>	<b>32</b>
3.1 DISEÑO DEL SOFTWARE DEL MICROCONTROLADOR	33
3.1.1 CÓDIGO DE PROGRAMA	34
3.1.2 DECLARACIÓN DE VARIABLES	34
3.1.3 SUBPROCESOS	35
3.1.4 INTERRUPCIÓN	39
3.1.5 PROCESO PRINCIPAL	39
3.2 DISEÑO DEL SOFTWARE CARGADOR DE BATERÍA	42

3.3 DISEÑO DEL SOFTWARE DEL PC	46
3.3.1 RECEPCIÓN DE DATOS	47
3.3.2 PROCESAMIENTO DE DATOS RECIBIDOS	50
3.3.2.1 FILTRADO DE DATOS RECIBIDOS	50
3.3.2.2 CÁLCULO DE ÁNGULO DE GIRO A PARTIR DE GIROSCOPIO	53
3.3.2.3 CÁLCULO DE ÁNGULO DE GIRO A PARTIR DE ACELERÓMETRO	55
3.3.2.4 CÁLCULO DE COMPENSACIÓN DE INCLINACIÓN	57
3.3.2.5 CÁLCULO DE ROLL EN EL SISTEMA COORDENADO DE NAVEGACIÓN	59
3.3.2.6 CÁLCULO DE YAW EN EL SISTEMA COORDENADO DE NAVEGACIÓN	63
3.3.2.7 CORRECCIÓN DE PITCH Y RILL A TRAVÉS DE DATOS DEL ACELERÓMETRO	64
3.3.2.8 CORRECCIÓN DE YAW A TRAVÉS DE DATOS DEL MAGNETÓMETRO	65
3.3.2.9 ACONDICIONAMIENTO DE SEÑALES PARA REALIDAD AUMENTADA	66
3.3.3 DISEÑO DE INTERFAZ DE USUARIO	68
<b>CAPÍTULO 4: ANÁLISIS DE RESULTADOS OBTENIDOS</b>	<b>76</b>
4.1 CÁLCULO DEL ERROR	76
4.2 PRUEBAS COMPARATIVAS MODELO 3D VS POSICIÓN REAL	86
4.2.1 PRUEBA #1	86
4.2.2 PRUEBA #2	87
4.2.3 PRUEBA #3	88
4.2.4 PRUEBA #4	89
4.2.5 PRUEBA #5	90
4.2.6 PRUEBA #6	91
4.2.7 FALLAS DEL SISTEMA	92
4.2.7.1 RETARDO DE REPRESENTACIÓN 3D	92
4.2.7.2 ÁNGULOS RECTOS	92
4.2.7.3 ALGORITMOS INCOMPLETOS	92
4.2.7.4 SENsoRES INERciaLES Poco CONFIABLES	93
4.2.7.5 PéRDIDA DE DATOS EN TRANSMISIÓN HASTA PC	93
<b>CAPÍTULO 5: PRESUPUESTO</b>	<b>94</b>
<b>CONCLUSIONES Y RECOMENDACIONES</b>	<b>95</b>
<b>BIBLIOGRAFÍA</b>	<b>98</b>

## **PRÓLOGO**

El análisis biomecánico del movimiento humano ha tenido una mejora sustancial desde que aparecieron métodos computarizados para la tarea, la combinación de los requerimientos de profesionales de la salud combinadas con las habilidades de diseño y desarrollo de herramientas de los profesionales de ramas técnicas, han resultado en que en la actualidad del proceso de análisis biomecánico el analista pueda encontrar rasgos personales del movimiento, daños patológicos, posiciones riesgosas e incluso obtener modelos matemáticos del movimiento de cada parte del cuerpo.

La herramienta diseñada en esta tesis ofrece las prestaciones perfectas para el análisis ergonómico de riesgo laboral, ofreciendo al analista la posibilidad de detectar posiciones que pueden ser peligrosas a largo plazo, detectarlas antes de que el problema patológico se evidencie en el paciente, de esta manera la rutina de trabajo no afectará su salud y desempeño diario dentro o fuera del entorno laboral.

## INTRODUCCIÓN

**D**esde un punto de vista conceptual la representación en realidad aumentada es una herramienta de alto nivel cuando se trata de análisis biomecánico de movimiento humano, sea cual sea su área de enfoque (ergonómico, deportivo, etc.). Desde una perspectiva funcional, la realidad aumentada es una simulación en la cual, modelos bidimensionales y tridimensionales generados por computadora están en constante cambio para adaptarse a la realidad que se intenta recrear, es así que la combinación de software y hardware para realizar tareas de este tipo requiere de procesos que involucran algoritmos precisos, eficaces y veloces.

Basado en las características necesarias para un sistema de realidad aumentada, se presenta en esta tesis el desarrollo de una herramienta que integra un modelo en realidad virtual para fines de monitoreo y análisis del movimiento humano enfocado al análisis ergonómico.

El desarrollo del hardware y software necesarios para tener un sistema como este, requieren de modelos de funcionamiento precisos, grandes cantidades de datos, un sistema de transmisión libre de fallas y un trabajo coordinado, organizado y sincronizado de cada una de las partes. El sistema mostrará los datos calculados y simulados a través de una interfaz gráfica que permitirá al usuario analizar los movimientos y ángulos en los que se encuentran las partes del cuerpo del paciente, además de tener una simulación en 3D que representa al sujeto de análisis en un modelo generalista del cuerpo humano.

El presente trabajo está estructurado de manera secuencial, describiendo en cada uno de sus capítulos desde el planteamiento de la idea, hasta el desarrollo y la solución de problemas para llegar al objetivo general, se han realizado pruebas para verificar la confiabilidad de los datos obtenidos y conocer las limitaciones del sistema diseñado, consiguiendo prometedores resultados que serán el aval para cualquier trabajo en el futuro.

## **OBJETIVO**

Esta tesis tiene como objetivo el diseño, construcción e implementación de un sistema de captura de movimiento para análisis ergonómico de riesgo laboral de extremidades superiores, la herramienta deberá presentar datos al profesional de la salud suficientes y confiables para que pueda realizar un análisis de riesgo ergonómico laboral basado en métodos científicos reconocidos en este ámbito a nivel mundial sobre el paciente en su entorno y rutina de labores. El sistema deberá ser amigable con el paciente también, tratando de ser lo menos intrusivo posible durante el análisis, permitiéndole realizar su rutina laboral de manera natural y cómoda.

## **CAPÍTULO 1**

### **MARCO TEÓRICO**

#### **1.1 ERGONOMÍA**

Es la disciplina científica que se ocupa de la comprensión fundamental de las interacciones entre los seres humanos y el resto de los componentes de un sistema, aplicando principios teóricos, datos y métodos para optimizar el bienestar de las personas y el rendimiento global del sistema.

##### **1.1.1 RIESGO LABORAL**

Este término representa la probabilidad de daño que una persona enfrenta en su actividad laboral, la cual si no es controlada y eliminada tiene la posibilidad de convertirse en un accidente laboral o en una enfermedad profesional.

##### **1.1.2 RIESGO ERGONÓMICO LABORAL**

Término utilizado para definir la probabilidad de que una persona sufra un accidente, o una enfermedad ocupacional tras su interacción cotidiana con su ambiente laboral, depende en gran medida a su tiempo de exposición, repetición, fuerza ejercida, posición, ángulos críticos de articulaciones, entre otros.

### **1.1.3 ANÁLISIS ERGONÓMICO DE PUESTO DE TRABAJO**

Al momento de hablar de un análisis ergonómico, no solo estamos hablando únicamente de factores físicos, sino que se debe tomar en cuenta elementos como Organización, Ambiente, Medios y Espacio de Trabajo, por lo que el análisis ergonómico se divide en:

- Ergonomía Geométrica
- Ergonomía Ambiental
- Ergonomía Cognitiva
- Ergonomía Organizativa

### **1.1.4 ERGONOMÍA AMBIENTAL**

Esta parte del análisis se encarga de analizar factores como:

- Ambiente Visual
- Ambiente Sonoro
- Ambiente Térmico
- Calidad del Aire

Este tipo de análisis son propios del medio ambiente laboral, y cuida sentidos como la vista y el oído del sujeto, además como sistema respiratorio y sistema inmunológico.

### **1.1.5 ERGONOMÍA COGNITIVA**

En este caso, este análisis busca evaluar aspectos mentales del trabajador como:

- Percepción
- Razonamientos
- Respuestas Motrices

### 1.1.6 ERGONOMÍA ORGANIZATIVA

Optimización de los sistemas socio técnicos, este análisis trata de analizar aspectos como:

- Comunicación dentro del trabajo
- Concepción del Trabajo
- Diseño de horarios de Trabajo

### 1.1.7 ERGONOMÍA GEOMÉTRICA

Este tipo de análisis ergonómico es en el que se hará énfasis durante el presente proyecto, ya que analiza la parte de actividad física de la rutina laboral, donde se analizarán principalmente:

- Posturas de Trabajo
- Manipulación de Objetos
- Movimientos Repetitivos
- Disposiciones de puestos de Trabajo

#### 1.1.7.1 PROCESO DE ANÁLISIS ERGONÓMICO GEOMÉTRICO

El análisis ergonómico geométrico se lo ha dividido en 5 etapas que forman un ciclo en constante repetición con efectos de estar siempre al pendiente del bienestar del sujeto en su rutina física cotidiana.

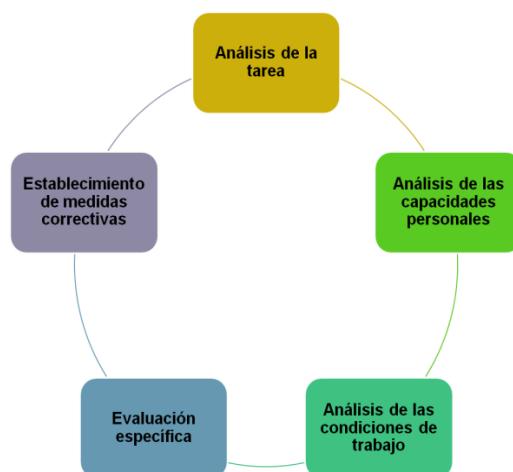


Figura 1.1: Ciclo del Análisis Ergonómico Geométrico.

- **Análisis de Tarea:** El ciclo comienza con el análisis de la tarea que realiza el sujeto, considerado el proceso clave y por lo tanto el más importante ya que será el que dirija a todo el resto de procesos a realizar, en esta parte del análisis, el especialista deberá fijarse

esencialmente en fuerza aplicada, repeticiones, tiempo de exposición, posiciones críticas, entre otros factores que podrían llevar al paciente a sufrir primero cansancio muscular, inflamaciones que puedan acabar en enfermedades profesionales que afecten el desempeño laboral y en otros aspectos al trabajador.

- **Análisis de las capacidades personales:** Este punto trata de delimitar la capacidad física del sujeto dado que no todos somos aptos para todo tipo de esfuerzo físico o posición, ya sea esta por una mala postura, una discapacidad u cualquier otro limitante a cierta actividad física.

- **Análisis de condiciones de trabajo:** Este tipo de análisis identifica riesgos en el puesto de trabajo físicamente hablando, es decir una altura incorrecta de una mesa de trabajo, una pantalla incorrectamente ubicada u otros parámetros que influyan a que por un mal diseño físico del puesto del trabajo se provoque posiciones forzadas y que causen malestar al trabajador.

- **Evaluación Específica:** Existen muchos métodos de evaluación de riesgos físicos en el puesto de trabajo, en todos estos se analiza de manera muy detallada parámetros que ya se habían explicado previamente como fuerza aplicada, repeticiones, posturas críticas, entre otros. Muy bien, esta evaluación específica analiza todos los datos recogidos a través de los 5 pasos anteriores para obtener una estimación del riesgo ergonómico laboral al cual está expuesto el individuo.

Existen diferentes métodos analíticos que son una herramienta para estimar el riesgo, todos ellos basados en criterios de medicina ocupacional así como ergonomía, los cuales ayudarán a obtener datos rápidamente acerca de la condición del sujeto observado y su nivel de riesgo ergonómico geométrico laboral.

- **Establecimiento de medidas correctivas:** Una vez conocido el nivel de riesgo al que está sometido un sujeto en su puesto de trabajo, queda actuar responsablemente y evitar un accidente o cualquier enfermedad laboral que a lo largo del tiempo se puede dar debido a una mala condición postural, o tal vez un exceso de fuerza, para lo cual se toman en cuenta los puntos de más riesgo y se actúa en primer lugar, estos puntos son los llamados de *alto riesgo*, para luego ir a *medio riesgo* y *bajo riesgo*, donde las de alto riesgo son totalmente inaceptadas y deberán ser corregidas de manera urgente para evitar un accidente, medio riesgo nos indica un riesgo de menor intensidad que

no es urgente, sin embargo es importante atenderla para evitar una enfermedad profesional dada por un tiempo de exposición que nunca se controló. Por último el riesgo bajo, la cual es un riesgo aceptable, con bajas probabilidades de causar una enfermedad profesional, y que debe ser atendido en caso de que se puedan tomar medidas correctivas.

### **1.1.7.2 MÉTODOS SISTEMÁTICOS DE ANÁLISIS ERGONÓMICO DE RIESGO DE UNA TAREA**

Hasta el momento se ha tenido un enfoque muy general de lo que respecta a Ergonomía, Análisis Ergonómico de Riesgo Laboral y otros conceptos básicos, pues bien, este proyecto se centra en construir un sistema de sensores para un sistema de análisis ergonómico de riesgo laboral; este sistema lo que hará es recolectar gigantescas cantidades de datos que detallen el movimiento específico del sujeto de análisis para dejarlos listos a aplicar un método sistemático que estime el riesgo ergonómico al que un trabajador realizando su específica rutina laboral está expuesto.

Los métodos más confiables y usados para el análisis de movimientos repetitivos son:

- RULA
- OWAS
- REBA
- OCRA
- NIOSH

En este proyecto, se tomará únicamente uno de ellos, el cual se adapta mejor a este proyecto en particular.

### **1.1.7.3 MÉTODO RULA**

El método RULA fue desarrollado por los doctores McAtamney y Corlettde la Universidad de Nottingham en 1993, para evaluar la exposición de los trabajadores a factores de riesgo que pueden ocasionar trastornos en los miembros superiores del

cuerpo: posturas, movimientos repetitivos, fuerzas aplicadas, actividad estática del sistema músculo-esquelético.

La utilización de este método empieza con la observación del trabajador en su período de trabajo, observar tareas repetitivas, posturas críticas en relación a la duración y la carga postural.

A partir de este análisis la evaluación se da sobre las posturas a las que está sometido el trabajador en forma angular, además del tiempo de exposición a estas.

El método RULA deberá estar conformado por los siguientes procesos para garantizar un correcto análisis:



Figura 1.2: Conjunto de Procesos para un correcto análisis ergonómico RULA.

Todos los procesos son esenciales para un correcto diagnóstico. En este proyecto, el objetivo será construir la herramienta que permita reemplazar el proceso de grabación y análisis de video, por un proceso que a través de un sistema de tecnología de punta permita analizar el movimiento del sujeto de una manera exacta y en un tiempo record, cosa que realizada manualmente, primero que pierde precisión y toma demasiado tiempo al especialista.

El análisis RULA divide al cuerpo en 2 Grupos:

**Grupo A:** Incluye miembros superiores (brazos, antebrazos y muñecas).

**Grupo B:** Incluye Piernas, el tronco y el cuello.

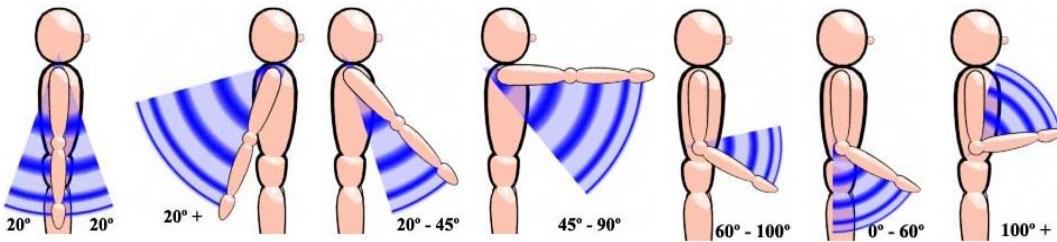


Figura 1.3: Grupos método RULA.

#### 1.1.7.4 PROCEDIMIENTO DE ANÁLISIS

El procedimiento para el análisis de riesgo ergonómico según el método RULA se basa en los siguientes parámetros.

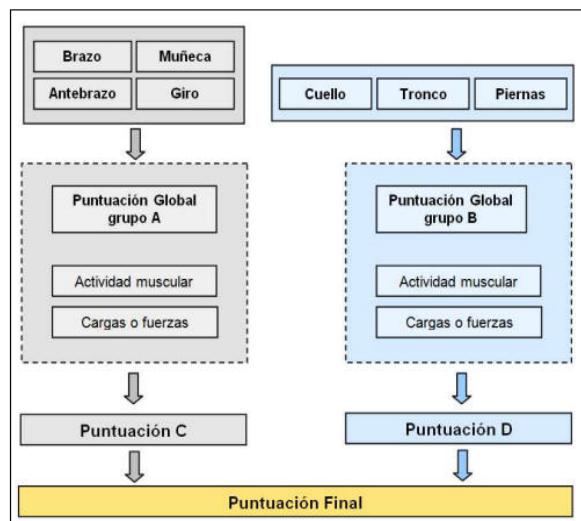
- a. Se deben determinar los ciclos de trabajo y observar al trabajador en los mismos.
- b. Se deben seleccionar las posturas que se evaluarán.
- c. Determinar si se evaluará lado izquierdo, derecho o ambos.
- d. Se deben fijar las puntuaciones para cada parte del cuerpo.
- e. Se debe obtener la puntuación final del método y el nivel de actuación que consideran, siendo el nivel uno el valor que evalúa la postura más aceptable y el nivel cuatro un cambio urgente de postura, por lo tanto a mayor valor, mayor riesgo de tarea.
- f. Se deberá aplicar correcciones a las tareas realizadas por grupos de mayor riesgo, esto puede ser un cambio de la rutina laboral o un rediseño del puesto de trabajo.
- g. En caso de tener cambios de actividad o diseño de puesto de trabajo, realizar nuevamente la evaluación a través del método RULA para observar mejora.



**Figura 1.4:** Algunas posiciones que se deberán tomar en cuenta para el análisis RULA de Riesgo Ergonómico Laboral.

### 1.1.7.5 EVALUACIÓN GLOBAL

El método RULA además de permitir un análisis específico del riesgo de trastorno músculo esquelético de cada miembro de cada grupo, brinda un pronóstico global del riesgo, este valor se lo obtiene con el valor estimado de cada uno de los grupos apreciados en la figura 1.4 a través del siguiente método:



**Figura 1.5:** Proceso para calcular riesgo ergonómico global RULA.

De esta manera se puede estimar el riesgo ergonómico de una tarea, el método sugiere métodos precisos para el cálculo de cada una de las puntuaciones según el grupo que se analice, la puntuación final nos dirá cuantitativamente cual es el riesgo ergonómico tiene una persona al realizar determinado trabajo.

## 1.2 SENSORES INERCIALES

La estructura de sensores de movimiento necesarios para medir desplazamiento tanto lineal como angular deben estar basados en sensores inerciales como son el

acelerómetro, el giroscopio y magnetómetro, estos sensores son llamados sensores iniciales.

Los sensores iniciales de movimiento tuvieron su inicio con las consolas de juegos como el Nintendo Wii, o en teléfonos inteligentes por primera vez en el Iphone, estos sensores detectan aceleraciones lineales y angulares, lo cual permite conocer su velocidad de desplazamiento, la distancia recorrida e incluso su dirección.

Este tipo de sensores son llamados MEMS (Sistemas Micro Electro Mecánicos), y se basan en la detección de 5 movimientos básicos como son:



Figura 1.6: 5 Movimientos básicos asociados con la aceleración de un objeto.

Aceleración, vibración, golpe (choque, shock), inclinación (tilt) y rotación (pan) son los cinco movimientos fundamentales que un sistema inteligente debe detectar a todo momento para tener un control pleno sobre el objeto que desea gobernar o interpretar. Todas son, en realidad, manifestaciones diferentes de una aceleración durante períodos de tiempo distintos. Sin embargo, los seres humanos no relacionamos de manera intuitiva estos movimientos como variaciones en la aceleración/desaceleración de un cuerpo. En cambio, si tenemos en cuenta y analizamos cada modalidad por separado podremos comprender de manera más sencilla muchas posibilidades que un acelerómetro puede ofrecernos. La aceleración (incluyendo el movimiento de traslación) mide la variación de velocidad en una unidad de tiempo. La velocidad se expresa en metros por segundo (m/s) e incluye tanto la tasa de desplazamiento como la dirección del movimiento (vector).

### **1.2.1 ACELERÓMETRO**

Los sensores utilizados para medir la aceleración se denominan acelerómetros. Un acelerómetro es un instrumento para medir la aceleración de un objeto al que va unido, lo hace midiendo respecto de una masa inercial interna.

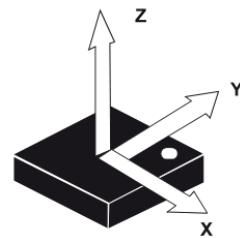
Los acelerómetros son sensores inerciales que miden la segunda derivada de la posición, mide la fuerza de inercia generada cuando una masa es afectada por un cambio de velocidad.

Existen varios tipos de tecnologías (piezo-eléctrico, piezo-resistivo, galgas extenso métricas, láser, térmico) y diseños que aunque todos tienen el mismo fin pueden ser muy distintos unos de otros según la aplicación a la cual van destinados y las condiciones en las que han de trabajar.

Hay dos parámetros principales a la hora de escoger el medidor adecuado, los rangos de funcionamiento de temperatura y frecuencia. Otros parámetros importantes pueden ser el tamaño, si tienen más funciones, la resistencia a golpes y por supuesto el precio.

Los avances en tecnología electromecánica micro de los sistemas (MEMS) han permitido la detección del movimiento o los sensores de inercia, conocidos como acelerómetros, para ser puesto en ejecución en muchos usos para las varias industrias.

Los acelerómetros están entre los primeros productos de micro sistemas (MST/MEMS) desarrollados, surgieron en el final de la década de 1980. Sin embargo, para alcanzar un éxito comercial necesitó el desarrollo que surgió durante las décadas de los 70, 80, hasta la del 90 con aplicaciones principalmente en los mercados de la automoción y aeronáutica. Los sensores micrómetro-clasificados miden el movimiento tal como aceleración, vibración, choque, e inclinación. Actualmente, con la tecnología muy madura, fabricación en volúmenes muy elevados y a un bajo costo, los acelerómetros están en la mejor posición para moverse con éxito hacia otras aplicaciones, tales como el área médica, industrial y de transporte.



**Figura 1.7: Lectura de aceleración lineal en los 3 ejes de un acelerómetro.**

El acelerómetro en resumen mide la aceleración lineal en m/s sobre cada uno de sus ejes, como se ve en la figura 1.7.

El modelo matemático de un acelerómetro es el siguiente:

$$a = G_a a_0 + b_a + n_a \quad (1.1)$$

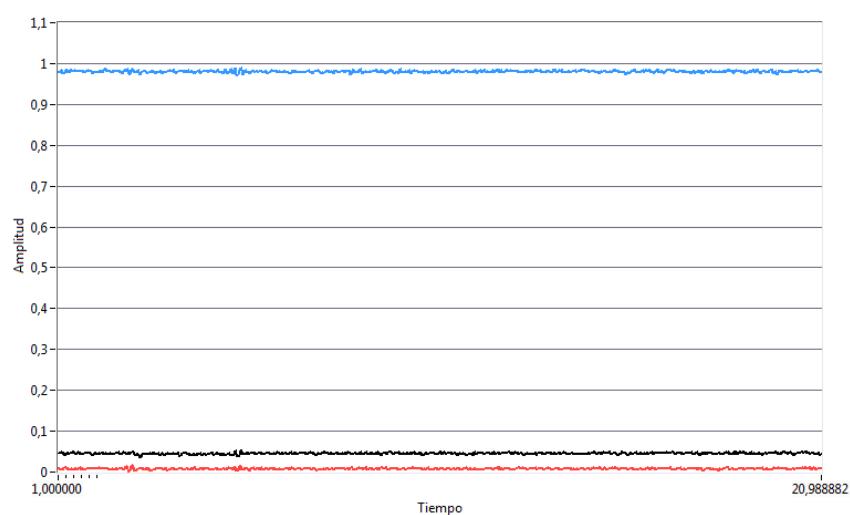
El término  $G_a$  es el factor de escala, este es dado por el fabricante.

El término  $a_0$  es el valor medido de aceleración por el sensor inercial.

El término  $b_a$  es el bias, que es un error sistemático que puede ser calculado, también es dado por el fabricante.

El término  $n_a$  es ruido blanco gaussiano.

Se debe tener muy en cuenta que tanto el factor de escala como el bias que entrega el fabricante no son parámetros exactos. Estos deben ser tomados en cuenta en caso de no tener apoyo alguno a los datos del acelerómetro.



**Figura 1.8: Lectura de los 3 ejes del acelerómetro en estado estático.**

Como se puede observar las mediciones de los 3 ejes del acelerómetro del integrado MPU 6050 que se revisará a fondo posteriormente los datos de medición en estado estático son inestables debido al error gaussiano y el bias.

La desviación estándar medida en cada eje es variable entre sensores, aunque pequeña no es despreciable, las mediciones en varios sensores dan mediciones de desviación estándar que se pueden notar van desde 0.0034 hasta 0.0096, sabiendo que 1 es el valor de la gravedad, es decir que la desviación estándar es de entre  $0.03332 \text{ m/s}^2$  a  $0.09408 \text{ m/s}^2$  una vez multiplicado por el factor de escala.

### 1.2.2 GIROSCOPIO

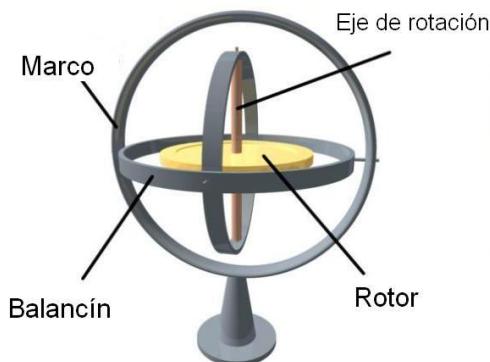


Figura 1.9: Esquema típico de un giroscopio.

Como se puede apreciar en la figura anterior un giroscopio está compuesto por 4 elementos. Un giroscopio o giróscopo es un dispositivo con característica esférica en su forma con un objeto en su centro en forma de disco, montado en un soporte cardánico, de manera que pueda rotar libremente en cualquier dirección sobre su eje de simetría. Su principio de funcionamiento está basado en la conservación del momento angular, por eso es utilizado para medir la orientación o para mantenerla haciendo uso de las fuerzas que ejercen en su sistema de balanceo.

La clave es entender que los movimientos de fuerzas externas para desviarlos están en realidad repartidos por la superficie giratoria de forma uniforme, con la fuerza dividida sobre todos los puntos de contacto, no únicamente en el punto sobre el que aparentemente se aplica.

La tendencia del eje del rotor de mantener la orientación original del movimiento a menos que sea forzado físicamente a girar en sentido contrario, permaneciendo su centro de masas estático.

El modelo matemático del giroscopio es parecido al del acelerómetro, aunque en realidad ahora no se mide aceleración lineal, sino velocidad angular:

$$\omega = G_g \omega_0 + b_g + n_g \quad (1.2)$$

El término  $G_g$  es el factor de escala, este es dado por el fabricante.

El término  $\omega_0$  es el valor medido de velocidad angular por el sensor inercial.

El término  $b_g$  es el bias, que es un error sistemático que puede ser calculado, también es dado por el fabricante.

El término  $n_g$  es ruido blanco gaussiano.

### 1.2.3 MAGNETÓMETRO

Es un sensor que mide la intensidad de campo magnético en 3 ejes. Gracias a estas 3 medidas se obtiene un vector de campo que da información del ángulo de azimut del móvil o de su actitud. Vendría a ser la versión ampliada a 3 dimensiones de la brújula, pero este instrumento es capaz de distinguir giros en los ejes de roll y pitch.

El campo se mide a través de magneto-resistencias que cambian su valor en función del campo que las atraviesa en su dirección.

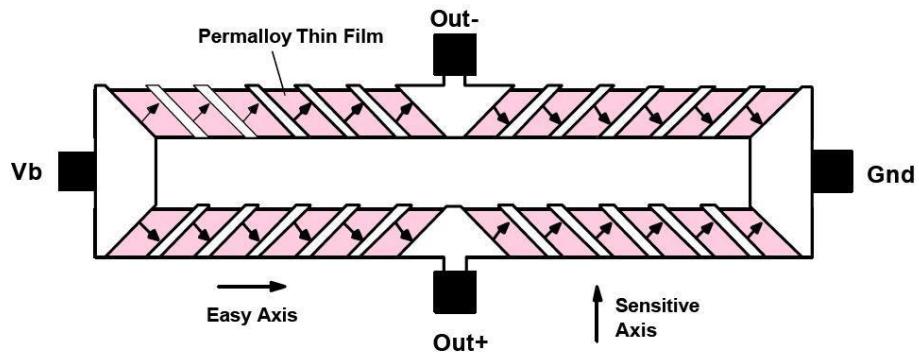


Figura 1.10: Estructura Interna de un Magnetómetro MEMS.

### 1.2.3.1 CAMPO MAGNÉTICO TERRESTRE

Para entender mejor el funcionamiento de este dispositivo y utilizarlo correctamente en aplicaciones de navegación, es necesario estudiar cómo actúa el campo Magnético Terrestre. La Tierra actúa como un dipolo, pero no está alineado con sus ejes de rotación. Por eso siempre aparece un término de Declinación, que es la separación entre el norte magnético y el norte geográfico. Por Norte magnético se entiende aquella región de la tierra donde las líneas de campo son perpendiculares a la superficie terrestre.

El campo magnético de la tierra se puede describir con 7 parámetros: declinación, inclinación, intensidad horizontal, componentes X (apuntando al norte geográfico) e Y (apuntando al Este) en el plano horizontal, intensidad vertical e intensidad total.

El valor de este campo en medida absoluta oscila entre los 25,000 y los 65,000 nT(100,000 nT = 1 gauss).

Gracias a las medidas en los 3 ejes del campo magnético, se puede calcular la orientación relativa de un objeto. En el caso de una brújula, siempre se supone que se mide en el plano horizontal a la Tierra para obtener la dirección del norte. Los compases magnéticos, utilizan un inclinómetro como referencia de posición para calcular la orientación de un objeto. De esta manera resulta sencillo calcular los ángulos de roll y pitch y posteriormente hacer la compensación de los ejes decampo magnético para calcular el heading.

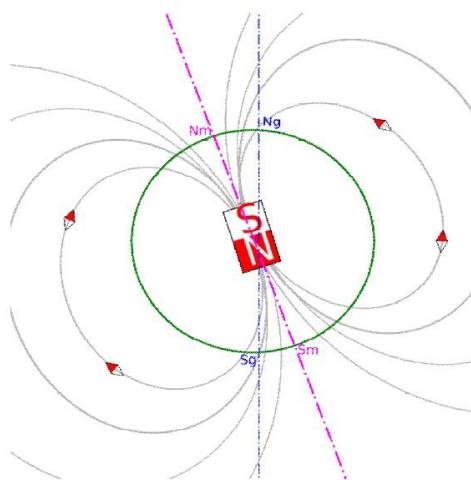


Figura 1.11: Líneas de Campo Magnético Terrestre.

El modelo matemático del magnetómetro es parecido al del acelerómetro, aunque en realidad ahora no se mide aceleración lineal, sino intensidad de campo magnético:

$$m = G_m m_0 + b_m + n_m \quad (1.3)$$

El término  $G_m$  es el factor de escala, este es dado por el fabricante.

El término  $m_0$  es el valor medido de intensidad de campo magnético.

El término  $b_m$  es el bias, que es un error sistemático que puede ser calculado, también es dado por el fabricante.

El término  $n_m$  es ruido blanco gaussiano.

## CAPÍTULO 2

### CONSTRUCCIÓN DE HARDWARE

#### 2.1 DISEÑO DEL MÓDULO DETECTOR DE MOVIMIENTO

Para la construcción de este módulo se van a utilizar como base sensores inerciales, más específicamente los 3 descritos anteriormente: acelerómetro de 3 ejes, giroscopio de 3 ejes y un magnetómetro de 3 ejes; por lo que en total el módulo detector de movimiento tendrá 9 grados de libertad. Para esto se adquirió el módulo de sensores inerciales GY-88 de Arduino:

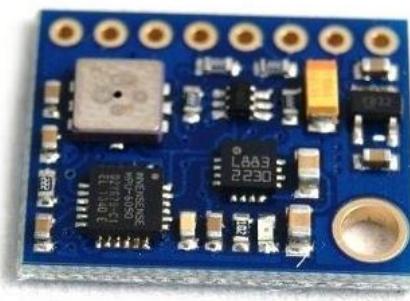


Figura 2.1: Tarjeta Arduino GY-88 IMU 10DOF.

Este módulo trae montados tres integrados que lo conforman:

**MPU-6050** es un integrado reconocido a nivel mundial de la marca INVENSENSE, este integrado tiene en su interior un acelerómetro de 3 ejes y un giroscopio de 3 ejes. La manera de acceder a sus datos es a través de comunicación I2C, puede trabajar con frecuencias de muestreo de 8KHz para el acelerómetro y de 1KHz para el giroscopio, además cada uno de los 6 grados de libertad de este integrado cuenta con registros con resolución de 16 bits, una capacidad de registro de hasta 16g para el acelerómetro y de 2000°/segundo para el giroscopio.

También está el **HMC5883L**, el cual es un magnetómetro de 3 ejes de la marca HONEYWELL. De igual manera los datos se leen a través de comunicación I2C, este integrado está conectado al mismo bus I2C que los demás integrados sobre la tarjeta. Cuenta con una frecuencia de muestreo de hasta 143Hz y para sus 3 grados de libertad cuenta con registros con resolución de 12 bits.

Esta tarjeta es de 10 grados de libertad, adicionalmente tiene un integrado que hace de barómetro, pero no se lo utiliza para este proyecto ya que esto es usado con fines aeronáuticos.

Está claro cómo y de donde obtener los datos iniciales para los módulos detectores de movimiento, ahora el problema principal es conectarse a estos y extraer los datos con una frecuencia que permita obtener estimaciones de movimiento minimizando los errores.

Cada uno de los módulos detectores de movimiento tendrá un cerebro, el cual es un microcontrolador, este se encargará de acceder a cada uno de los 18 registros de 8 bits de los sensores iniciales con una frecuencia de 100 veces por segundo, ordenarlos, y prepararlos para que estos sean enviados a procesar en el computador.

Además de esto, el microcontrolador tiene que tener la capacidad de comunicarse con un módulo que enviará los datos inalámbricamente hasta el computador usando protocolo TCP.

El microcontrolador elegido es el **PIC18F2550** de MICROCHIP, este PIC cuenta con comunicación I2C de hasta 400KHz y una velocidad de operación de 48MHz, lo cual le permite poder manejar 18 registros de información, ordenarlos y enviarlos 100 veces por segundo.

Ahora, se debe escoger el método para enviar los datos recogidos de los sensores iniciales. El objetivo es que los módulos detectores de movimiento sean lo más pequeños posibles y no sean invasivos para la persona que los usa, lo que indica que este debe ser un módulo inalámbrico, tener largo alcance dentro de lugares cerrados y una excelente velocidad de transmisión.

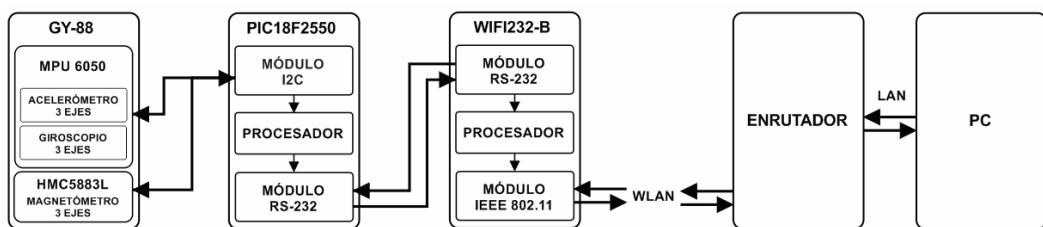
El módulo de comunicación a usar es el WIFI232-B de la marca USR, un módulo que utiliza comunicación serial de entrada y es capaz de enviar datos en protocolo TCP de salida, de manera que se lo puede utilizar para convertir la comunicación serial de un micro controlador sencillo en una interfaz inalámbrica manejada con IP y que usa el protocolo TCP para enviar los datos sobre estándar IEEE 802.11 b/g/n el cual se usa para redes WLAN o más conocido como Wifi.



**Figura 2.2: Módulo de Comunicación Serial/WIFI 802.11 b/g/n WIFI232-B.**

Este módulo tiene la capacidad de convertir tramas de datos recibidas a través de su puerto serie en tramas TCP, además de contar con posibilidad de trabajar como un simple componente con dirección IP o también funcionar como enrutador o como un punto de acceso, trabaja con protocolo IEEE 802.11 b/g/n a una velocidad máxima de 150MBPS teóricamente.

Una vez que estos componentes trabajen en conjunto, el dispositivo deberá tomar datos de aceleración lineal, velocidad angular y campo magnético terrestre, mientras inmediatamente prepara el paquete de datos el cual se envía por lo menos 100 veces por segundo hasta el PC, el cual se encargará de procesar el paquete de datos y transformarlos en una medida del movimiento de cada una de las articulaciones del paciente.



**Figura 2.3: Esquema de Funcionamiento de Módulo Detector de Movimiento.**

Como se puede apreciar, de esta manera irá estructurado cada uno de los 7 módulos detectores de movimiento del sistema

Para que sea inalámbrico, cada uno de los módulos además de poder transferir datos usando una red Wifi, deberá tener su propia fuente de alimentación. Los módulos consumen alrededor de 350mA, esto quiere decir que se necesita baterías pequeñas y

de un buen desempeño. Se ha optado por baterías de tipo Ni-Mh, estas son baterías recargables, tiene un ánodo de oxihidróxido de níquel como las baterías de Ni-Cd (típicas de muchos juguetes con batería recargable), pero cuyo cátodo es una aleación de metal hidruro, esto permite tener bajo costo, mayor capacidad de carga en tamaño más pequeño y menor efecto de memoria, además de tener una capacidad de carga de entre 500 y 2000 veces. Estas las podemos encontrar en las típicas pilas AAA recargables.

Cada pila de estas tiene un potencial de 1.2V, para cada uno de los módulos se utilizarán 5 pilas colocadas en serie, para obtener un potencial de 6V que alimente a los reguladores necesarios para los sensores iniciales y módulo de comunicación que funcionan a 3.3V y el microcontrolador que funciona con 5V.

## 2.2 INTERCONEXIÓN DE PARTES

Tras varias pruebas la interconexión de las partes esquematizadas en la figura 2.3 es tal y como se muestra en el esquema de circuitería para cada módulo detector de movimiento en la figura 2.4.

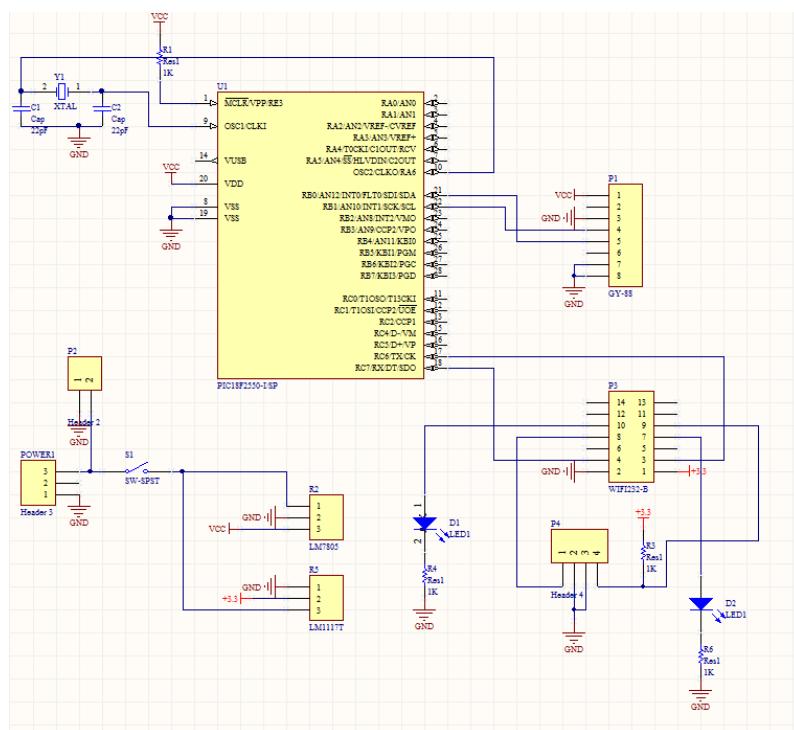


Figura 2.4: Esquema de circuitería para cada módulo detector de movimiento.

## 2.3 DISEÑO Y CONSTRUCCIÓN DE TARJETA ELECTRÓNICA

El diseño del PCB para cada módulo detector de movimiento se lo ha realizado tal cual se muestra en el circuito de la figura 2.4, este módulo será totalmente inalámbrico e independiente uno de otro.

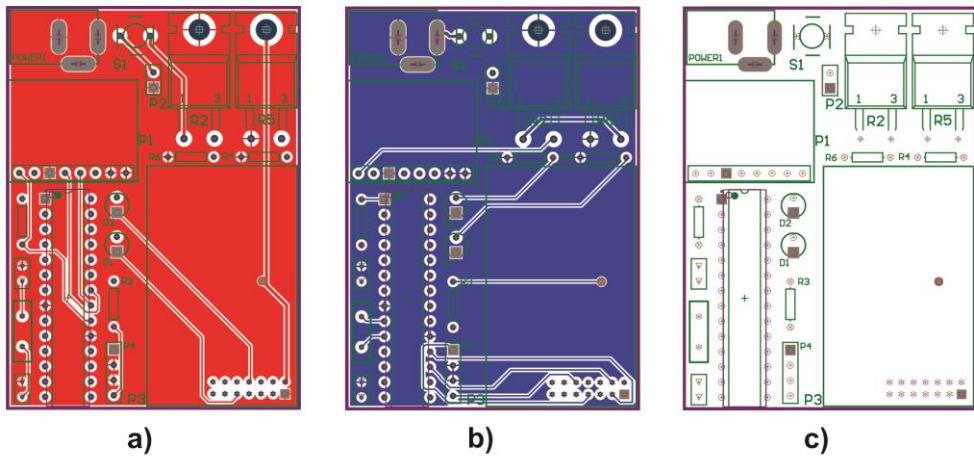


Figura 2.5: a) Capa Superior, b) Capa Inferior, c) Ubicación de Componentes y agujeros.

La PCB para cada módulo detector de movimiento fue construida según el diseño la figura 14, el PCB tiene una dimensión de 4.7 x 6.5cm, un tamaño lo suficientemente pequeño para sujetarlo al cuerpo sin problema.

Además la tarjeta cuenta con una terminal de carga de batería que permite cargar todos los módulos sin tener que extraer la batería.

También se cuenta con un switch de encendido / apagado.

Cada módulo cuenta con 2 leds de notificación los cuales indican el correcto inicio del software y la conexión con el enrutador. La PCB construida se muestra en la figura 2.6.

La tarjeta electrónica con todos sus componentes se la puede apreciar en la figura 2.7, se puede observar que la tarjeta incluida la batería de alimentación tiene un tamaño muy manejable a la hora de sujetarlo al cuerpo del usuario. La sujeción se realizará con una prenda especialmente diseñada que se verá posteriormente.

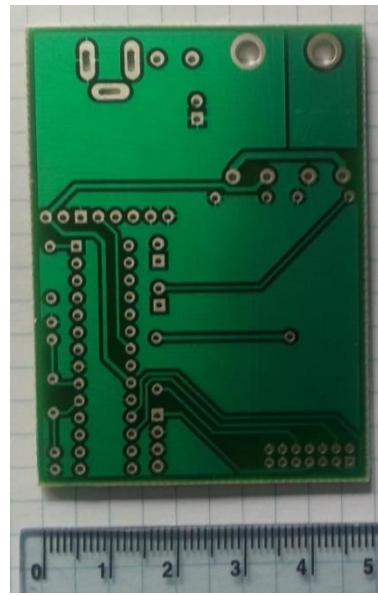


Figura 2.6: PCB de cada módulo detector de movimiento.



Figura 2.7: Módulo detector de movimiento terminado.

### 2.3.1 BATERÍA Y SISTEMA DE CARGA

La batería de cada módulo detector de movimiento está conformada por 5 baterías AAA colocadas en serie, cada batería tiene un voltaje medio de 1.2V y puede entregar hasta 1100mAh.

En total se tendrá una batería de 6V, la cual alimenta a 2 reguladores de voltaje de 5 y 3.3V, ventajosamente el regulador de 5V únicamente alimenta al microcontrolador el cual necesita alrededor de 25mA para trabajar, por eso el regulador puede mantener 5V a su salida con una entrada de apenas 1V superior.



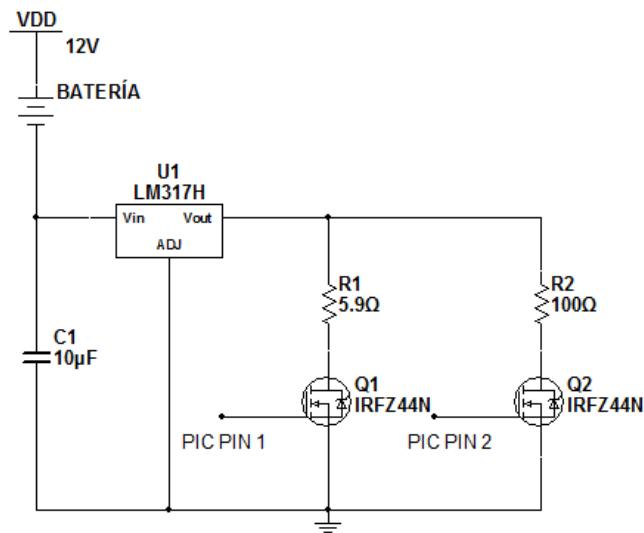
**Figura 2.8: Batería para cada módulo detector de movimiento.**

Para poder recargar estas baterías se ha diseñado un software y hardware que pueden cargar hasta 8 de estas baterías simultáneamente, es decir es como tener un cargador para 40 pilas AAA.

El cargador de baterías es controlado por un microcontrolador PIC16F877A, donde este se encarga de medir el voltaje y tiempo de carga de cada una de las baterías.

### 2.3.2 DISEÑO DE CIRCUITO CARGADOR DE BATERÍA NI-MH

El cargador de batería se basa en un circuito que básicamente controla la corriente que pasa por las baterías hasta llegar a un voltaje determinado en cada una. El circuito es el siguiente:



**Figura 2.9: Circuito cargador para una batería.**

El cargador lo que hace es controlar la corriente de carga de cada batería. Al activar o desactivar los transistores MOSFET Q1 o Q2, estos causarán que diferentes tasas de corriente atraviesen sobre las baterías debido a la demanda de corriente en las resistencias R1 o R2. El voltaje a la salida de U1 siempre será 1.2V, lo que causará una corriente de hasta 200mA que cargará el circuito en alrededor de 4 horas.

Para detectar la finalización de la carga el microcontrolador deberá leer el voltaje de la batería y detectar caídas de voltaje en la batería, esto indicará que la batería está completamente cargada, cargarla más puede ocasionar que la batería se caliente y se dañe permanentemente.

Según recomendaciones de fabricantes la secuencia de carga debe iniciar con una tasa de corriente baja, alrededor de 1/100 de la capacidad de la batería en un período de 30 a 60 segundos, para después pasar a la velocidad de carga más alta que soporte la celda de carga, en este caso se trabajará con 1/5c, que quiere decir 1/5 de la capacidad de corriente máxima de la batería.

En el prototipo final se usan 16 pines para controlar las compuertas de 16 transistores MOSFET, 2 por cada batería, esto para controlar las 2 etapas de carga de cada una de las baterías.

Además se utilizan 8 canales del conversor A/D para verificar el voltaje de cada batería y poder controlar y decidir cuando la carga de la batería se detiene.

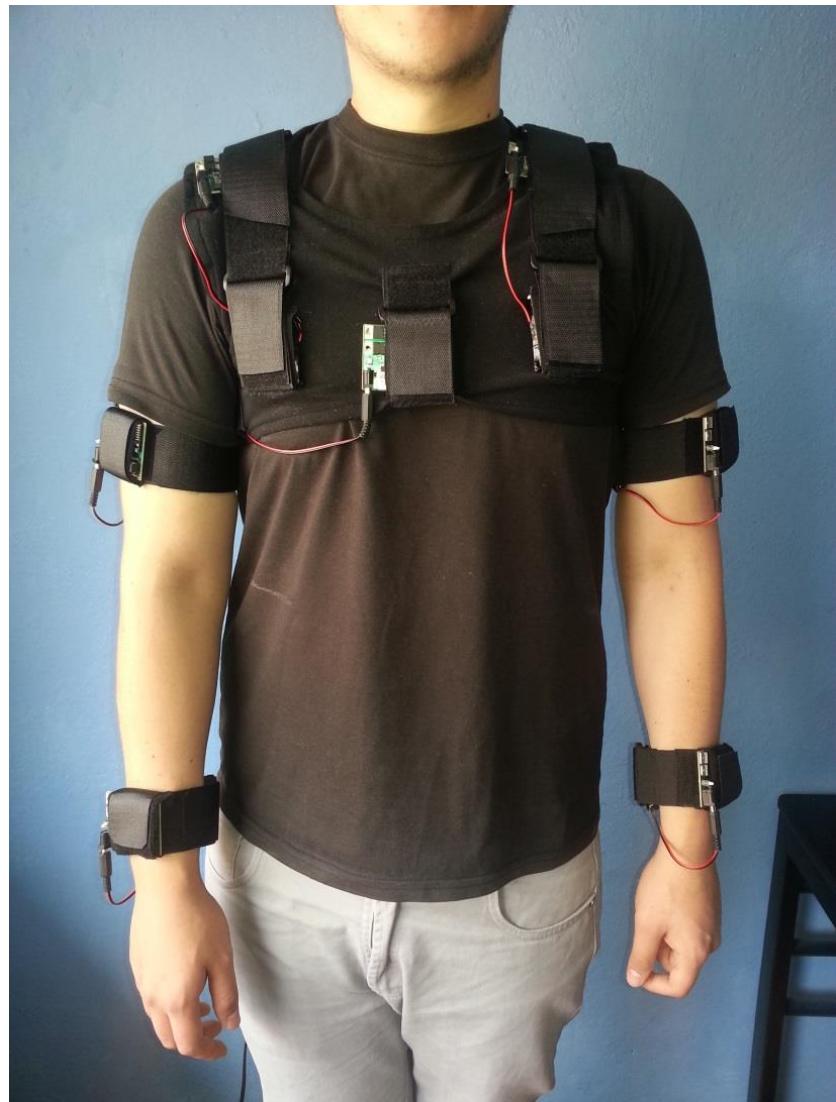
## **2.4 DISEÑO DE PRENDA PARA USUARIO**

Una vez construidos los módulos detectores de movimiento, estos deben colocarse sobre el cuerpo del usuario a fin de representar sus movimientos.

Cada uno de los módulos es inalámbrico, lo que facilita la sujeción de estos a cualquier estructura, en este caso una prenda de vestir de material flexible y con seguros para sujeción.

Como se puede ver en la figura 2.10, la prenda ha sido diseñada para no obstruir el movimiento de las articulaciones del sujeto, se pueden sujetar los 7 sensores sobre ésta de manera segura, cómoda y sin un solo cable hasta el PC.

Los sensores se guardan en bolsillos cerrados y pegados a cada segmento del cuerpo del usuario.



**Figura 2.10: Prenda de sujeción para usuario.**

## CAPÍTULO 3

### 3 DISEÑO DEL SOFTWARE

El sistema consiste en 7 módulos detectores de movimiento que envían los datos recogidos por sus sensores inerciales 100 veces por segundo, en cada uno de estos módulos se realizan los siguientes 3 procesos:

- Se solicitan los datos a los sensores inerciales en el sub módulo GY-88 mediante comunicación I2C.
- Se ordenan los datos recibidos en tramas de 450bytes para ser enviadas cada 250ms.
- Se envían 1800 bytes por segundo por el puerto RS-232 hasta el sub módulo WIFI232-B.

Los datos recogidos por el módulo detector de movimiento pasan al programa en el computador diseñado en LabView, que realiza los siguientes procesos:

- Recibe los datos enviados de los 7 sensores a través de TCP.
- Ordena los datos y los divide según el sensor inercial de procedencia: acelerómetro, giroscopio o magnetómetro; y según el eje al que pertenezcan: x, y o z.
- Empieza a calcular la orientación de cada sensor en 3 dimensiones, calcula Roll, Pitch y Yaw.
- Unifica la orientación de todos los sensores en un modelo en 3D del cuerpo del usuario.
- Analiza ángulos de articulaciones y muestra resultados.

Sabemos que el hardware está correctamente concebido para el sistema que se pretende construir, ahora se debe programar software de manera que todo funcione en bloque para conseguir obtener la orientación en 3D de los módulos detectores de movimiento.

Cada parte del software tanto de los módulos detectores de movimiento como del PC se explican detalladamente en los siguientes incisos.

### 3.1 DISEÑO DE SOFTWARE DEL MICROCONTROLADOR

El funcionamiento del programa se puede esquematizar en la siguiente figura.

El programa del microcontrolador funciona como ya se había mencionado sobre un integrado PIC18F2550, trabaja a una frecuencia de ciclo de 48MHz. El micro controlador corre con oscilador interno apoyado en uno externo de 20MHz y el *Postscaler* interno para obtener los 48MHz.

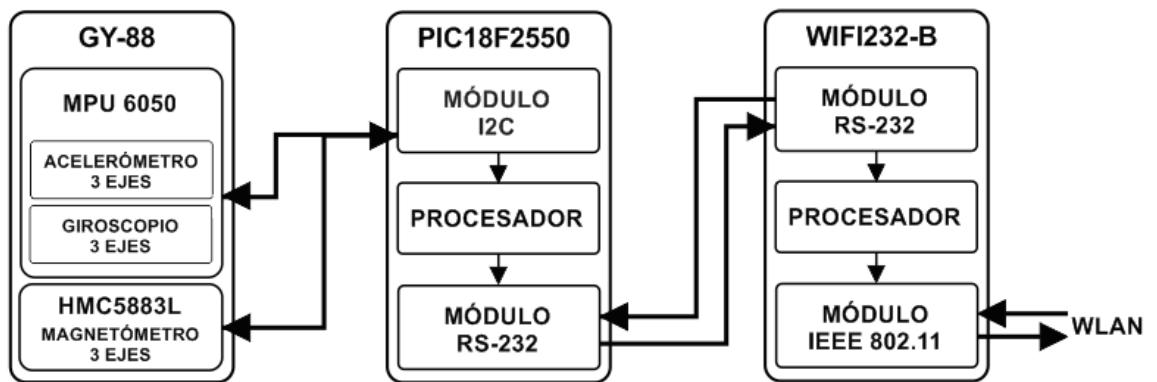


Figura 3.1: Esquema de trabajo del software del módulo detector de movimiento.

El PIC18F2550 se escogió debido a las siguientes necesidades:

- Velocidad de procesamiento alta (48MHz)
- Tamaño medio (28 pines)
- Comunicación I2C y RS-232 simultánea
- Temporizador de 16 bits

El objetivo único de este programa es transferir los datos recogidos de los sensores iniciales y magnetómetro hasta el PC de manera ordenada 4 veces por segundo en tramas de 452bytes, estas tramas tendrán la información equivalente a tomar 100 muestras por segundo los datos de cada uno de los 3 sensores que tiene cada módulo detector de movimiento.

### 3.1.1 CÓDIGO DE PROGRAMA

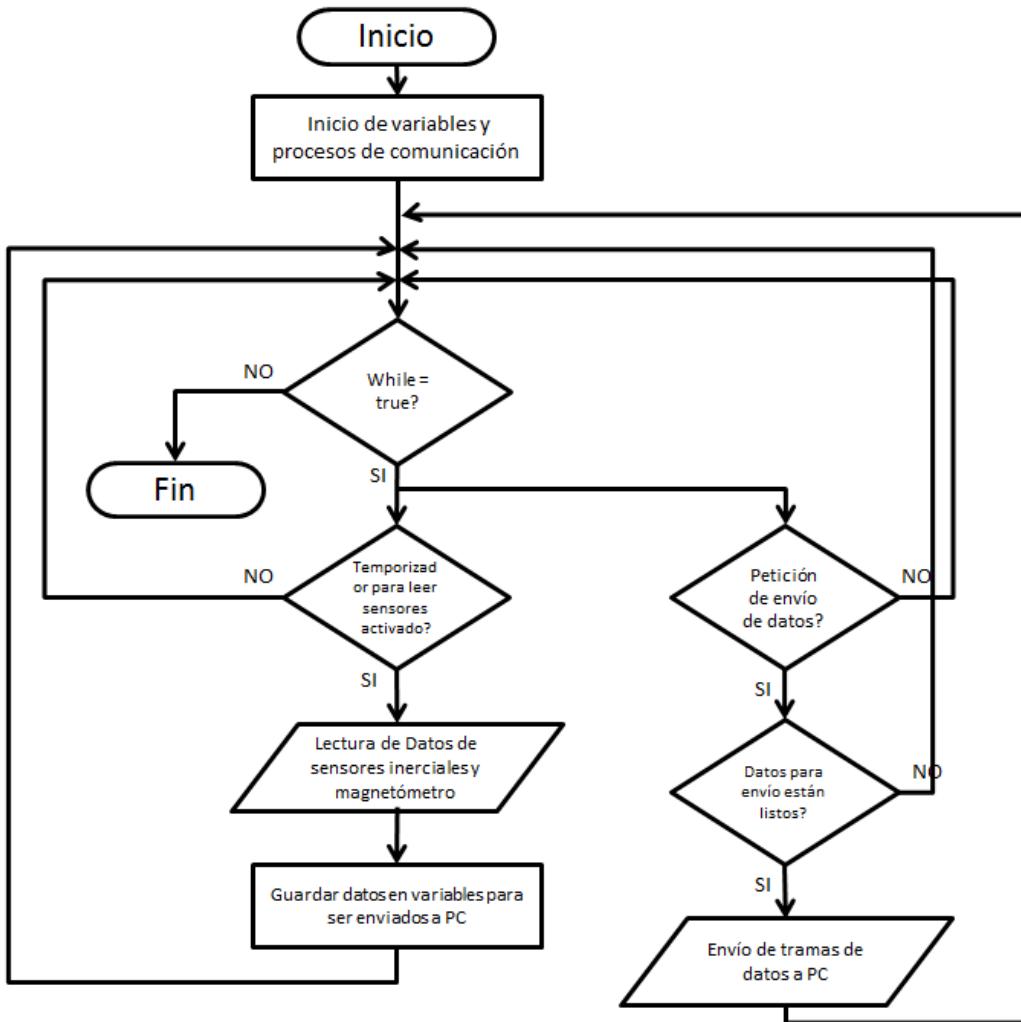


Figura 3.2: Diagrama de Flujo de Software de Microcontrolador.

El programa se lo realizó en el compilador MikroBasic, de MICROCHIP.

El funcionamiento del programa se muestra en el diagrama de flujo de la figura 3.2.

El código de programa se colocará tal cual está escrito en el compilador y se explicará cada parte.

Los procesos realizados por el programa del micro controlador son los siguientes:

### 3.1.2 DECLARACIÓN DE VARIABLES

Como se puede apreciar, existen 4 variables de tipo matriz de bytes de un tamaño de 452 ítems, estos se llenarán con los datos leídos de los sensores iniciales.

```

.     program Sensores
.         'Declaracion de Variables
.         dim datos1 as byte[452]
.         dim datos2 as byte[452]
.         dim datos3 as byte[452]
.         dim datos4 as byte[452]
.         dim cont1 as word
.         dim int_cont as byte
.         dim bandera_leer as boolean
10        dim enviar1 as boolean
11        dim enviar2 as boolean
.         dim enviar3 as boolean
.         dim enviar4 as boolean
.         dim leer1 as boolean
.         dim leer2 as boolean
.         dim leer3 as boolean
.         dim leer4 as boolean
.         dim secuencia as byte
.         dim muestra as byte
20        dim secuencia2 as byte
.         dim secuencia3 as byte
.         dim recibido as boolean
.         dim enviado as byte
.         dim enviar as byte
.         'Declaracion de Constantes
.         const MPU6050W=208
.         const MPU6050R=209
.         const axh=59
30        const axl=60
.         const ayh=61
.         const ayl=62
.         const azh=63
.         const azl=64
.         const gxh=67
.         const gxl=68
.         const gyh=69
.         const gy1=70
.         const gzh=71
40        const gz1=72
.         const HMC5883LW=0x3C
.         const HMC5883LR=0x3D

```

Las Constantes ayudarán a acceder rápidamente a la dirección de los registros de los sensores inerciales a través de comunicación I2C. Para más información diríjase a la hoja de datos de cada uno de los sensores (Referencias bibliográficas [13], [14], [15], [16] y [17]).

### 3.1.3 SUBPROCESOS

Los subprocessos son procesos que se pueden incluir en el proceso principal con solo llamarlos, estos se declaran fuera del proceso principal.

En el siguiente subprocesso se describe la lectura de los sensores inerciales a través del protocolo I2C, como se puede apreciar en el código a continuación se tiene 4 procesos que hacen lo mismo, desde el proceso *leer\_sensores1* hasta *leer\_sensores4*.

En cada uno de estos procesos se ingresa al registro de los sensores inerciales se toma la información y se la guarda en una de las matrices de 452 bytes descritas en la declaración de variables para posteriormente ser enviado. Se incluye una secuencia de un número al final de cada trama, este número va en orden, del 1 al 100 y permite al programa del computador detectar cualquier falla en la transmisión de datos. Este número es la variable *secuencia*. Este proceso tiene procesos hermanos casi iguales que son *leer\_sensores2*, *3* y *4*, cada uno sirve para guardar datos en las diferentes matrices.

```

.   'Subprocesos para lectura de sensores inerciales, se organizan en 4 tramas de 450 bytes, total 1800 bytes por segundo.
.   sub procedure leer_sensores1
.
.       'Inicio de comunicacion I2C
.       I2C1_Start()
.       'Se inicia la lectura en Inerciales
.       I2C1_Wr(MPU6050W)
50      'Lectura aceleracion en X y luego los siguientes registros
.       I2C1_Wr(axh)
.       I2C1_Repeated_Start()
.       I2C1_Wr(MPU6050R)
.       datos1[(muestra*18)] = I2C1_Rd(1)      'AXH
.       datos1[(muestra*18)+1] = I2C1_Rd(1)      'AXL
.       datos1[(muestra*18)+2] = I2C1_Rd(1)      'AYH
58      datos1[(muestra*18)+3] = I2C1_Rd(1)      'AYL
.       datos1[(muestra*18)+4] = I2C1_Rd(1)      'AZH
60      datos1[(muestra*18)+5] = I2C1_Rd(1)      'AZL
.       datos1[(muestra*18)+6] = I2C1_Rd(1)      'TEMPH
.       datos1[(muestra*18)+7] = I2C1_Rd(1)      'TEMPL
.       datos1[(muestra*18)+6] = I2C1_Rd(1)      'GXH
.       datos1[(muestra*18)+7] = I2C1_Rd(1)      'GXL
.       datos1[(muestra*18)+8] = I2C1_Rd(1)      'GYH
.       datos1[(muestra*18)+9] = I2C1_Rd(1)      'GYL
.       datos1[(muestra*18)+10] = I2C1_Rd(1)      'GZH
.       datos1[(muestra*18)+11] = I2C1_Rd(0)      'GZL
.       I2C1_Stop()
70      I2C1_Start()
.       'Se inicia lectura en Magnetometro
.       I2C1_Wr(HMC5883LW)
.       'Lectura en X y luego el resto de registros
.       I2C1_Wr(3)
.       I2C1_Repeated_Start()
.       I2C1_Wr(HMC5883LR)
.       datos1[(muestra*18)+12] = I2C1_Rd(1)      'MXH
.       datos1[(muestra*18)+13] = I2C1_Rd(1)      'MXL
.       datos1[(muestra*18)+16] = I2C1_Rd(1)      'MYH
80      datos1[(muestra*18)+17] = I2C1_Rd(1)      'MYL
.       datos1[(muestra*18)+14] = I2C1_Rd(1)      'MZH
.       datos1[(muestra*18)+15] = I2C1_Rd(0)      'MZL
.       'Se detiene la comunicacion I2C
.       I2C1_Stop()
.       'logica para ordenar los datos recibidos
.       if muestra = 24 then
90         muestra = 0
.         leer1=%0
.         leer2=%1
.         enviar1=%1
.         datos1[450]=secuencia
.         datos1[451]=1
.         if secuencia= 100 then
.             secuencia=1
.         else
.             secuencia=secuencia+1
.         end if
.       else
.         muestra = muestra + 1
100    end if
.
.     end sub
.     'Se repite el subproceso anterior 3 veces mas, de esta manera se conseguiran 4 tramas de 450bytes por segundo.
.   sub procedure leer_sensores2
.
.       'Inicio de comunicacion I2C
.       I2C1_Start()
.       I2C1_Wr(MPU6050W)
.       'Lectura aceleracion en X
110     I2C1_Wr(axh)
.       I2C1_Repeated_Start()
.       I2C1_Wr(MPU6050R)
.       datos2[(muestra*18)] = I2C1_Rd(1)      'AXH
.       datos2[(muestra*18)+1] = I2C1_Rd(1)      'AXL
.       datos2[(muestra*18)+2] = I2C1_Rd(1)      'AYH
.       datos2[(muestra*18)+3] = I2C1_Rd(1)      'AYL

```

```

.
.           datos2[(muestra*18)+4] = I2C1_Rd(1)      'AZH
.           datos2[(muestra*18)+5] = I2C1_Rd(1)      'AZL
.           datos2[(muestra*18)+6] = I2C1_Rd(1)      'TEML
120          datos2[(muestra*18)+7] = I2C1_Rd(1)      'TEMPL
.           datos2[(muestra*18)+8] = I2C1_Rd(1)      'GXH
.           datos2[(muestra*18)+9] = I2C1_Rd(1)      'GXL
.           datos2[(muestra*18)+10] = I2C1_Rd(1)     'GYH
.           datos2[(muestra*18)+11] = I2C1_Rd(0)      'GZH
.
.           I2C1_Stop()
.           I2C1_Start()
.           'Se inicia lectura en Magnetometro
130          I2C1_Wr(HMC5883LW)
.           'Lectura aceleracion en X
.           I2C1_Wr(3)
.           I2C1_Repeated_Start()
.           I2C1_Wr(HMC5883LR)
.           datos2[(muestra*18)+12] = I2C1_Rd(1)      'MXH
.           datos2[(muestra*18)+13] = I2C1_Rd(1)      'MXL
.           datos2[(muestra*18)+14] = I2C1_Rd(1)      'MYH
.           datos2[(muestra*18)+15] = I2C1_Rd(1)      'MYL
140          datos2[(muestra*18)+16] = I2C1_Rd(1)      'MZH
.           datos2[(muestra*18)+17] = I2C1_Rd(0)      'MZL
.           'Se detiene la comunicacion I2C
.           I2C1_Stop()

.
.           if muestra = 24 then
.               muestra = 0
.               leer2=%0
.               leer3=%1
.               enviar2=%1
.               datos2[450]=secuencia
150          datos2[451]=2
.               if secuencia= 100 then
.                   secuencia=1
.               else
.                   secuencia=secuencia+1
.               end if
.           else
.               muestra = muestra + 1
.           end if
.
160          end sub
.
.           sub procedure leer_sensores3
.
.           'Inicio de comunicacion I2C
.           I2C1_Start()
.           I2C1_Wr(MPU6050W)
.           'Lectura aceleracion en X
.           I2C1_Wr(axh)
.           I2C1_Repeated_Start()
.           I2C1_Wr(MPU6050R)
.           datos3[(muestra*18)] = I2C1_Rd(1)      'AXH
.           datos3[(muestra*18)+1] = I2C1_Rd(1)      'AXL
.           datos3[(muestra*18)+2] = I2C1_Rd(1)      'AYH
.           datos3[(muestra*18)+3] = I2C1_Rd(1)      'AYL
.           datos3[(muestra*18)+4] = I2C1_Rd(1)      'AZH
.           datos3[(muestra*18)+5] = I2C1_Rd(1)      'AZL
.           datos3[(muestra*18)+6] = I2C1_Rd(1)      'TEML
.           datos3[(muestra*18)+7] = I2C1_Rd(1)      'TEMPL
170          datos3[(muestra*18)+8] = I2C1_Rd(1)      'GXH
.           datos3[(muestra*18)+9] = I2C1_Rd(1)      'GXL
.           datos3[(muestra*18)+10] = I2C1_Rd(1)     'GYH
.           datos3[(muestra*18)+11] = I2C1_Rd(0)      'GZH
.
.           I2C1_Stop()
.           I2C1_Start()
.           'Se inicia lectura en Magnetometro
.           I2C1_Wr(HMC5883LW)
.           'Lectura aceleracion en X
180          I2C1_Wr(3)
.           I2C1_Repeated_Start()
.           I2C1_Wr(HMC5883LR)
.           datos3[(muestra*18)+12] = I2C1_Rd(1)      'MXH
.           datos3[(muestra*18)+13] = I2C1_Rd(1)      'MXL
.           datos3[(muestra*18)+14] = I2C1_Rd(1)      'MYH
.           datos3[(muestra*18)+15] = I2C1_Rd(1)      'MYL
.           datos3[(muestra*18)+16] = I2C1_Rd(1)      'MZH
.           datos3[(muestra*18)+17] = I2C1_Rd(0)      'MZL
.           'Se detiene la comunicacion I2C
.           I2C1_Stop()
200        
```

```

    . if muestra = 24 then
    .   muestra = 0
    .   leer3=%0
    .   leer4=%1
    .   enviar3=%1
    .   datos3[450]=secuencia
    .   datos3[451]=3
    .   if secuencia= 100 then
    .     secuencia=1
    .   else
    .     secuencia=secuencia+1
    .   end if
    .   else
    .     muestra = muestra + 1
    .   end if
    .
    . end sub
    .

220  sub procedure leer_sensores4
    .
    .   'Inicio de comunicacion I2C
    .   I2C1_Start()
    .   I2C1_Wr(MPU6050W)
    .   'Lectura aceleracion en X
    .   I2C1_Wr(axh)
    .   I2C1_Repeated_Start()
    .   I2C1_Wr(MPU6050R)
    .
    .   datos4[(muestra*18)] = I2C1_Rd(1)      'AXH
230   datos4[(muestra*18)+1] = I2C1_Rd(1)      'AXL
    .   datos4[(muestra*18)+2] = I2C1_Rd(1)      'AYH
    .   datos4[(muestra*18)+3] = I2C1_Rd(1)      'AYL
    .   datos4[(muestra*18)+4] = I2C1_Rd(1)      'AZH
    .   datos4[(muestra*18)+5] = I2C1_Rd(1)      'AZL
    .   datos4[(muestra*18)+6] = I2C1_Rd(1)      'TEMPH
    .   datos4[(muestra*18)+7] = I2C1_Rd(1)      'TEMPL
    .   datos4[(muestra*18)+8] = I2C1_Rd(1)      'GXH
    .   datos4[(muestra*18)+9] = I2C1_Rd(1)      'GXL
    .   datos4[(muestra*18)+10] = I2C1_Rd(1)     'GYH
    .   datos4[(muestra*18)+11] = I2C1_Rd(1)     'GYL
    .   datos4[(muestra*18)+12] = I2C1_Rd(1)     'GZH
    .   datos4[(muestra*18)+13] = I2C1_Rd(0)     'GZL
    .
    .   I2C1_Stop()
    .   I2C1_Start()
    .   'Se inicia lectura en Magnetometro
    .   I2C1_Wr(HMC5883LW)
    .   'Lectura aceleracion en X
    .   I2C1_Wr(3)
    .   I2C1_Repeated_Start()
    .   I2C1_Wr(HMC5883LR)
    .
    .   datos4[(muestra*18)+12] = I2C1_Rd(1)      'MXH
    .   datos4[(muestra*18)+13] = I2C1_Rd(1)      'MXL
    .   datos4[(muestra*18)+16] = I2C1_Rd(1)      'MYH
    .   datos4[(muestra*18)+17] = I2C1_Rd(1)      'MYL
    .   datos4[(muestra*18)+14] = I2C1_Rd(1)      'MZB
    .   datos4[(muestra*18)+15] = I2C1_Rd(0)      'MZA
    .
    .   'Se detiene la comunicacion I2C
    .   I2C1_Stop()
    .
260   if muestra = 24 then
    .     muestra = 0
    .     leer4=%0
    .     leer1=%1
    .     enviar4=%1
    .     datos4[450]=secuencia
    .     datos4[451]=4
    .     if secuencia= 100 then
    .       secuencia=1
    .     else
    .       secuencia=secuencia+1
    .     end if
    .   else
    .     muestra = muestra + 1
    .   end if
    .
    . end sub
    .

```

Como se puede apreciar en este conjunto de subprocessos, todos realizan la misma acción, leer y guardar datos de los sensores iniciales, lo que cambia es las variables donde se guardan, estas 4 variables de 452 bytes son las que conforman el grupo de 1808 bytes enviados por segundo hasta el computador.

### 3.1.4 INTERRUPCIÓN

La interrupción en realidad es un sub proceso del programa, pero este no se invoca llamándolo, sino por un desborde en el contador del Timer 2 del PIC, el cual se activa cada vez que llega a 5ms en escala de tiempo. La bandera de este Timer se tiene que desactivar por software para la siguiente operación.

Debido a que los temporizadores del micro controlador no pueden llegar a cuantificar tiempos tan grandes como 10ms, se ha creado la variable *int\_cont* que se incrementa de cero hasta uno para cuantificar las dos interrupciones de 5ms cada una que indican al programa ejecutar la lectura de los sensores iniciales activando la variable *bandera\_leer*.

```
281     ' Le proceso de interrupcion controla el momento en que se toman las muestras
.     sub procedure interrupt
.         if PIR1.TMR2IF = 1 then
.             ClearBit(PIR1, TMR2IF)
.             if int_cont=1 then
.                 bandera_leer=%1
.                 int_cont=0
.             else
.                 int_cont=int_cont+1
.             end if
.         end if
.     end sub
```

### 3.1.5 PROCESO PRINCIPAL

El proceso principal es en el que se desarrolla todo el programa del micro controlador, en este se llaman a variables y subprocesos anteriormente descritos para conformar la labor designada al módulo detector de movimiento.

Se empieza configurando los registros **TRIS** de los puertos que se van a usar, luego, se inician variables, se inicia comunicación por el puerto UART y comunicación I2C.

```
291     main:
.         'Configuracion de Puerto B
.         TRISB = %11111100
.         PORTB = 0
.
.         'Datos de inicio de variables
.         bandera_leer=%0
.         enviar1=%0
.         enviar2=%0
.         enviar3=%0
.         enviar4=%0
.
.         leer1=%1
.         leer2=%0
.         leer3=%0
.         leer4=%0
.         secuencia=1
.         muestra=0
.         secuencia2=1
.         secuencia3=200
.
.         'Secuencia de Inicio de puerto RS-232
.         UART1_Init(115200)
.         I2C1_Init(400000)
.         Delay_ms(4000)
.
.         'Secuencia de inicio de comunicacion y configuracion I2C de PIC y Sensores Iniciales
.         I2C1_Start()
.         I2C1_Wr(MPU6050W)      'Configure el Acelerometro de los sensores iniciales
.         I2C1_Wr(0X6B)
.         I2C1_Wr(0XB0)
.         I2C1_Stop()
.
.         I2C1_Start()
.         I2C1_Wr(HMC5883LN)    'Configure el Magnetometro de los sensores iniciales
.         I2C1_Wr(2)
```

```

.
.
.
I2C1_Wr(0)
I2C1_Stop()
330
.
I2C1_Start()
I2C1_Wr(MMC5883LW)      'Configura el Magnetometro de los sensores inerciales
.
I2C1_Wr(0)
I2C1_Wr(24)
I2C1_Stop()
.
.
.
DELAY_MS(1000)
I2C1_Start()
I2C1_Wr(MPU6050W)
.
I2C1_Wr(0x6B)
I2C1_Wr(0X00)
I2C1_Stop()
.
.
I2C1_Start()
I2C1_Wr(MPU6050W)      'Configura el Giroscopio de los sensores inerciales
.
I2C1_Wr(0XEC)
I2C1_Wr(0X00)
I2C1_Stop()
.
.
I2C1_Start()
I2C1_Wr(MPU6050W)      'Configura el Acelerometro de los sensores inerciales
.
I2C1_Wr(0X1B)
I2C1_Wr($00010000)
I2C1_Stop()
.
.
I2C1_Start()
I2C1_Wr(MPU6050W)      'Configura el Giroscopio de los sensores inerciales
.
I2C1_Wr(0X1C)
I2C1_Wr(0X00)
360
I2C1_Stop()
.

```

Como se puede ver la velocidad de comunicación I2C es de 400000 bps (bits por segundo), mientras que la comunicación serial es de 115200 bps, velocidades que permiten enviar los 1808 bytes por segundo que se necesita.

Después de iniciado se configura el funcionamiento de inicio de los sensores inerciales a través de la escritura en sus registros de configuración.

```

.
.
.
'Inicio Timer2
.
.
.
PR2=235          'Configurar: PR2=235, delta t=0.005seg, Fs=100Hz
.
.
T2CON = %01111111 'Prescaler=16 y Postscaler=16
INTCON.GIE=%1    'Habilita interrupciones generales
INTCON.PEIE=%1   'Habilita interrupciones perifericos
370
IPR1.TMR2IF=%1  'Setea interrupcion Timer2 a Alta Prioridad
.
PIE1.TMR2IE=%1  'Habilita interrupcion del Timer2
.
PIE1.7=%0
.
IPR1.7=%0
.
PIR1.7=%0
.
```

El Timer 2 es el encargado de avisar al programa que es tiempo de realizar la lectura de los sensores inerciales, está configurado para que la interrupción se dé a una frecuencia de 5ms, es decir 200 veces por segundo, mientras que las medidas se toman cada que se cuenta dos interrupciones.

Es importante revisar todos los registros que se involucran para que la interrupción tenga efecto y asignarles el valor requerido según las condiciones de trabajo, caso contrario el programa nunca detectará la interrupción.

El programa tiene una secuencia infinita la cual es un bucle while, en este while se da el proceso de lectura y envío de datos llamando a los sub procesos antes mencionados.

```

    . 'Inicio del Loop del programa
    . While true
    .   if (UART1_Data_Ready() = 1) then
    .     secuencia2=UART1_Read()
    .   Select case secuencia2
    .     case 4
    .       enviar4=%0
    .       if enviar1=%1 then
    .         for cont1=0 to 451
    .           UART1_Write(datos1[cont1])
    .             if bandera_leer=%1 then
    .               bandera_leer=%0
    .               if leer1=%1 then
    .                 leer_sensores1
    .               end if
    .               if leer2=%1 then
    .                 leer_sensores2
    .               end if
    .               if leer3=%1 then
    .                 leer_sensores3
    .               end if
    .               if leer4=%1 then
    .                 leer_sensores4
    .               end if
    .             end if
    .           next cont1
    .         end if
    .       case 1
    .         enviar1=%0
    .         if enviar2=%1 then
    .           for cont1=0 to 451
    .             UART1_Write(datos2[cont1])
    .               if bandera_leer=%1 then
    .                 bandera_leer=%0
    .                 if leer1=%1 then
    .                   leer_sensores1
    .                 end if
    .                 if leer2=%1 then
    .                   leer_sensores2
    .                 end if
    .                 if leer3=%1 then
    .                   leer_sensores3
    .                 end if
    .                 if leer4=%1 then
    .                   leer_sensores4
    .                 end if
    .               end if
    .             next cont1
    .           end if
    .         case 2
    .           enviar2=%0
    .           if enviar3=%1 then
    .             for cont1=0 to 451
    .               UART1_Write(datos3[cont1])
    .                 if bandera_leer=%1 then
    .                   bandera_leer=%0
    .                   if leer1=%1 then
    .                     leer_sensores1
    .                   end if
    .                   if leer2=%1 then
    .                     leer_sensores2
    .                   end if
    .                   if leer3=%1 then
    .                     leer_sensores3
    .                   end if
    .                   if leer4=%1 then
    .                     leer_sensores4
    .                   end if
    .                 end if
    .               next cont1
    .             end if
    .           case 3
    .             enviar3=%0
    .             if enviar4=%1 then
    .               for cont1=0 to 451
    .                 UART1_Write(datos4[cont1])
    .                   if bandera_leer=%1 then
    .                     bandera_leer=%0
    .                     if leer1=%1 then
    .                       leer_sensores1
    .                     end if
    .                     if leer2=%1 then
    .                       leer_sensores2
    .                     end if
    .                     if leer3=%1 then
    .                       leer_sensores3
    .                     end if
    .                     if leer4=%1 then
    .                       leer_sensores4
    .                     end if
    .                   end if
    .                 next cont1
    .               end if
    .             end select
    .           end if

```

El Selector **Select Case** revisa cuál de las 4 matrices está lista para enviarse, para cualquiera de los casos inicia un contador para enviar los datos desde la posición 0 hasta la 451. Además en cada iteración revisa si la bandera que indica que se deben leer los datos se activa, si es el caso se dirige a los subprocesos para leer datos.

```

.    final:
.    if bandera_leer =%1 then
.        bandera_leer =%0
.        if leer1=%1 then
.            | leer_sensores1
.            end if
.            if leer2 =%1 then
.                | leer_sensores2
.                end if
.                if leer3 =%1 then
.                    | leer_sensores3
.                    end if
.                    if leer4 =%1 then
.                        | leer_sensores4
.                        end if
.                    end if
.                wend
.            end.
480
490

```

Como parte final el programa verificará si se debe leer los datos de los sensores inerciales aun cuando el computador no solicite esto, esto permitirá que la conexión con el computador se establezca en cualquier momento y los datos no estén desactualizados.

### **3.2 DISEÑO DEL SOFTWARE CARGADOR DE BATERÍA**

Cada módulo detector de movimiento tiene su propia fuente de alimentación conformada por 5 pilas AAA tipo Ni-Mh, esto quiere decir que tienen un ánodo de Niquel y un cátodo de Metal Hidruro.

Este tipo de baterías son recargables, de larga duración, no tiene efecto memoria y pueden cargarse en un tiempo de velocidad media.

Para diseñar un sistema de carga para este tipo de baterías se debe tener en cuenta aspectos importantes como:

- El circuito de carga debe controlar la corriente que pasa a través de la batería en todo momento.
- No se debe sobrepasar la cantidad de corriente especificada por el fabricante para carga o la celda se puede dañar permanentemente.
- Se debe reconocer a través de un control electrónico el final de carga del conjunto de baterías.

Los dos primeros puntos expuestos anteriormente se pueden controlar simplemente con un buen diseño del circuito.

El tercer punto debe ser controlado por software. En la gráfica de la figura 23 se puede observar la curva de carga de una batería NI-MH, al final del proceso de carga, el voltaje de esta empezará a disminuir, esa diferencia es la que se debe considerar para cargar este tipo de baterías, debido a que si se somete la batería a sobre carga se dañará permanentemente.

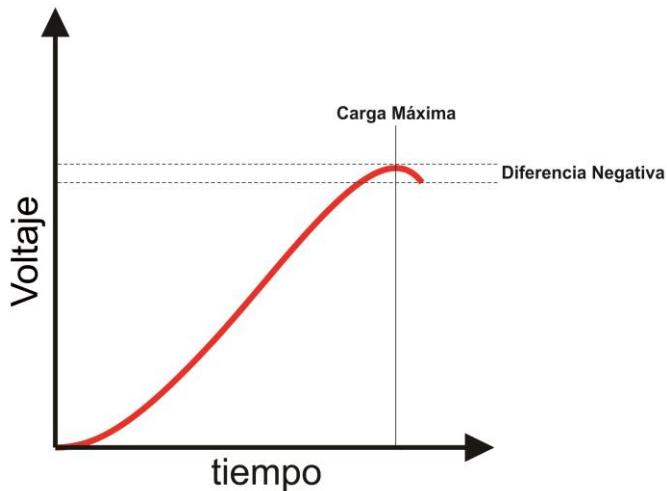


Figura 3.3: Curva de Carga de una batería AAA.

La manera de detectar esta caída de voltaje en la carga de la batería es usando un conversor analógico digital y comparar la lectura con el valor más alto de carga registrado.

El programa realiza esto para controlar el voltaje de 8 baterías en carga simultánea.

El programa entero se describe a continuación:

```

1 program Cargador
. .
. dim seg as byte[7]
. dim min as byte[7]
. dim hor as byte[7]
. dim volt as word[7]
. dim voltmax as word[7]
. dim conectado as byte[7]
. dim cont as byte
10 dim aux as word
. dim etapas as word[7]
. dim cargada as word[7]
. dim deltaV as integer
. dim voltaje as float[7]
. dim voltajemax as float[7]
. .

. main:
.     TRISA = 0xFF
.     TRISB = 0
.     TRISC = 0
.     TRISD = 0
.     TRISE = #111
.     ADC_Init()
.     ADCON1 = %00000000
.     PORTB=0
.     PORTC=0
.     PORTD=255
.     cont=0
.     aux=0
30    for cont=0 to 6
.         volt[cont]=0
.         voltmax[cont]=0
.         etapa[cont]=0
.         hor[cont]=0
.         min[cont]=0
.         seg[cont]=0
.     next cont

```

El programa se encarga de medir el voltaje de cada batería y controlar su etapa de carga, así como de finalizar la carga de forma individual de cada batería.

La variable *etapa* es un vector de 8 bytes el cual especifica la etapa de carga de cada batería. El resto de variables que conforman el programa también son vectores de 8 bytes.

```

.     delay_ms(500)
.     PORTD=0
40    while true
.         for cont=0 to 3
.             aux=ADC_Read(cont)
.             delay_ms(8)
.             if aux<100 then
.                 conectado[cont]=0
.                 seg[cont]=0
.                 min[cont]=0
.                 hor[cont]=0
.                 etapa[cont]=0
.                 PORTD.(cont)=0
.             else
.                 if etapa[cont]=0 then
.                     etapa[cont]=1
.                     PORTB.((cont*2)+1)=#1
.                 end if
.             end if
.
.             if etapa[cont]=1 then
60             if seg[cont]>30 then
.                 PORTB.((cont*2)+1)=#1
.                 etapa[cont]=2
.             end if
.         end if
.         if etapa[cont]=2 then
.             if aux < 350 then
.                 if voltmax[cont]=0 then
.                     voltmax[cont]=aux
.                 end if
.                 if aux < voltmax[cont] then
.                     voltmax[cont]=aux
.                 end if
.                 if aux > voltmax[cont] then
.                     if aux-voltmax[cont] > 7 then
.                         etapa[cont]=3
.                     end if
.                 end if
.             end if
.         end if
.         if etapa[cont]=3 then
.             PORTB.((cont*2)+2)=#0
.             PORTB.((cont*2)+1)=#0
.         end if
.
.         if etapa[cont]>0 then
.             seg[cont] = seg[cont]+1
.         end if
.         if seg[cont]=120 then
90         seg[cont]=0
.             min[cont]=min[cont]+1
.         end if
.         if min[cont]=120 then
.             min[cont]=0
.             hor[cont]=hor[cont]+1
.         end if
.         if etapa[cont]=1 then
.             PORTD.(cont) = not PORTD.(cont)
.         end if
.         if etapa[cont]=2 then
.             PORTD.(cont) = not PORTD.(cont)
.         end if
.         if etapa[cont]=3 then
.             PORTD.(cont) = #1
.         end if
.     next cont
.     for cont=0 to 3
.         aux=ADC_Read(cont)
.         delay_ms(8)
.         if aux<100 then
.             conectado[cont]=0
.
```

```

        seg[cont]=0
        min[cont]=0
        hor[cont]=0
        etapa[cont]=0
        PORTD.(cont+4)=0
    else
        if etapa[cont]=0 then
            etapa[cont]=1
            PORTC.(cont*2)=\$1
        end if
    end if

    if etapa[cont]=1 then
        if seg[cont]>30 then
            PORTC.((cont*2)+1)=\$1
            etapa[cont]=2
        end if
    end if
130 if etapa[cont]=2 then
    if aux < 350 then
        if voltmax[cont]=0 then
            voltmax[cont]=aux
        end if
        if aux < voltmax[cont] then
            voltmax[cont]=aux
        end if
        if aux > voltmax[cont] then
            if aux-voltmax[cont] > 7 then
                etapa[cont]=3
            end if
        end if
    end if
end if
if etapa[cont]=3 then
    PORTC.(cont*2)=\$0
    PORTC.((cont*2)+1)=\$0
end if

if etapa[cont]=3 then
    PORTC.(cont*2)=\$0
    PORTC.((cont*2)+1)=\$0
end if

150 if etapa[cont]>0 then
    seg[cont] = seg[cont]+1
end if
if seg[cont]=120 then
    seg[cont]=0
    min[cont]=min[cont]+1
end if
if min[cont]=120 then
    min[cont]=0
    hor[cont]=hor[cont]+1
end if
if etapa[cont]=1 then
    PORTD.(cont+4) = not PORTD.(cont+4)
end if
if etapa[cont]=2 then
    PORTD.(cont+4) = not PORTD.(cont+4)
end if
if etapa[cont]=3 then
    PORTD.(cont+4) = \$1
end if
170 next cont
.

if etapa[cont]=1 then
    delay_ms(500)
else
    delay_ms(250)
end if
.
wend
end.

```

De esta manera usando un micro controlador PIC16F877A se puede cargar 8 baterías con 5 pilas AAA cada una simultáneamente.

### 3.3 DISEÑO DEL SOFTWARE DEL PC

El programa para el PC se realiza sobre el compilador LabView, este programa se encarga del procesamiento de los datos adquiridos por los módulos detectores de movimiento y transformarlos en una representación visual y estadística de los movimientos del usuario del sistema.

El diagrama de flujo del programa del PC se muestra en la figura 24.

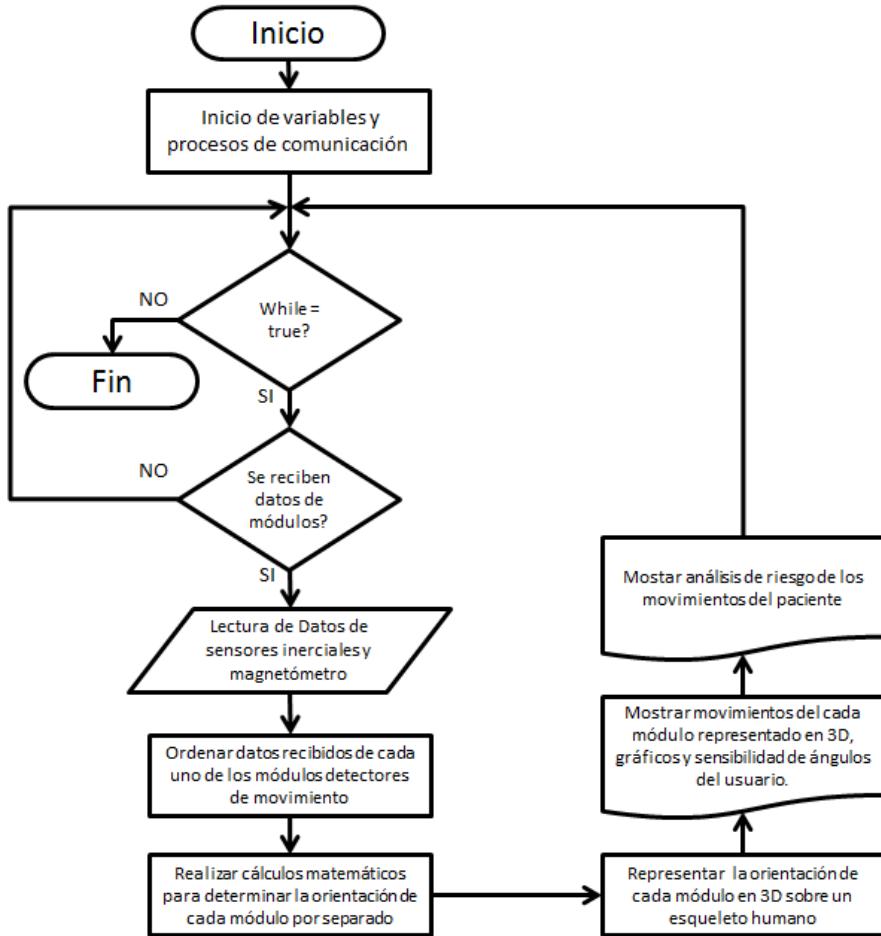


Figura 3.4: Diagrama de Flujo de Software de PC.

El PC recibe los datos provenientes de los módulos detectores de movimiento y procesa los datos hasta convertirlos en indicadores visuales entendibles para el usuario.

El bloque *While* de la figura 3.4, se realiza 100 veces por segundo para cada uno de los 7 módulos detectores de movimiento.

El programa se irá analizando parte por parte al igual que en apartados anteriores.

### 3.3.1 RECEPCIÓN DE DATOS

La recepción de datos de los 7 módulos detectores de movimiento se la realiza a través de protocolo TCP, cada uno de los módulos es un servidor TCP, además tiene una dirección IP única.

Los datos recibidos pertenecen a los sensores iniciales, cada módulo detector de movimiento envía 100 muestras por segundo de sus 3 sensores: acelerómetros, giroscopio y magnetómetro; cada uno de estos tiene 3 ejes, es decir cada módulo envía  $3(\text{sensores}) \times 3(\text{ejes}) \times 2 (\text{registros de 8bits}) = 18 \text{ bytes} \times 100 (\text{veces por segundo}) = 1800 \text{ Kb/segundo}$  por cada módulo detector de movimiento.

Los módulos detectores de movimiento detectan velocidad angular en cada uno de sus ejes, aceleración lineal en cada uno de sus ejes e intensidad de campo magnético en cada uno de sus ejes.

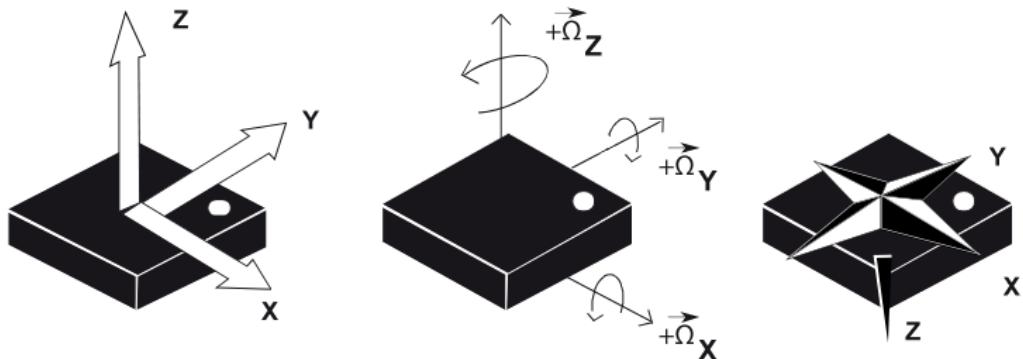


Figura 3.5: Magnitudes medidas en cada módulo detector de movimiento.

La figura 3.5 muestra cómo se miden cada una de las magnitudes antes descritas en cada módulo detector de movimiento las cuales se reciben en el PC con el siguiente bloque de programación en LabView:

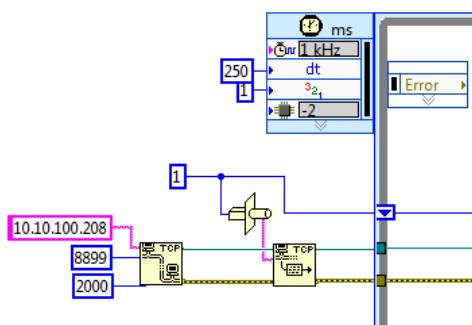


Figura 3.6: Abrir comunicación con servidor TCP remoto y bloque *While* temporizado.

En la figura 3.6 se abre la comunicación con el servidor TCP ubicado en cada uno de los módulos detectores de movimiento, seguido se envía el byte “1”, lo que indica la petición de envío de la primera de cuatro tramas de 452 bytes que se recibirá por segundo, los 2 últimos bytes de esta trama contienen información para comprobar la correcta secuencia de recepción de datos.

El lazo de programación utilizado es un *While* temporizado, se ha programado cada iteración con duración de 250ms con prioridad 1, de tal manera que se ejecute exactamente 4 veces por segundo.

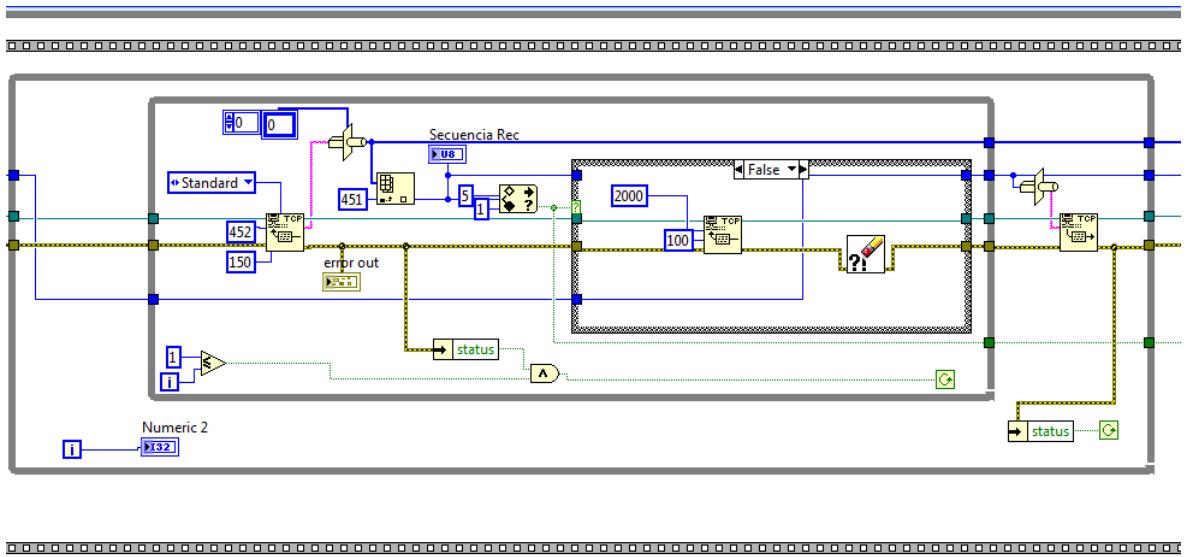


Figura 3.7: Bloque de recepción de datos.

La figura 3.7 muestra 2 bloques *While* funcionando uno dentro de otro. El más pequeño se encarga de recibir las tramas provenientes de los sensores. El programa esperará 150ms hasta recibir las tramas de 452 bytes, en caso de no recibir o recibir en desorden o incompleta alguna trama no procederá a guardar y reenviará la petición al módulo para que reenvíe los datos de forma correcta. En caso de no recibir los datos correctos en más de 2 intentos se declara perdida la trama y se solicita la siguiente de la secuencia.

Se ha registrado una pérdida de datos muy baja, menos del **0.05%** de los datos registrados en una hora de funcionamiento del sistema.

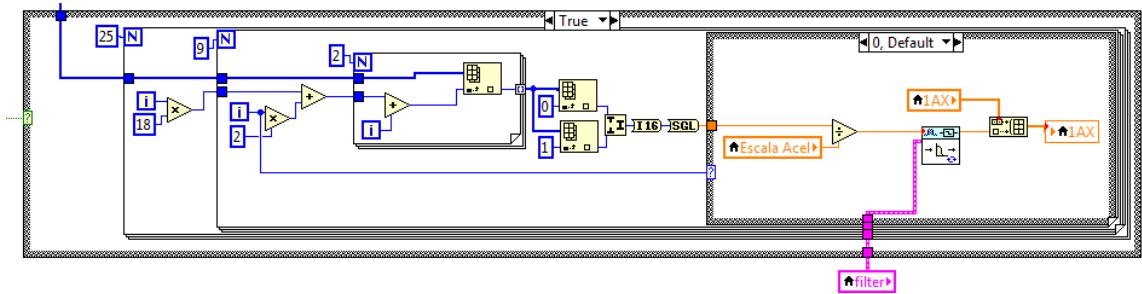


Figura 3.8: Almacenamiento de datos recibidos.

En la figura 3.8 se muestra el bloque de programa ejecutado si se recibe correctamente los datos. La sentencia **Case** que engloba todo el bloque de programa que almacena los datos de cada módulo detector de movimiento si la recepción de la trama es correcta en el bloque de la figura 3.7, será correcta si la secuencia de recepción es la que corresponde.

Los bloques **FOR** se encargan de desarmar cada una de las tramas recibidas y ordenarla en variables específicas para cada sensor y cada eje.

Existen 33 variables para cada módulo de movimiento, cada variable son vectores que pueden tener extensión de hasta 2000 registros, la mayoría son registros de 32bits, de precisión simple.

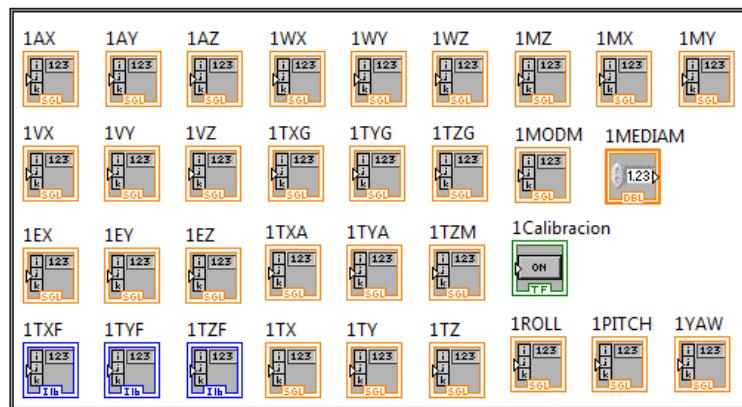


Figura 3.9: Variables usadas por cada módulo detector de movimiento.

Como se puede apreciar existen 31 vectores, 28 de ellos son de precisión simple y los 3 restantes enteros con signo de 16 bits. Los datos se guardan sobre los vectores de la fila superior. Cada segundo se almacenan 100 nuevos datos en todos los vectores de la figura 3.9.

Si se decide se puede finalizar la comunicación con uno de los módulos usando el bloque de la figura 3.10.

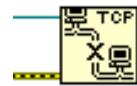


Figura 3.10: Bloque para cerrar conexiones TCP.

La comunicación solo se abre y cierra una vez, no se cierra entre iteraciones para agilizar la recepción de los datos.

Los datos recibidos y almacenados en las 9 variables mostradas en la parte superior de la figura 3.9 están listos para ser procesados para encontrar los datos de 22 vectores restantes.

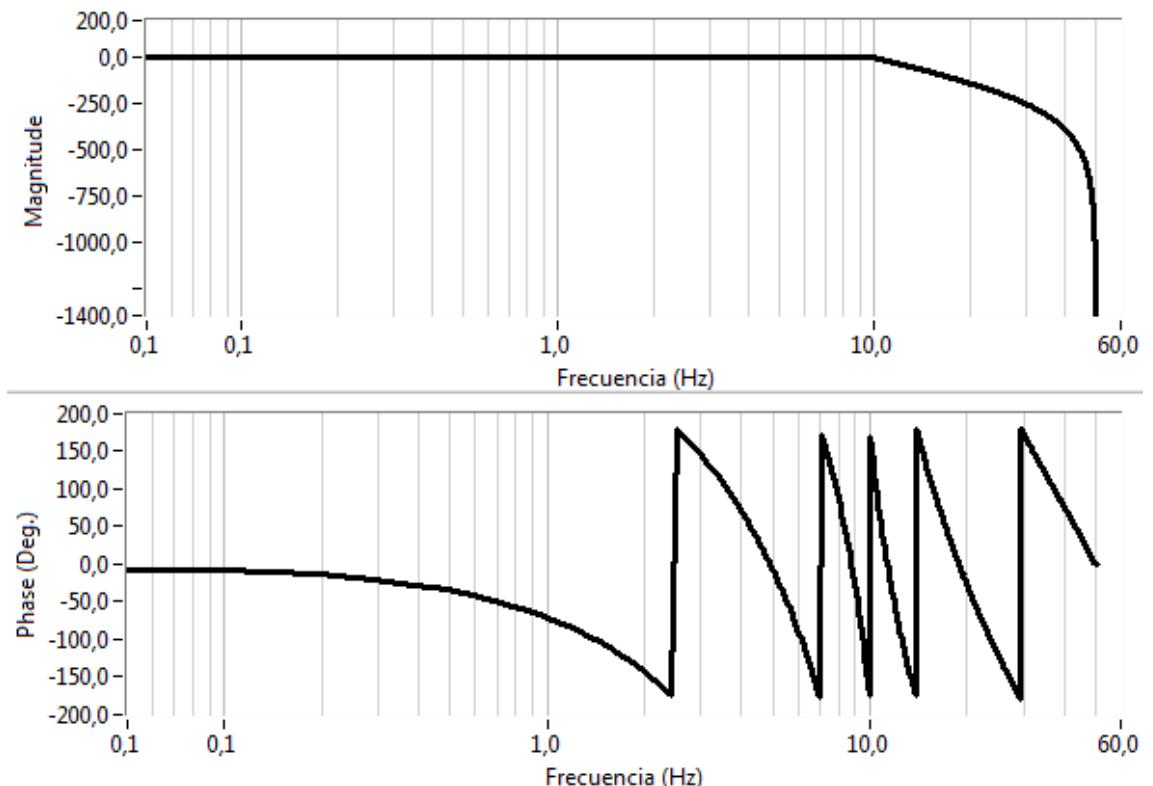
### 3.3.2 PROCESAMIENTO DE DATOS RECIBIDOS

El procesamiento de los datos recibidos se realiza con cálculos matemáticos para convertir los datos recibidos de los sensores en magnitudes físicas que pueda leerse en informes y gráficos de realidad aumentada por el usuario.

#### 3.3.2.1 FILTRADO DE DATOS RECIBIDOS

El procesamiento comienza con un filtrado de los datos recibidos de manera que se tenga la señal más estable posible.

El filtro se lo realiza basándose en criterios lógicos para estabilizar la señal de la mejor manera posible, primero se debe saber que ningún movimiento de una persona podrá superar una frecuencia de 10Hz, es decir jamás se podrá mover una parte del cuerpo 10 de arriba abajo 10 veces por segundo.



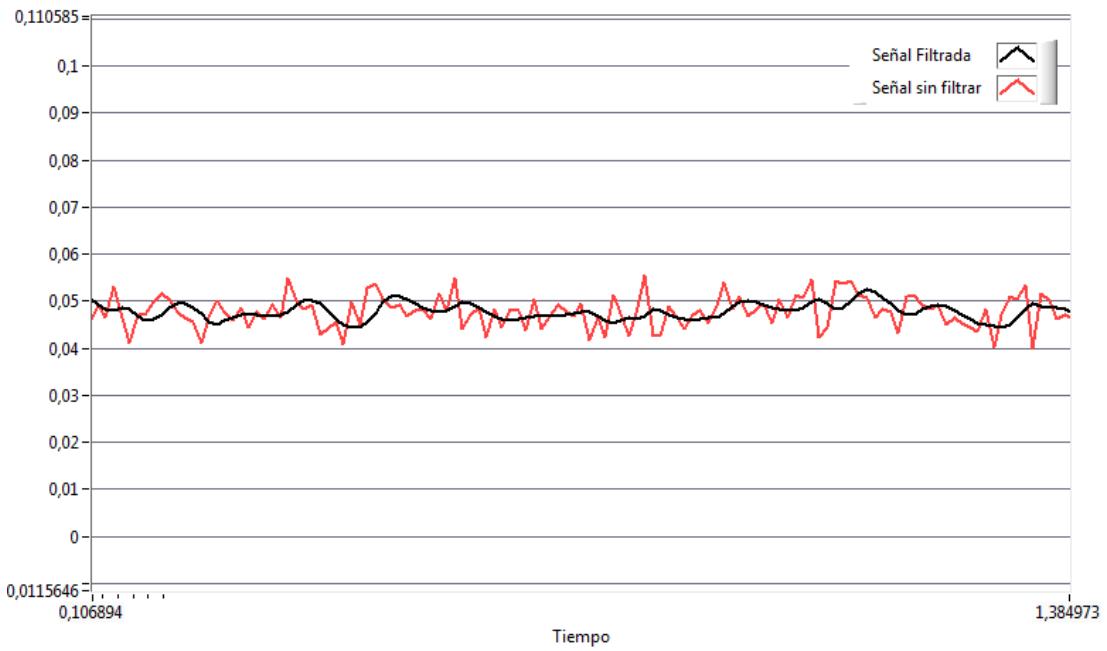
**Figura 3.11: Respuesta en frecuencia del filtro Butterworth diseñado.**

EL filtro usado es un filtro digital Butterworth de 20 polos, el cual tiene una respuesta muy buena en frecuencia para el tipo de señales que se manejan.

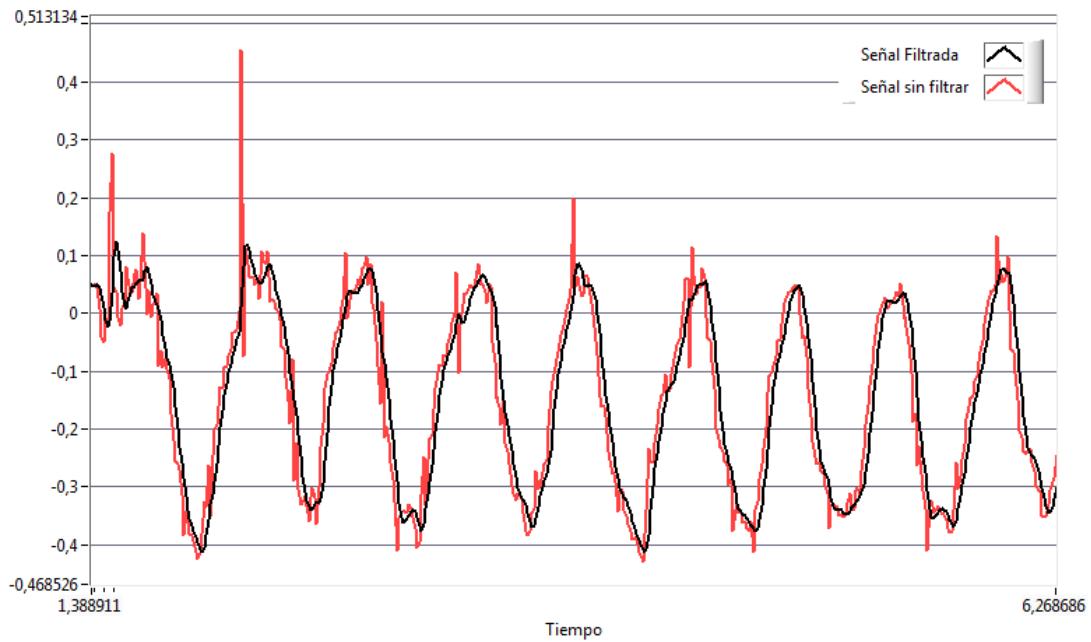
Se puede ver la mejora de la estabilidad en condiciones de estática y movimiento con el uso del filtro.

La figura 3.12 muestra en tiempo la salida de filtro Butterworth diseñado, como se puede apreciar la señal es mucho más estable y su desviación estándar se reduce a simple vista, de 0.0045 a 0.0021.

La figura 3.13 muestra en tiempo la salida del filtro Butterworth diseñado, pero esta vez en condición de movimiento, como se puede observar, mejora tremadamente las inestables respuestas a cambios de aceleración del acelerómetro, básicamente se eliminan picos causados por la inercia del movimiento.



**Figura 3.12: Respuesta en tiempo de eje X de aceleración del filtro Butterworth en condición estática.**



**Figura 3.13: Respuesta en tiempo de eje X de aceleración del filtro Butterworth en condiciones de movimiento. Magnitud representada en escala 1g.**

El mismo filtro se usa en todos los ejes del acelerómetro y magnetómetro, señales que tienen un grave problema de inestabilidad por causas de bias y ruido blanco gaussiano. La eliminación de picos fantasma causados por la inercia evita tener movimientos inexistentes a la salida visual para el usuario que pueden ser hasta 4 veces más grandes que la medida real.

El bloque de programa del diseño del filtro se lo puede ver en la figura 3.14, es un filtro IIR Butterworth de 20 polos con frecuencia de corte de 10Hz y frecuencia de muestreo de 100Hz.

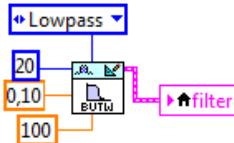


Figura 3.14: Construcción del Filtro IIR Butterworth,  $F_c=10\text{Hz}$ ,  $F_s=100\text{Hz}$ .

El valor del diseño del filtro se guarda en la variable filter para aplicar en todas las señales a filtrar. El filtro se aplica en el bloque de programa mostrado en la figura 3.15.

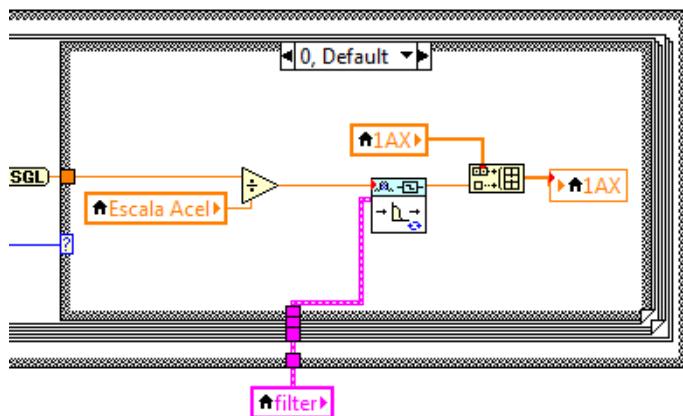


Figura 3.14: Aplicación de filtro Butterworth a señales recibidas.

### 3.3.2.2 CÁLCULO DE ÁNGULO DE GIRO A PARTIR DE GIROSCÓPIO

El giroscopio como ya se vio en capítulos anteriores puede medir la velocidad angular sobre cada uno de sus tres ejes de forma independiente.

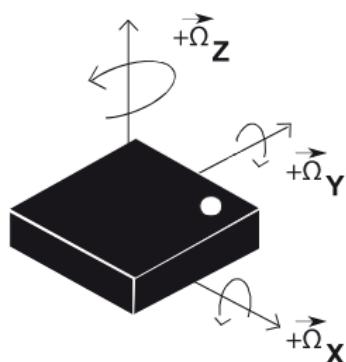


Figura 3.15: Medición de velocidad angular cuantificada por giroscopio de 3 ejes.

Ya que se tiene el valor de la velocidad angular por cada eje del módulo detector de movimiento, se puede calcular el ángulo de desplazamiento también, únicamente se debe aplicar los principios básicos del movimiento uniformemente variado.

El cálculo del ángulo de giro se realiza basándose en la siguiente fórmula:

$$\theta = \theta_0 + \omega_0 t + \frac{\alpha t^2}{2} \quad (3.1)$$

Dónde:

$\theta$  es el ángulo de giro de cualquiera de los ejes.

$\theta_0$  es el ángulo previo de giro del eje que se está calculando.

$\omega_0$  es la velocidad angular medida por el giroscopio.

$t$  es el tiempo entre cada muestra, para nuestro caso es  $\approx 0.01$  segundos.

$\alpha$  es la aceleración angular.

El parámetro  $\alpha$  puede ser calculado usando la fórmula:

$$\alpha = \frac{\omega - \omega_0}{t} \quad (3.2)$$

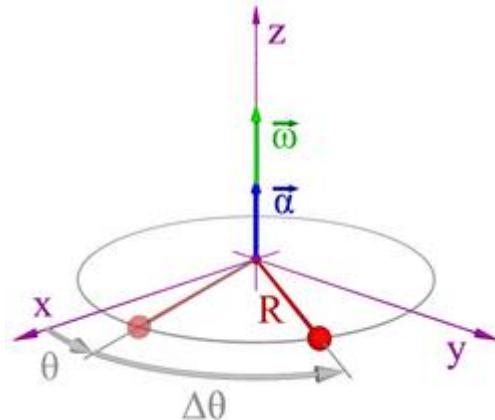


Figura 3.16: Representación gráfica de cálculo de  $\theta_z$ .

El bloque de programa donde se realiza este cálculo se puede ver en la figura 3.17, se puede ver que el ángulo calculado se suma al anterior para obtener la orientación sobre los tres ejes en todo momento, además los datos calculados se representan en valores que van desde:  $-180^\circ < \theta \leq 180^\circ$ .

Además para evitar que los errores por bias y ruido blanco gaussiano creen incrementos que no existen los datos menores a la desviación estándar en modo estático se desprecian.

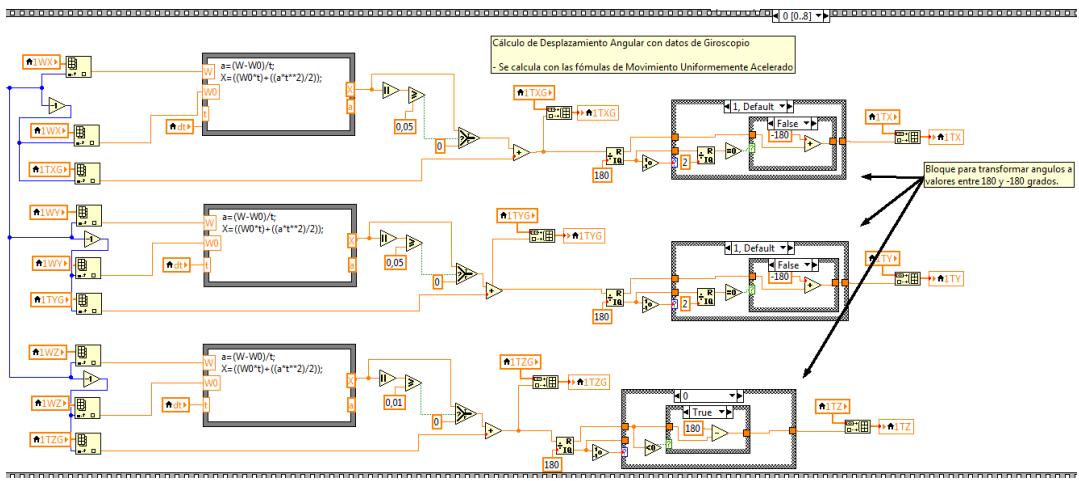


Figura 3.17: Medición de velocidad angular cuantificada por giroscopio de 3 ejes.

Sin embargo, este ángulo calculado únicamente con el giroscopio no tiene buena precisión en períodos largos de tiempo y al ruido blanco gaussiano los errores se acumulan de manera muy rápida. Además está el *drift*, que es una característica negativa de los giroscopios, un fenómeno físico causado por la inercia del movimiento.

Por eso es necesario basarse en datos de otros sensores para tener siempre un valor real de orientación a través de la combinación de información.

### 3.3.2.3 CÁLCULO DE ÁNGULO DE GIRO A PARTIR DE ACELERÓMETRO

Es necesario complementar los datos del giroscopio con datos que no dependan de un eje arbitrario sino se ubiquen en el espacio utilizado magnitudes omnipresentes como son el campo magnético terrestre o la gravedad.

El cálculo del *Roll* y *Pitch* a través del acelerómetro y el vector de gravedad provee de un valor real en cualquier instante de tiempo, siempre y cuando el módulo detector de movimiento este en estado estático en cualquier orientación.

Una vez calculado este valor servirá como herramienta para corregir los errores que se obtienen en los datos calculados por el giroscopio.

Es decir se usarán los datos del giroscopio en períodos de tiempo cortos hasta corregir el error que pueda existir con los datos calculados a través del acelerómetro, datos reales basados en una constante planetaria como es la gravedad en la Tierra.

El cálculo del *Pitch* y *Roll* utilizando el vector de gravedad se lo realiza con la fórmula 3.3 y 3.4.

$$Roll = \sin\left(\frac{A_y}{\sqrt{A_x^2 + A_z^2}}\right) \quad (3.3)$$

$$Pitch = \sin\left(\frac{A_z}{\sqrt{A_x^2 + A_y^2}}\right) \quad (3.4)$$

El diagrama de cuerpo libre utilizado para obtener la ecuación 3.3 se puede apreciar en la figura 3.18. El ángulo  $\theta$  representa al *Roll* del módulo detector de movimiento.

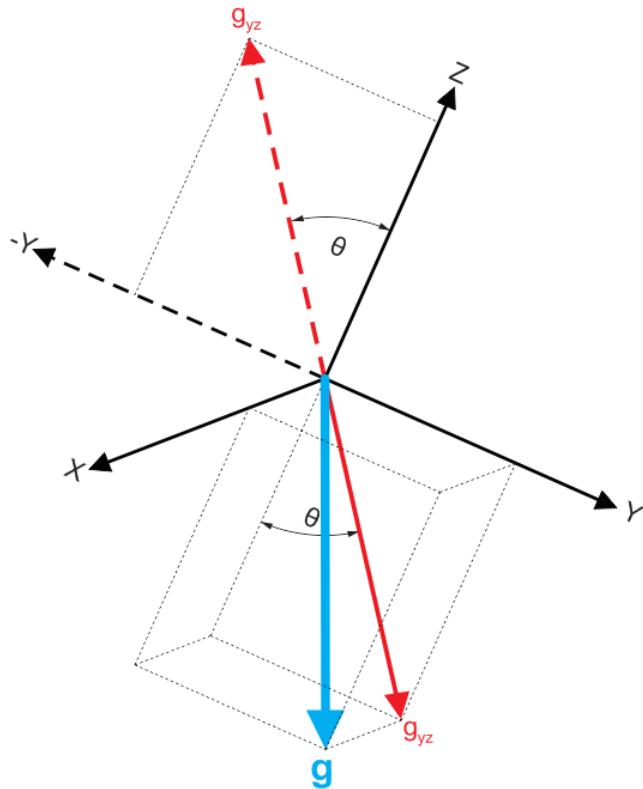


Figura 3.18: Diagrama de cuerpo libre para cálculo de *Roll* o giro sobre eje X.

El cálculo del *Yaw* no se lo puede realizar mediante este método ya que el eje Z será casi colineal con el vector de gravedad causando inestabilidad en el cálculo.

El bloque de programa donde se realiza este cálculo se lo muestra en la figura 3.19, además se utilizan estos dos datos calculados para un futuro cálculo.

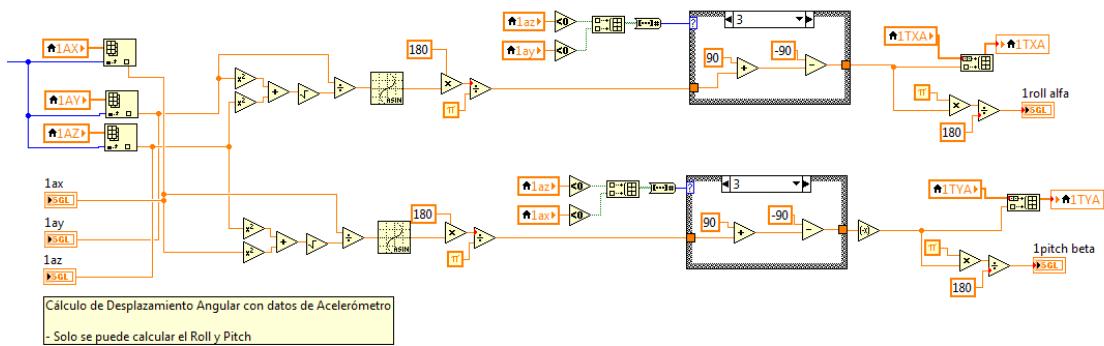


Figura 3.19: Bloque de programa de cálculo de *Roll* y *Pitch* usando datos de acelerómetro.

### 3.3.2.4 CÁLCULO DE COMPENSACIÓN DE INCLINACIÓN

Para poder calcular la orientación de cada módulo detector de movimiento es necesario ubicar el norte magnético, de manera que todos los sensores se ubiquen bajo el mismo sistema de referencia. Ya se ha logrado calcular el *Roll* y *Pitch* en un sistema de referencia real, ahora se deberá usar el campo magnético de la Tierra para poder ubicar el *Yaw* de todos los módulos en el mismo sistema real de referencia.

Este cálculo no se lo puede hacer directamente, debido a que el módulo no permanece normalmente paralelo al campo magnético de la Tierra, es decir este puede atravesar al magnetómetro del módulo en direcciones arbitrarias que no permiten establecer una fórmula que calcule directamente este ángulo con respecto al norte magnético usando simplemente las componentes de intensidad magnética.

Para obtener la correcta orientación se deberá utilizar proyecciones de las componentes magnéticas sobre la horizontal, de manera que siempre se muestre la dirección a la que apunte el vector de campo magnético de la Tierra.

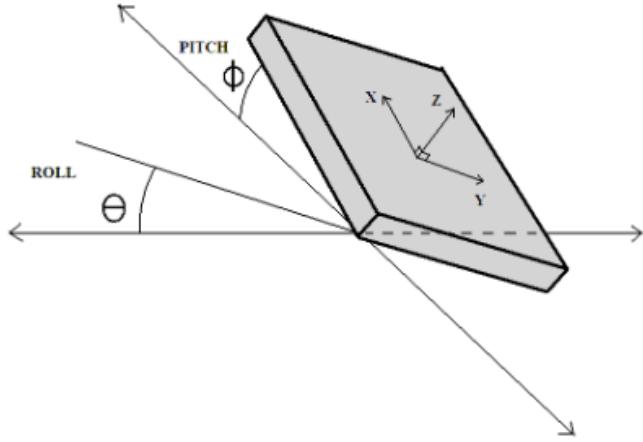


Figura 3.20: Conversión de los ángulos para compensación de inclinación.

Las proyecciones del campo magnético en la horizontal se forman al aplicar 2 secuencias de cálculos, primero se calcula la compensación de inclinación del *Roll*, es decir con un giro sobre el eje X. Expresado en las ecuaciones 3.5.

$$\begin{aligned} B_{x_{roll}} &= B_x \\ B_{y_{roll}} &= B_y \cos \theta + B_z \sin \theta \\ B_{z_{roll}} &= -B_y \sin \theta + B_z \cos \theta \end{aligned} \quad (3.5)$$

Y las ecuaciones para calcular la compensación sobre el ángulo *Pitch*.

$$\begin{aligned} B'_x &= B_{x_{roll}} \cos \Phi + B_z \sin \Phi \\ B'_y &= B_{y_{roll}} \\ B'_z &= B_{z_{roll}} \sin \Phi + B_{x_{roll}} \cos \Phi \end{aligned} \quad (3.6)$$

Reemplazando las ecuaciones 3.5 en 3.6 se obtienen los componentes de compensación sobre la horizontal:

$$B'_x = B_x \cos \Phi + B_y \sin \Phi \sin \theta + B_z \sin \Phi \cos \theta \quad (3.7)$$

$$B'_y = B_y \cos \theta + B_z \sin \theta \quad (3.8)$$

$$B'_z = B_x \sin \Phi - B_y \sin \theta \cos \Phi + B_z \cos \theta \cos \Phi \quad (3.9)$$

De esta manera, se procede a calcular el ángulo entre estas componentes mediante bloques de programación y el arco tangente de su división, como se muestra en la figura 3.21.

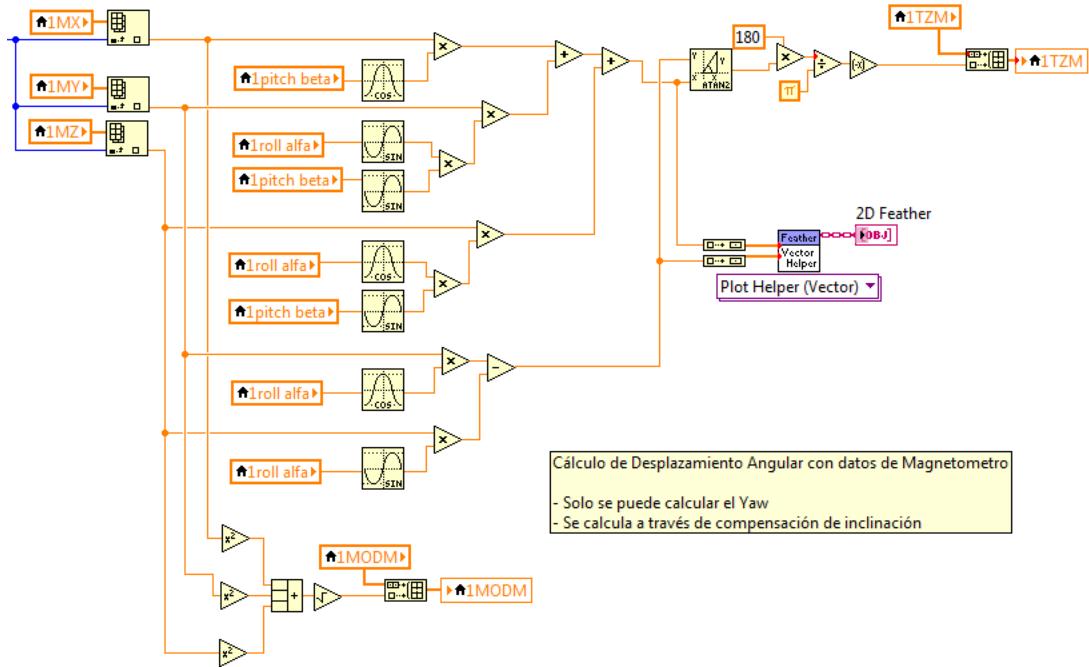


Figura 3.21: Compensación de inclinación, bloque de programa.

En esta parte del programa también se realiza un registro de la amplitud del campo magnético para evitar futuras interferencias que puedan existir y alteren la el valor de la orientación.

### 3.3.2.5 CÁLCULO DE ROLL EN EL SISTEMA COORDENADO DE NAVEGACIÓN

Hasta ahora los cálculos de ángulos basados en datos del acelerómetro si bien tienen valores basados en referencias generales como son el vector de gravedad o como de campo magnético terrestre, existe un problema con los valores que fueron medidos por el giroscopio, estos están en relación al sistema coordenado del módulo detector de movimiento y no en relación a un sistema coordenado general que lo llamaremos *sistema coordinado de navegación* y estará representado por los ejes *XYZ*, el objetivo es obtener todas las mediciones en todos los sensores en relación a un solo marco de referencia *XYZ*, en este caso se comenzará con el cálculo del ángulo *Roll* del sistema coordinado de navegación.

El sistema de navegación mide el *Roll* como lo hacen varios sistemas de navegación utilizados para la orientación, en específico se los llama ángulos de Tait Bryan.

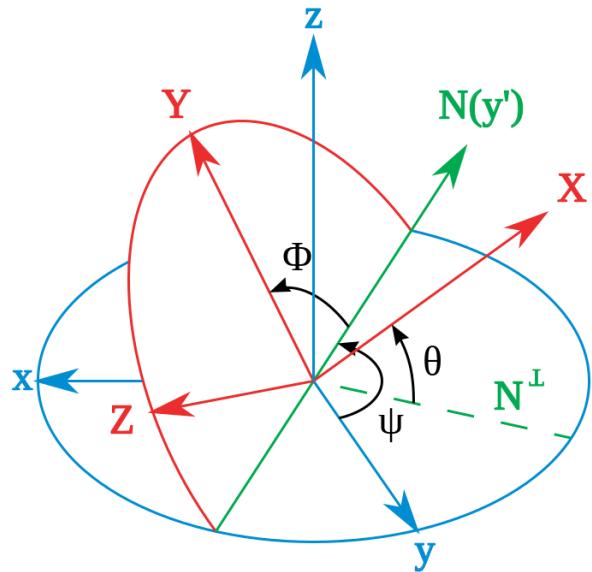


Figura 3.22: Secuencia de ángulos de Tait Bryan. Fuente: [18].

Esta secuencia de ángulos es capaz de representar la orientación en el espacio de cualquier objeto y es usado en la navegación, sobre todo para aéreo naves. En la figura 3.22 se puede observar claramente 3 ángulos,  $\theta$  es el *Roll*,  $\psi$  es el *Yaw* y  $\Phi$  es el *Pitch*, los cuales deben calcularse en ese orden.

Una muestra menos abstracta de cómo es que estos ángulos pueden proporcionar datos de orientación de un objeto es la figura 3.23, donde se muestra como una aéreo nave utiliza estos ángulos para obtener su orientación que sumados al GPS de la nave proveen un sistema completo de localización en el espacio de cada una de sus partes.

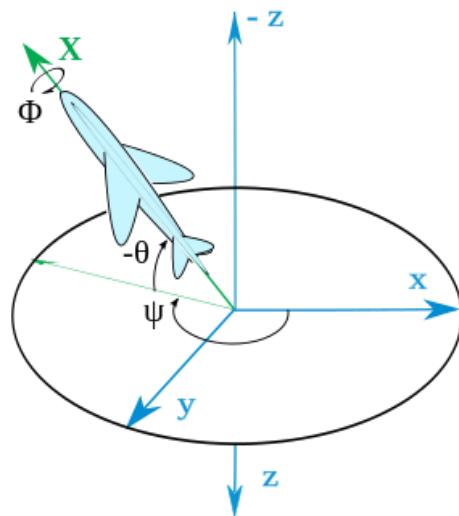
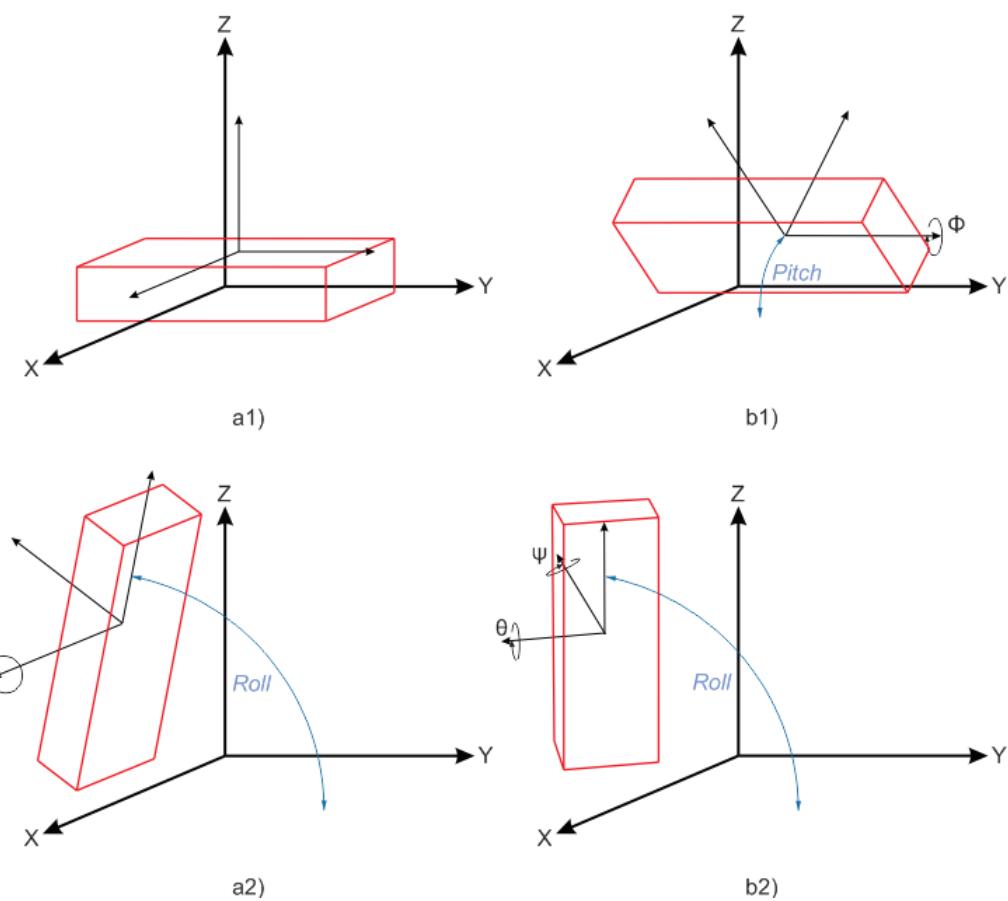


Figura 3.23: Orientación de una aéreo nave con ángulos de Tait Bryan en Sistema Coordenado de Navegación.

Fuente: [18].

Para calcular estos ángulos basados en un sistema absoluto como es el sistema coordenado **XYZ**, se debe saber que al momento en que los ejes del módulo detector de movimiento estén desalineados de los ejes de referencia global, los cálculos de los ángulos del giroscopio serán errados, ya que no es lo mismo calcular el *Roll* de navegación usando datos del giroscopio con el eje X paralelo a la horizontal global es decir con un ángulo *Pitch* de  $0^\circ$ , que hacerlo con un *Pitch* de diferente a  $0^\circ$ . Obsérvese la figura 3.24 para verificar gráficamente que le incremento de estos ángulos de dos formas diferentes puede producir diferencias de orientación en el sistema global.



**Figura 3.24: Diferencia de cálculo de ángulos de navegación en sistemas no coplanares.**

Para incrementar el ángulo *Roll* del módulo detector de movimiento con relación al sistema de coordenado de navegación, se debe tener en cuenta la orientación inicial de los sensores. En la figura 3.24, se pude apreciar la a a1) la cual inicia con sus ejes paralelos a los ejes **XYZ**. En la figura a2) se muestra un incremento del ángulo *Roll* de navegación el cual se incrementa en la misma intensidad que el ángulo  $\theta$  o *Roll* del

módulo detector de movimiento debido a su condición inicial de ejes paralelos a los ejes **XYZ**.

En el caso de la figura *b1*), el objeto inicia con los ejes no paralelos a los ejes **XYZ**, ya que se observa que se ha producido un giro sobre el eje *y* correspondiente al ángulo  $\Phi$  o *Pitch* del módulo detector de movimiento. Dada la condición inicial *b1*), se puede ver en *b2*) un incremento del ángulo *Roll* en el sistema coordenado de navegación o **XYZ**, para conseguir este incremento el módulo detector de movimiento con ejes no paralelos a **XYZ** debe realizar giros sobre dos de sus ángulos, en este caso el ángulo  $\theta$  o *Roll* del módulo detector de movimiento y sobre  $\Psi$  o *Yaw* del módulo detector de movimiento. La relación entre estos dos ángulos y el ángulo de navegación *Roll* del sistema coordenado **XYZ** se expresan en la ecuación 3.10.

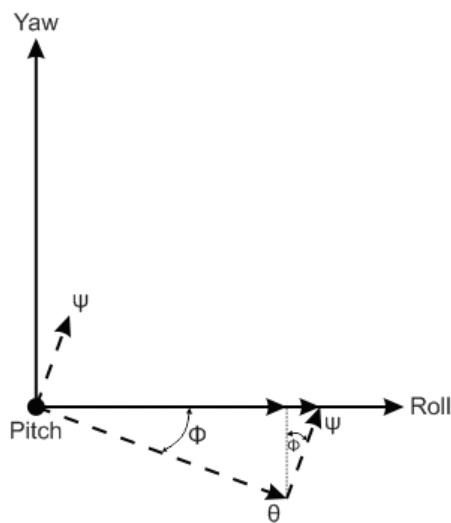


Figura 3.25: Diagrama de cálculo de ángulo *Roll* de navegación.

Según la figura 3.25, la ecuación que relaciona los ángulos  $\theta$  y  $\Psi$  del sistema coordinado del módulo detector de movimiento con el ángulo *Roll* de navegación, el cálculo depende directamente del ángulo  $\Phi$  del sistema coordinado del módulo detector de movimiento.

Se considera a todos los ángulos como escalares, excepto el ángulo  $\Phi$ , luego se calcula como se tratase de obtener la hipotenusa de un triángulo rectángulo dividido en dos triángulos rectángulos.

$$Roll = \theta \cos \phi + \Psi \sin \phi \quad (3.10)$$

Aplicando la ecuación 3.10, el bloque de programación queda como se muestra a continuación.

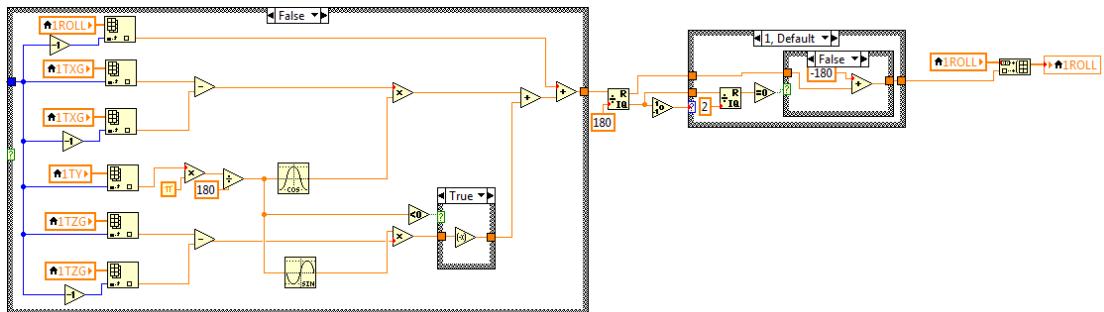


Figura 3.26: Bloque de programación para cálculo de ángulo *Roll* del sistema coordenado de navegación.

En la figura también se muestra la lógica utilizada para mantener el valor de *Roll* en el rango de  $-180^\circ < \text{Roll} \leq 180^\circ$ .

### 3.3.2.6 CÁLCULO DE YAW EN EL SISTEMA COORDENADO DE NAVEGACIÓN

De igual manera que en el apartado 3.3.2.5, es necesario realizar las mismas consideraciones para el cálculo del *Yaw* del sistema coordinado de navegación. Nuevamente el valor de este ángulo depende del ángulo  $\theta$  y  $\Psi$  actuando como valores escalares, mientras que  $\Phi$  si será representado como ángulo.

Para este caso se tiene otra interpretación de la figura 3.25, la figura 3.27 muestra la manera de relacionar las mismas variables utilizadas anteriormente para calcular el ángulo *Yaw* del sistema coordinado de navegación.

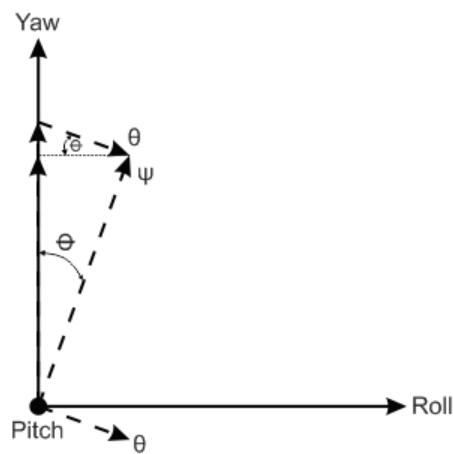


Figura 3.27: Diagrama de cálculo de ángulo *Yaw* de navegación.

$$Yaw = \Psi \cos \phi + \theta \sin \phi \quad (3.11)$$

La ecuación 3.11 representa la manera de representar el valor del ángulo *Yaw* del sistema coordenado de navegación.

### 3.3.2.7 CORRECCIÓN DE *PITCH* Y *ROLL* A TRAVÉS DE DATOS DEL ACELERÓMETRO

Como ya se ha dicho anteriormente, no se puede confiar en los valores del giroscopio por períodos prolongados de tiempo, sobre todo cuando existen movimientos bruscos en varios sentidos ya que este tipo de sensores son afectados por la inercia de su masa produciendo *drift* (giro sobre sus propios ejes debido a la inercia generada).

Según las ecuaciones 3.3 y 3.4 del inciso 3.3.2.3 es posible calcular el valor del *Roll* y *Pitch* del sistema de coordenadas de navegación basándose en la orientación módulo detector de movimiento en comparación con el vector de gravedad.

Esta corrección solo se puede realizar con el módulo en estado de reposo, es por eso que el programa analizará 100 veces por segundo el instante en que el módulo se encuentre en la posición óptima para realizar la corrección.

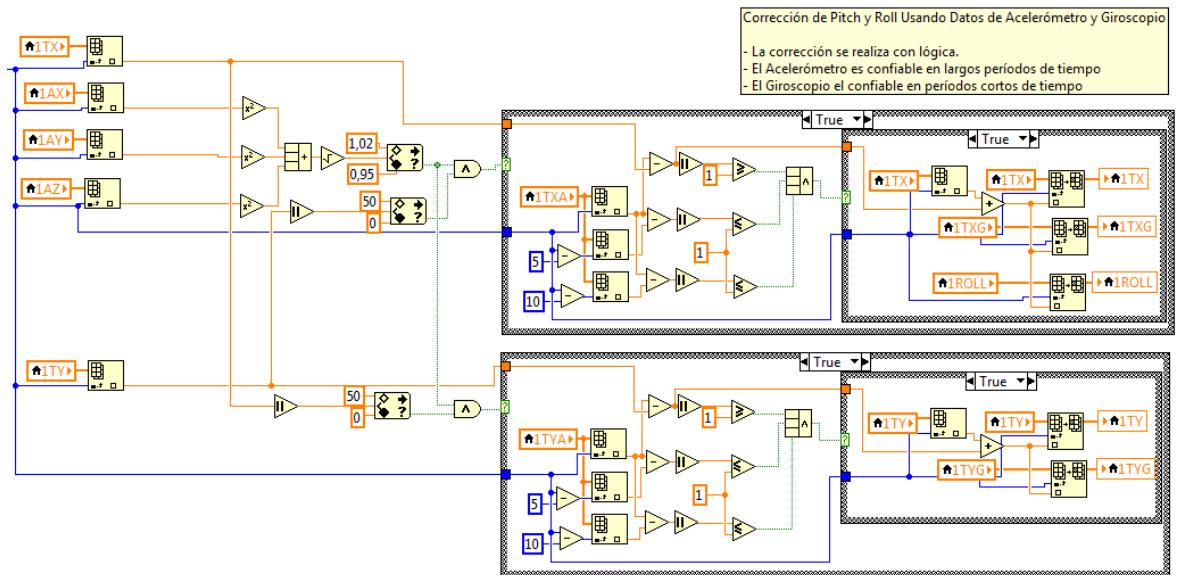
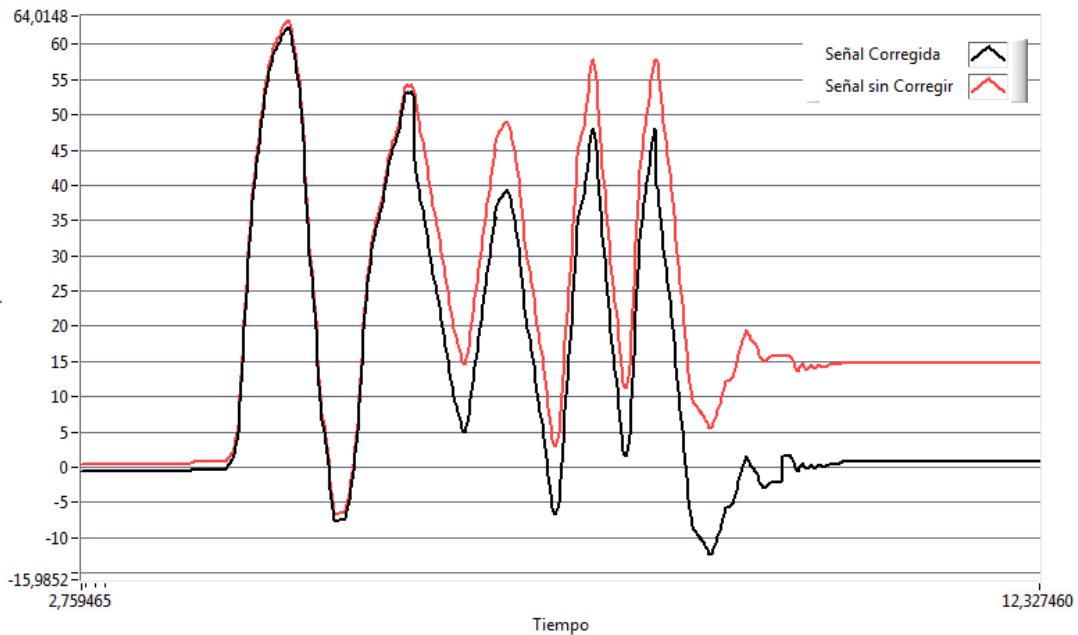


Figura 3.28: Bloque de programación para corrección de *Pitch* y *Roll*.

El bloque de programación se muestra en la figura 3.28, el programa realizará la corrección solamente bajo ciertas condiciones, como la condición de estática del

módulo detector de movimiento, tener por lo menos 200ms de una señal del acelerómetro estable y que la diferencia se mayor a 1 grado.

Como se puede ver, se toma como estado de reposo a un valor de gravedad de entre 0.95g y 1.02g.



**Figura 3.29: Señal con corrección de error vs Señal sin corrección de error.**

En la figura 3.29, se realiza la corrección del error generado por el drift del giroscopio a la señal de giro sobre el eje X. Al cabo de alrededor de 8 segundos el valor real puede diferir en 15 grados tras 5 oscilaciones del ángulo con alrededor de 55 grados de diferencia pico-pico con un regreso a la posición inicial en la parte final, situación que simula un movimiento bastante acelerado de una parte del cuerpo del usuario.

De no existir esta corrección sería imposible confiar en los datos que brinda el giroscopio, dado que acumula error en cada cambio de dirección de giro o movimiento brusco.

### 3.3.2.8 CORRECCIÓN DE YAW A TRAVÉS DE DATOS DEL MAGNETÓMETRO

Para el caso del ángulo *Yaw*, la corrección se la realiza usando el valor de ángulo calculado en el inciso 3.3.2.4. Este ángulo va a indicar la orientación teniendo como referencia el norte magnético de la Tierra, es decir todos los módulos detectores de

movimiento tendrán la capacidad de orientarse en relación al Norte en todo momento.

La corrección de este valor se la realiza ya no en la condición en reposo, sino el programa examina si es que la magnitud del campo magnético de la Tierra está dentro de parámetros normales, mide su dirección y la aplica la corrección al valor calculado por el giroscopio si este es diferente.

El bloque de programa es el siguiente:

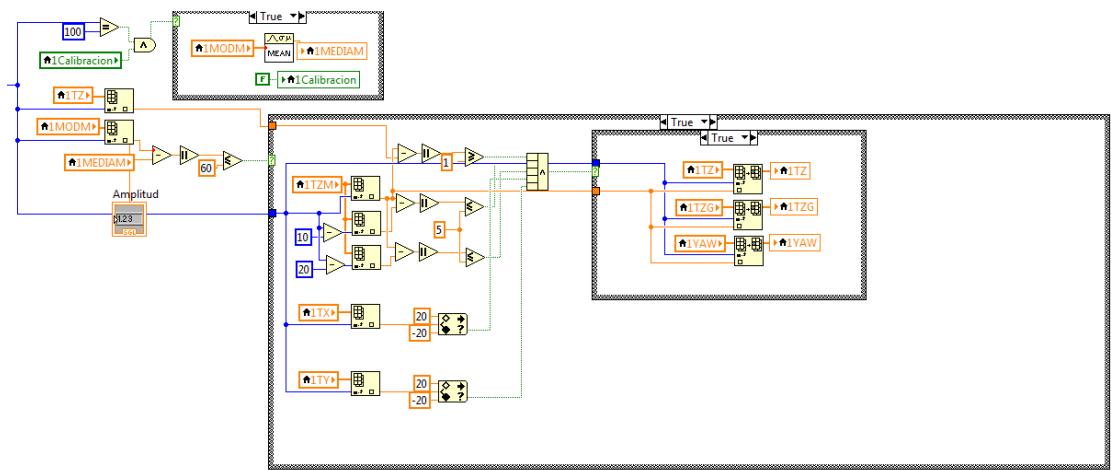


Figura 3.30: Bloque de programación para corrección de Yaw.

La corrección no se puede dar en todo instante de tiempo, también se debe tomar en cuenta la inclinación del módulo detector de movimiento sobre su eje y para que no afecte al ángulo de orientación del norte magnético.

Sin embargo estas 2 correcciones anteriores realizan un proceso de revalorización al ángulo corregido, creando un cambio brusco en el valor, si esto se llevara a un gráfico en 3D se podría apreciar un cambio brusco y anti natural del ángulo, para mejorar este aspecto se ha creado el siguiente código de programa.

### 3.3.2.9 ACONDICIONAMIENTO DE SEÑALES PARA REALIDAD AUMENTADA

Para evitar cambios bruscos que se pueden causar por correcciones de errores en tiempo real, se ha creado un bloque de código que convertirá esta corrección en un cambio natural.

El programa reacciona ante diferencias grandes de magnitud de ángulo y las convierte en cambios ligeros a través de adición de un factor de corrección a sus valores anteriores en el tiempo. De esta manera cuando se grafique en 3D no se tendrá cambios bruscos de ángulos y la representación será más cercana a la realidad.

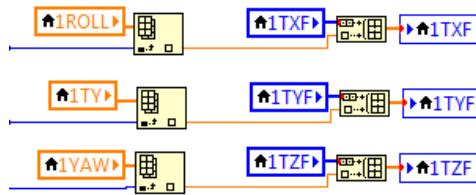


Figura 3.31: Grabación de variables para gráficos 3D.

En la figura 3.31 se registran los valores de ángulos que se representarán en el modelo en 3D. Para ahorrar tiempo de procesamiento, se ha utilizado variables de 16 bits enteros con signo.

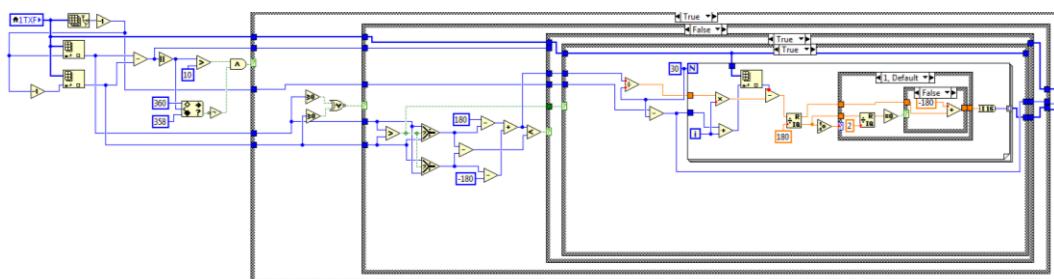


Figura 3.32: Bloque de programa de acondicionamiento de la señal ante cambios bruscos antinaturales.

El proceso descrito en la figura 3.32 se aplica para los 3 ángulos calculados finales, de tal manera que los datos que van a mostrarse en las gráficas y el modelado 3D sea lo más apegado a la realidad posible.

Como se puede ver en la figura 3.33, la señal se acondiciona para que cuando existan cambios bruscos por corrección de errores como los mostrados en la señal en rojo se acondicione a señales continuas con cambios naturales, de esta manera el Avatar que se graficará tendrá siempre movimientos casi exactos a los reales hechos por el usuario.

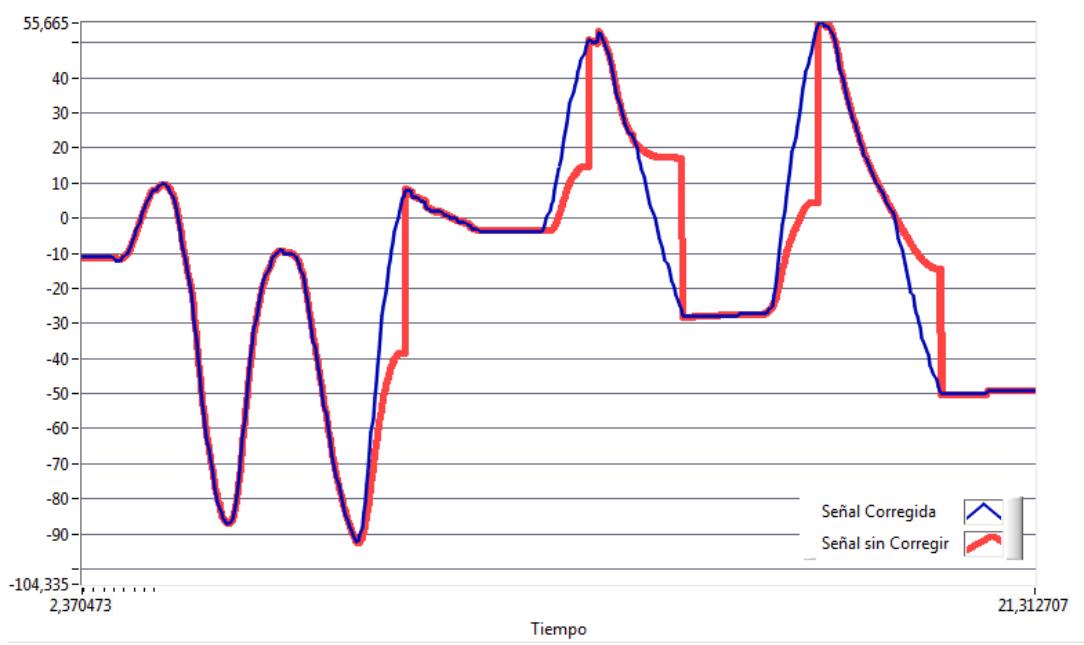


Figura 3.33: Señal *Roll* acondicionada para graficar en 3D.

### 3.3.3 DISEÑO DE INTERFAZ DE USUARIO

La interfaz de usuario se ha diseñado realmente simple, básicamente está orientada a mostrar al usuario 2 bloques de información.

Primero, del lado izquierdo se mostrarán los ángulos de cada una de las partes del cuerpo que son relevantes para un análisis de riesgo ergonómico como el propuesto al inicio de este proyecto.

Segundo, del lado derecho se mostrará la representación 3D del movimiento del paciente de cada uno de las partes superiores del cuerpo. En la figura 3.34 se muestra la interfaz de usuario.

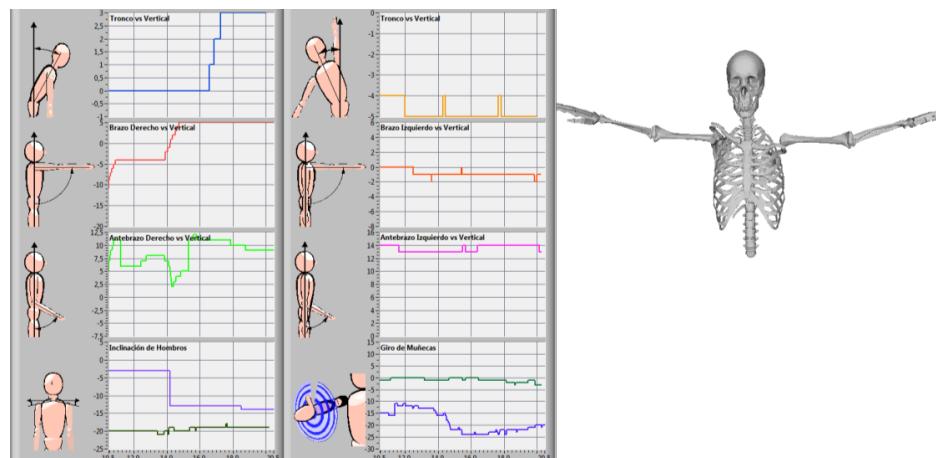


Figura 3.34: Interfaz de Usuario.

Para la construcción de la interfaz de usuario se necesitaron algunos bloques de programación, como es el caso de la creación de las gráficas que representan el ángulo de cada extremidad o parte del cuerpo con relación a un eje fijo en el espacio.

Las gráficas se generan con las variables finales descritas en la sección 3.3.2.9, las cuales representan el ángulo *Roll*, *Pitch* y *Yaw* de cada una de las partes del cuerpo analizadas con los módulos detectores de movimiento.

Estas variables, muestran a una tasa de 33 veces por segundo los ángulos de orientación en 3D de cada parte del cuerpo. El bloque de programación para la gráfica de estos ángulos es el siguiente.

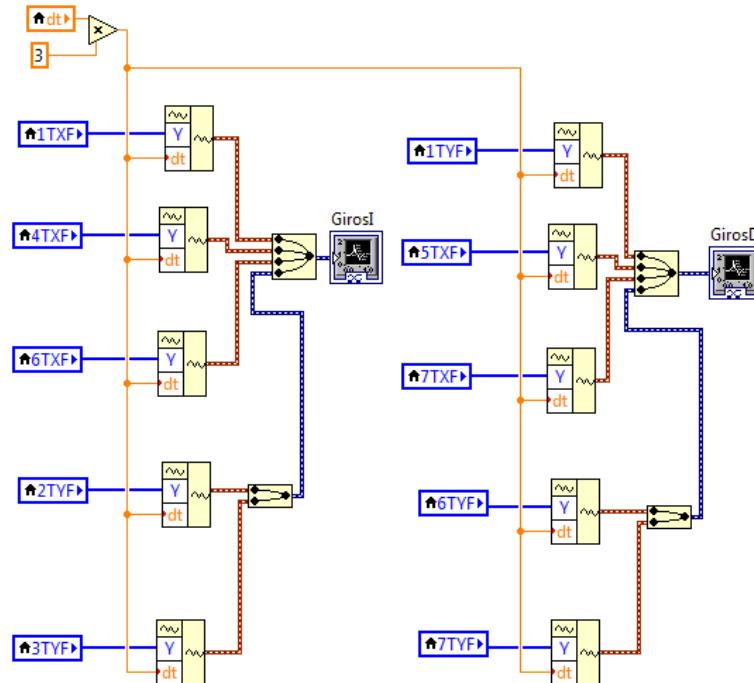


Figura 3.35: Bloque de Programación para mostrar gráficas de ángulos.

Para la representación de movimientos en 3D del esqueleto se utilizaron los datos provenientes de las variables calculadas en la sección 3.3.2.9, a través de las cuales se calculó el movimiento que se debe realizar a cada parte del cuerpo en 3D para representar correctamente el movimiento.

Cada parte del cuerpo tiene un modelo en 3D en formato STL, este modelo se importa al entorno gráfico del programa y se lo mueve y rota en dependencia de los datos recibidos de los módulos detectores de movimiento.

El bloque de programación para importar los modelos STL al programa se muestra en la figura 3.36.

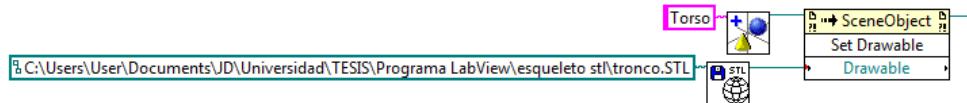


Figura 3.36: Bloque de programación para importar modelos 3D tipo STL.

Una vez importados los modelos necesarios de cada uno de las partes del cuerpo se deben acomodar en puntos específicos en el espacio tridimensional para posteriormente ser utilizados.

La figura 3.37 muestra la forma de acomodar cada parte del cuerpo para que se puedan mover y rotar bajo sistemas coordinados independientes, de esta manera la rotación o movimiento de una parte del cuerpo no afectará los movimientos de otra parte del cuerpo.

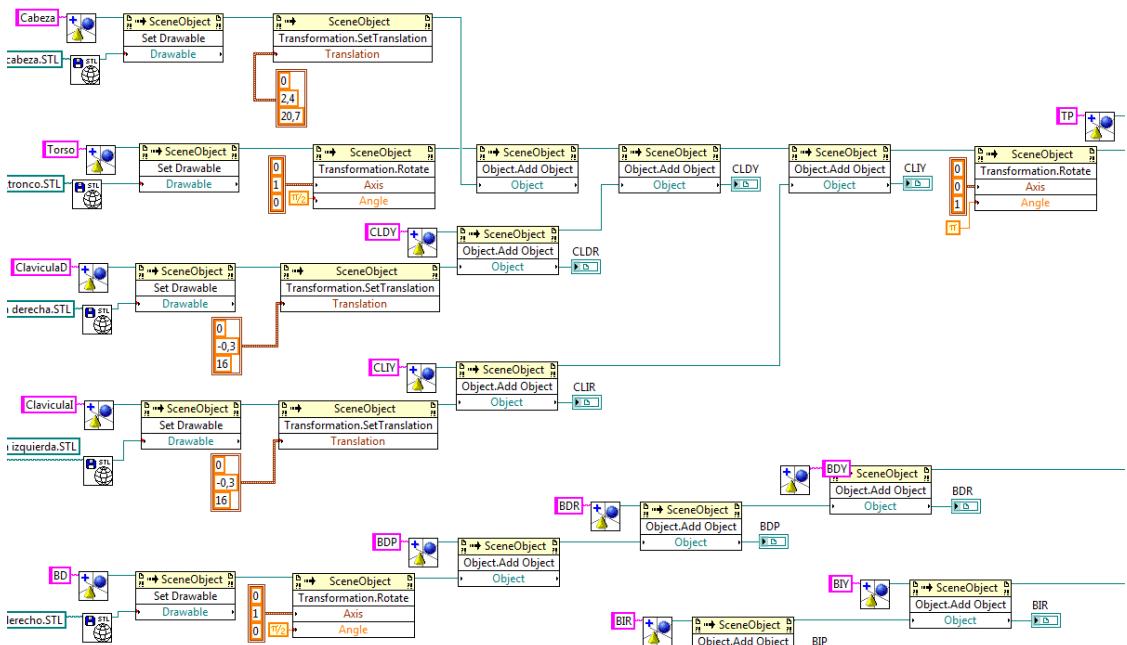


Figura 3.37: Movimiento y rotación inicial de cada parte del cuerpo.

En la figura 3.37 se muestran las configuraciones iniciales de cada una de las partes del cuerpo, de manera que posteriormente se las pueda mover y rotar de manera independiente y conseguir una representación de los movimientos en 3D.

Para mover cada una de las partes representadas por modelos 3D, se utilizan bloques repetitivos que se ajustan según los datos recibidos de los módulos detectores de movimiento.

En la figura 3.38, se utilizan los valores de posición del torso guardados en las variables 1TXF, 1TYF y 1TZF para realizar los movimientos de los modelos 3D.

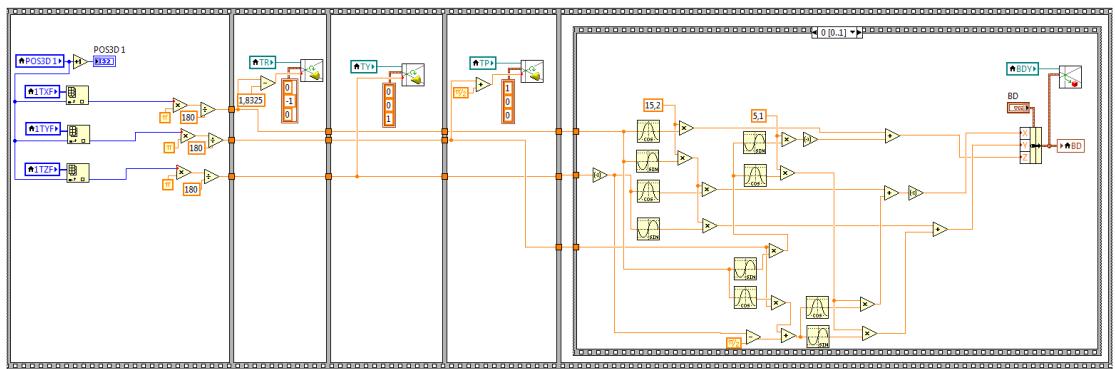


Figura 3.38: Bloque de programación para movimiento de Torso y cálculo de posición inicial de brazos.

Una vez realizado el movimiento se debe calcular los puntos de origen del brazo derecho e izquierdo, esto se calcula con simples cálculos de posiciones con teorema de Pitágoras y los ángulos de *Roll* y *Pitch* calculados previamente.

Mediante estos ángulos se calcula el valor rotado y traslado de la parte del cuerpo que se requiere posicionar, como por ejemplo el valor de posición inicial  $P_2(x,y,z)$  del brazo izquierdo y derecho con respecto al valor de rotación del torso y su punto inicial  $P_0(x,y,z)$ , pasando por el punto  $P_1(x,y,z)$ .

Para calcular estos puntos se debe aplicar en 2 ocasiones el teorema de Pitágoras para encontrar puntos en 3D con ángulos previamente calculados. La figura 3.40 ilustra la manera de calcular esta posición.

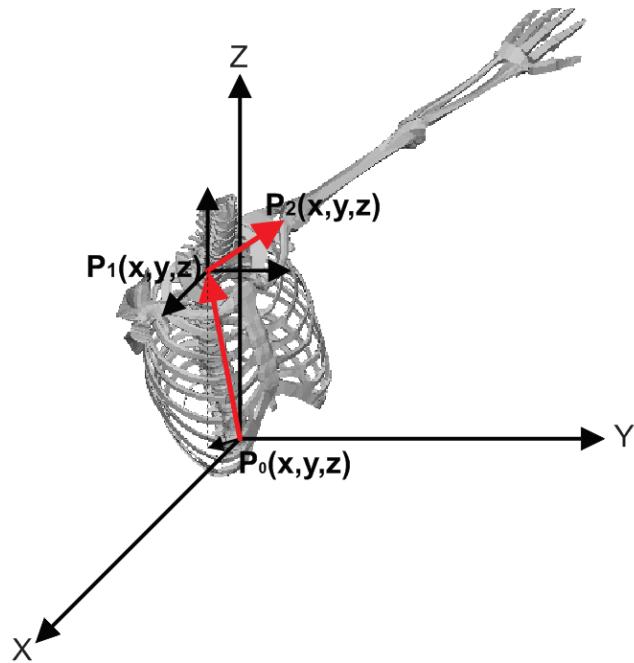


Figura 3.39: Pasos para llegar al punto de cada brazo en relación a la traslación y rotación del torso.

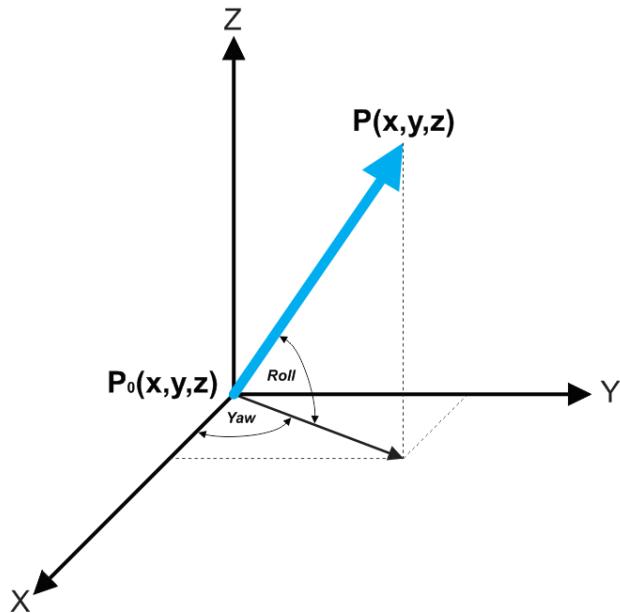


Figura 3.40: Método de cálculo de posición de brazo izquierdo y derecho utilizando teorema de Pitágoras.

Las fórmulas para el cálculo de la posición de cada brazo se muestran a continuación.

$$z = h_T * \cos(Roll) + L_C * \sin(Roll) * \sin(Pitch * \sin(Roll)) \quad (3.12)$$

$$x = h_T * \sin(Roll) * \cos(Yaw) + L_C * \cos(Pitch * \sin(Roll)) * \cos(\cos(Roll) * Pitch) + (Yaw - 90^\circ) \quad (3.13)$$

$$y = h_T * \text{Sen}(Roll) * \text{Sen}(Yaw) + L_C * \text{Cos}(Pitch * \text{Sen}(Roll)) * \text{Sen}(\text{Cos}(Roll) * \\ \text{Pitch}) + (Yaw - 90^\circ) \quad (3.14)$$

El bloque de programación así como las fórmulas de cálculo los antebrazos son diferentes ya que se parte de condiciones iniciales, para los antebrazos solo se necesita un paso aplicando el teorema de Pitágoras.

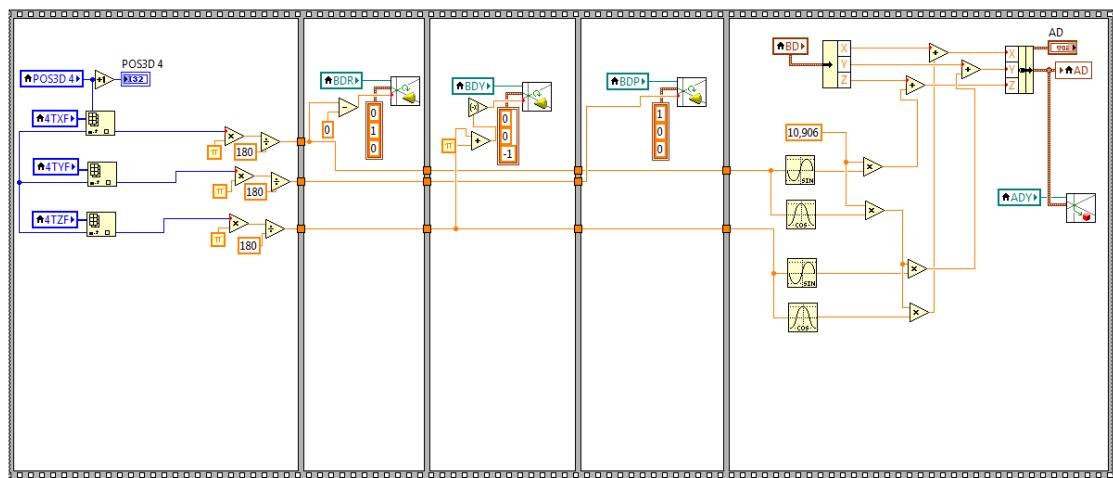


Figura 3.41: Bloque de programación para movimiento de Brazos y cálculo de posición inicial de antebrazos.

Para el cálculo del punto inicial de cada antebrazo, se debe partir del cálculo previamente realizado del punto inicial de los brazos.

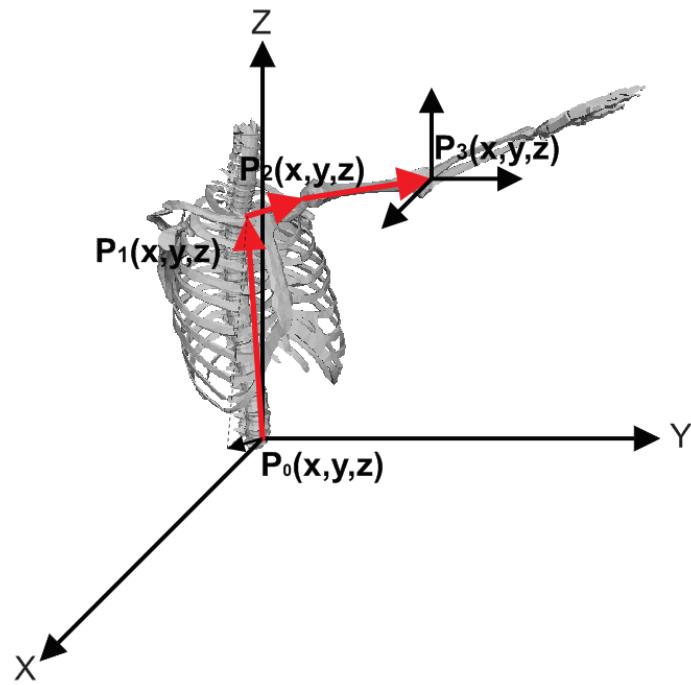
Las fórmulas necesarias para el cálculo se muestran a continuación:

$$z = x_T + h_B * \text{Sen}(Roll) \quad (3.15)$$

$$y = y_T + h_B * \text{Cos}(Roll) * \text{Sen}(Yaw) \quad (3.16)$$

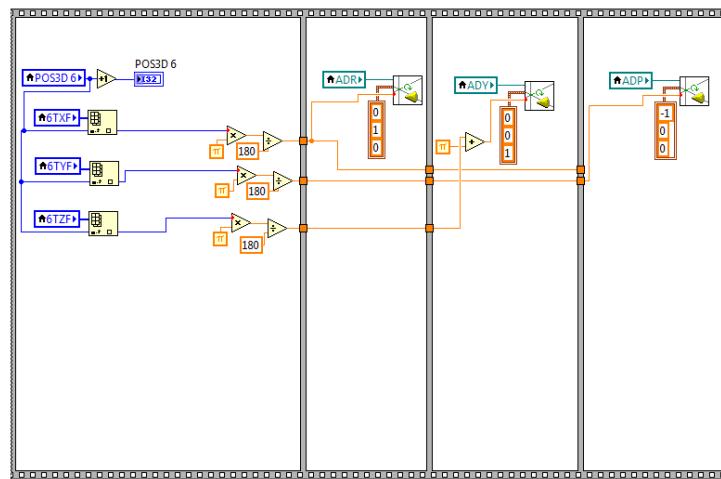
$$x = x_T + h_B * \text{Cos}(Roll) * \text{Cos}(Yaw) \quad (3.17)$$

Donde los términos  $x_T$ ,  $y_T$  y  $z_T$  son los términos previamente calculados de la posición de cada uno de los brazos, según sea el caso. El término  $h_B$  representa la longitud del brazo.



**Figura 3.42: Puntos para llegar a la posición del antebrazo a partir de la posición del brazo.**

Una vez calculados los puntos correspondientes a cada antebrazo y trasladados como se ve en la figura 3.41, se debe rotar según los datos calculados de los módulos detectores de movimiento.



**Figura 3.43: Bloque de programa para rotaciones del modelo 3D del antebrazo.**

La figura 3.43 muestra las rotaciones necesarias sobre el modelo 3D de cada antebrazo para que se ajusten a los datos medidos por los sensores inerciales.

Otros modelos 3D a rotar son las clavículas, estas se colocan sobre una posición fija y solo se rotan en relación a su eje según el movimiento detectado por los sensores. El proceso es el mismo tanto para la clavícula izquierda como derecha.

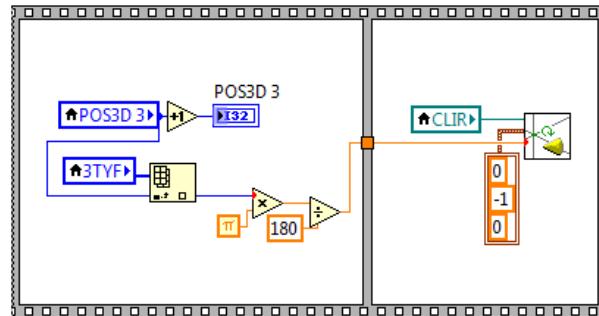


Figura 3.44: Bloque de programa para giro de clavícula.

Una vez realizadas las traslaciones y rotaciones el modelo 3D completo del esqueleto queda como se muestra en la figura 3.45.

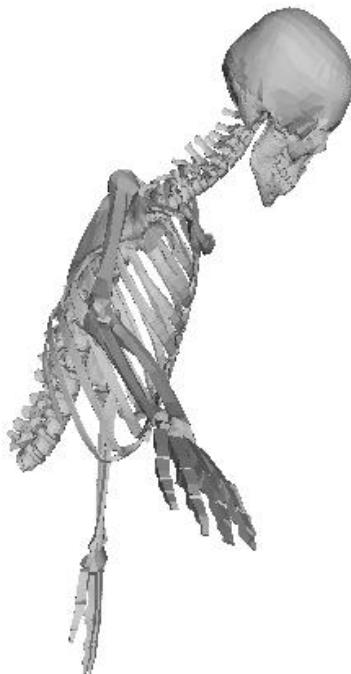


Figura 3.45: Imagen 3D construida a partir de los datos de sensores inerciales.

## CAPÍTULO 4

### 4 ANÁLISIS DE RESULTADOS OBTENIDOS

En el presente capítulo se analizarán los resultados obtenidos a través de los cálculos explicados en el capítulo anterior. El cálculo de error de los módulos detectores de movimiento y pruebas realizadas sobre usuarios se mostrarán en los incisos a continuación.

#### 4.1 CÁLCULO DE ERROR

Para calcular el error de cálculo de los módulos detectores de movimiento se necesita un método que permita obtener una medida patrón confiable para que sea comparada con la medida de ángulo calculado.

Cada módulo detector de movimiento está compuesto de sensores inerciales del mismo fabricante, por lo que el cálculo del modelo del error de uno coincidirá con el de los módulos detectores de movimiento restantes.

Para realizar las pruebas sobre el cálculo del ángulo, se ha diseñado un método que permita medir con exactitud utilizando el instrumento patrón.

En la figura 4.1, se puede ver la plataforma construida para la medición de los ángulos sobre un eje del módulo detector de movimiento, la plataforma gira sobre una bisagra halada por un cordel que se sujetá sobre un eje fijo en la parte superior.

Una vez que la plataforma gira en torno a la bisagra produce un ángulo de elevación que equivale al ángulo de rotación sobre el eje del módulo detector de movimiento que está alineado con el movimiento de rotación. El ángulo es calculado y se comparará con la medida del instrumento patrón.

El instrumento patrón para el cálculo del ángulo de rotación es el programa para diseño gráfico *Corel Draw X5*, el cual es capaz de medir ángulos con precisión a través de una fotografía. La plataforma de la figura 4.1 se fotografía y luego se usa la fotografía para el cálculo patrón de varias medidas de ángulo de elevación y compararlas con medidas del sistema diseñado.

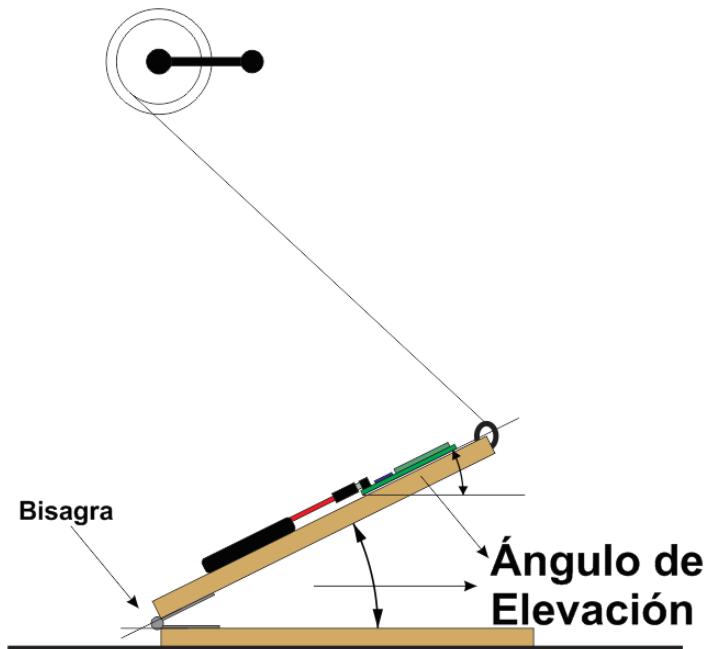


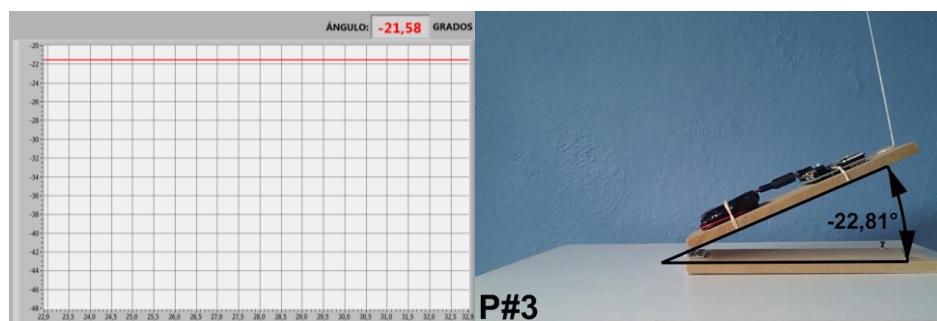
Figura 4.1: Método para medición del ángulo de elevación del módulo detector de movimiento.

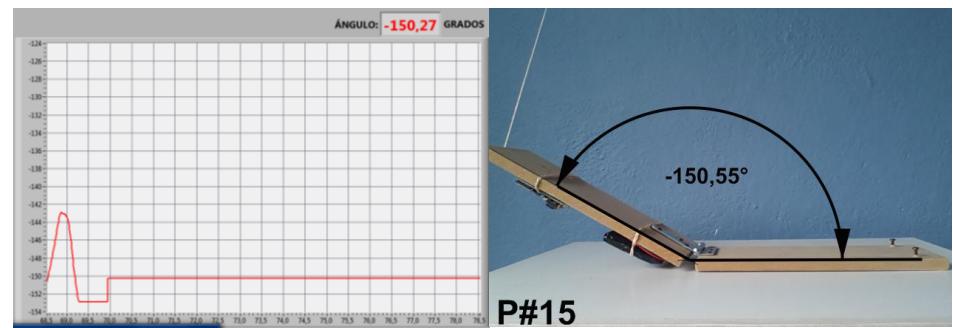
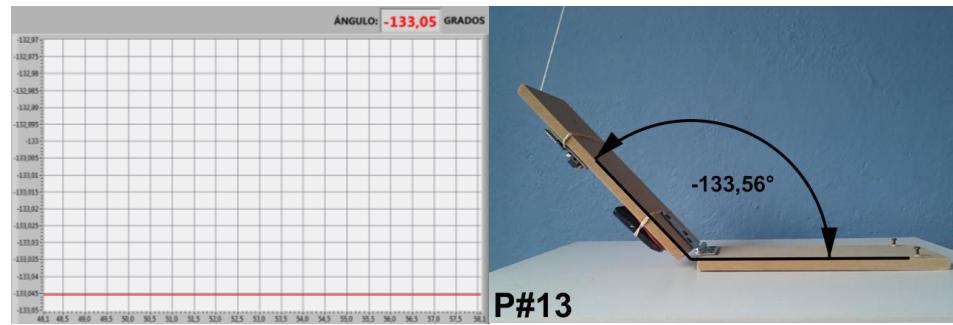
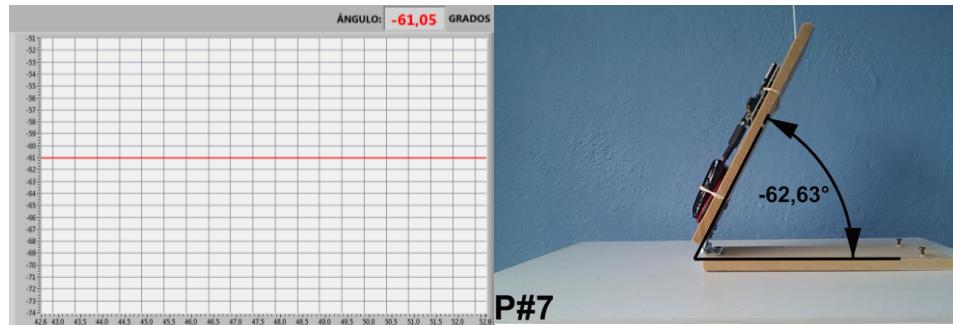
Las pruebas realizadas se realizan sobre los 3 ejes del módulo detector de movimiento a la vez. Las pruebas sobre uno de los ejes de un módulo será aplicables a los otros ejes, y por tanto a los otros módulos detectores de movimiento, lo que se busca es conocer la precisión del sistema diseñado.

Las pruebas realizadas se muestran a continuación en comparación con la medida patrón.

De lado izquierdo se encuentra el *Valor Medido*, que es el valor calculado con el sistema diseñado, mientras que de lado derecho se encuentra el *Valor Real*, que es el valor medido con el instrumento patrón.

En las gráficas del *Valor Medido* se muestran un historial de 5 segundos a 100 muestras por segundo, son 500 muestras por medición.





**Figura 4.2: Comparación de ángulo de elevación del módulo detector de movimiento real y medido.**  
**Algunas pruebas realizadas.**

El cálculo del error se lo realiza a través de la siguiente fórmula:

$$\text{Error} = \text{Valor Real} - \text{Valor Medido} \quad (4.1)$$

La tabla 4.1, 4.2 y 4.3 muestran los datos medidos y reales que se han obtenido en los 3 ejes en las pruebas realizadas, además el error, la distribución normal y el error

porcentual contra un error de 180 grados considerado la máxima falla que puede tener una medida equivalente al 100% de error.

Valor Medido	Valor Real	Error	Error Porcentual	Distribución Normal
-86,08	-89,74	-3,66	4,08%	0,022115095
152,36	149,13	-3,23	2,17%	0,05921574
122,78	119,99	-2,79	2,33%	0,131790158
140,8	138,12	-2,68	1,94%	0,155768902
113,47	111	-2,47	2,23%	0,206648964
159,24	156,8	-2,44	1,56%	0,214324493
131,3	129,04	-2,26	1,75%	0,261331277
100,74	98,92	-1,82	1,84%	0,365923019
-61,05	-62,63	-1,58	2,52%	0,40239977
-30,83	-32,28	-1,45	4,49%	0,412734725
-53,02	-54,44	-1,42	2,61%	0,414076776
81,09	79,69	-1,40	1,76%	0,414748686
-123,74	-125,1	-1,36	1,09%	0,415554012
-69,42	-70,69	-1,27	1,80%	0,414729173
-39,95	-41,21	-1,26	3,06%	0,414412595
-100,75	-102	-1,25	1,23%	0,414051301
-21,58	-22,81	-1,23	5,39%	0,413195049
-115,19	-116,2	-1,01	0,87%	0,392479819
-10,11	-11,01	-0,90	8,17%	0,375050922
-140,31	-141,1	-0,79	0,56%	0,353718417
-154,61	-155,38	-0,77	0,50%	0,349478599
12,46	11,94	-0,52	4,36%	0,289763037
-133,05	-133,56	-0,51	0,38%	0,287193691
1,01	0,53	-0,48	90,57%	0,279439448
71,14	70,77	-0,37	0,52%	0,250651686
40,11	39,79	-0,32	0,80%	0,237532048
-150,27	-150,55	-0,28	0,19%	0,227088286
52,7	52,44	-0,26	0,50%	0,221895286
60,24	60,08	-0,16	0,27%	0,196374746
76,05	75,93	-0,12	0,16%	0,186440554
23,2	23,17	-0,03	0,13%	0,164837167
32,04	32,39	0,35	1,08%	0,088944942
-0,01	0,53	0,54	101,89%	0,061605736

Tabla 4.1: Eje X: Cálculo de error y distribución normal de los datos obtenidos.

Valor Medido	Valor Real	Error	Error Porcentual	Distribución Normal
-81,75	-84,21	-2,46	2,92%	0,0515868
24,67	22,54	-2,13	9,45%	0,07306006
-28,59	-30,69	-2,10	6,84%	0,07524728
-110,61	-112,35	-1,74	1,55%	0,10426522
2,11	0,48	-1,63	339,58%	0,11401771
102,54	101,36	-1,18	1,16%	0,15636872
-99,05	-100,06	-1,01	1,01%	0,17254528
89,88	88,92	-0,96	1,08%	0,17722746
-19,22	-19,88	-0,66	3,32%	0,2038279
0,10	-0,53	-0,63	118,87%	0,20629325
-124,05	-124,54	-0,49	0,39%	0,21717187
-141,83	-142,22	-0,39	0,27%	0,22422303
74,97	74,85	-0,12	0,16%	0,23963542
-12,70	-12,63	0,07	0,55%	0,246801
40,86	41,12	0,26	0,63%	0,25057062
153,20	153,47	0,27	0,18%	0,25067126
62,20	63,1	0,90	1,43%	0,23735404
141,21	142,15	0,94	0,66%	0,23528014
131,30	132,63	1,33	1,00%	0,20892691
49,02	50,55	1,53	3,03%	0,19203654
114,13	115,73	1,60	1,38%	0,18575644
9,65	11,36	1,71	15,05%	0,17560979
31,29	33,24	1,95	5,87%	0,15278999
-75,66	-73,71	1,95	2,65%	0,15278999
-133,07	-131,09	1,98	1,51%	0,14991365
-63,39	-61,24	2,15	3,51%	0,13370403
-156,88	-154,65	2,23	1,44%	0,12619347
-41,88	-39,45	2,43	6,16%	0,1080063
-56,12	-53,64	2,48	4,62%	0,10362787
123,09	125,8	2,71	2,15%	0,08457877

Tabla 4.2: Eje Y: Cálculo de error y distribución normal de los datos obtenidos.

Valor Medido	Valor Real	Error	Error Porcentual	Distribución Normal
53,80	51,22	-2,58	5,04%	0,06363659
92,37	89,95	-2,42	2,69%	0,07461728
-111,76	-114,12	-2,36	2,07%	0,0790052
102,22	100,08	-2,14	2,14%	0,09626661
-10,42	-12,36	-1,94	15,70%	0,11335482
-19,31	-20,88	-1,57	7,52%	0,14723589
-62,03	-63,46	-1,43	2,25%	0,16032148
-83,78	-85,13	-1,35	1,59%	0,16774286
22,48	21,4	-1,08	5,05%	0,19188238
-40,63	-41,65	-1,02	2,45%	0,19694646
134,06	133,33	-0,73	0,55%	0,21903053
-0,32	-0,53	-0,21	39,62%	0,24428556
0,68	0,53	-0,15	28,30%	0,24572193
33,27	33,13	-0,14	0,42%	0,24592886
-125,52	-125,33	0,19	0,15%	0,24742942
40,14	40,39	0,25	0,62%	0,246585
121,20	121,58	0,38	0,31%	0,24359911
-143,54	-143,12	0,42	0,29%	0,24236878
-131,07	-130,47	0,60	0,46%	0,23510165
11,04	11,66	0,62	5,32%	0,23412661
-50,87	-49,87	1,00	2,01%	0,21008427
61,51	62,75	1,24	1,98%	0,19062406
73,92	75,54	1,62	2,14%	0,15615343
110,73	112,36	1,63	1,45%	0,15521883
-79,50	-77,86	1,64	2,11%	0,15428385
-34,49	-32,62	1,87	5,73%	0,13284175
139,53	141,78	2,25	1,59%	0,09919763
-101,87	-99,53	2,34	2,35%	0,09181416
-154,79	-152,36	2,43	1,59%	0,08471449
153,50	156,32	2,82	1,80%	0,0576473

Tabla 4.3: Eje Z: Cálculo de error y distribución normal de los datos obtenidos.

En las tablas 4.1, 4.2 y 4.3 muestran casilleros del cálculo del error porcentual remarcados en color azul, cabe recalcar que el valor alto de estos valores no se debe a la existencia de una magnitud de error grande, sino más bien a que la magnitud del ángulo medido es muy pequeña; el error porcentual es muy grande, mientras que el error en grados se comporta con la misma tendencia que en otras medidas.

El valor de la media del error calculado en la tabla 4.1 se obtiene con la fórmula:

$$\mu = \frac{x_0 + x_1 + \dots + x_n}{n} \quad (4.2)$$

Donde  $n$  es el número de muestras.

La desviación estándar de los valores de error se calcula con la fórmula:

$$\sigma = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2 \quad (4.3)$$

Donde  $n$  es el número de muestras.

Con los valores de error de la tabla 4.1, 4.2, 4.3 y las fórmulas 4.2 y 4.3, se calcula el valor de la media y desviación estándar.

$$\mu_x = -1,33 \quad \sigma_x = 0.9597$$

$$\mu_y = 0.37 \quad \sigma_y = 1.0588$$

$$\mu_z = 0.07 \quad \sigma_z = 1.0608$$

La distribución normal se calcula a través de los valores de error, media y desviación estándar mostrados anteriormente y la siguiente fórmula:

$$Z = \frac{x - \mu}{\sigma} \quad (4.4)$$

El valor obtenido de Z en la ecuación 4.4 se reemplaza en la tabla 4.4, para obtener el valor de distribución normal.

Las gráficas de la distribución normal de los 3 ejes se muestran en las figuras 4.2, 4.3 y 4.4.

Normal Deviate z	.00	.01	.02	.03	.04	.05	.06	.07	.08	.09
-4.0	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
-3.9	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
-3.8	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
-3.7	.0001	.0001	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
-3.6	.0002	.0002	.0001	.0001	.0001	.0001	.0001	.0001	.0001	.0001
-3.5	.0002	.0002	.0002	.0002	.0002	.0002	.0002	.0002	.0002	.0002
-3.4	.0003	.0003	.0003	.0003	.0003	.0003	.0003	.0003	.0003	.0002
-3.3	.0005	.0005	.0005	.0004	.0004	.0004	.0004	.0004	.0004	.0003
-3.2	.0007	.0007	.0006	.0006	.0006	.0006	.0006	.0005	.0005	.0005
-3.1	.0010	.0009	.0009	.0009	.0008	.0008	.0008	.0008	.0007	.0007
-3.0	.0013	.0013	.0013	.0012	.0012	.0011	.0011	.0011	.0010	.0010
-2.9	.0019	.0018	.0018	.0017	.0016	.0016	.0015	.0015	.0014	.0014
-2.8	.0026	.0025	.0024	.0023	.0023	.0022	.0021	.0021	.0020	.0019
-2.7	.0035	.0034	.0033	.0032	.0031	.0030	.0029	.0028	.0027	.0026
-2.6	.0047	.0045	.0044	.0043	.0041	.0040	.0039	.0038	.0037	.0036
-2.5	.0062	.0060	.0059	.0057	.0055	.0054	.0052	.0051	.0049	.0048
-2.4	.0082	.0080	.0078	.0075	.0073	.0071	.0069	.0068	.0066	.0064
-2.3	.0107	.0104	.0102	.0099	.0096	.0094	.0091	.0089	.0087	.0084
-2.2	.0139	.0136	.0132	.0129	.0125	.0122	.0119	.0116	.0113	.0110
-2.1	.0179	.0174	.0170	.0166	.0162	.0158	.0154	.0150	.0146	.0143
-2.0	.0228	.0222	.0217	.0212	.0207	.0202	.0197	.0192	.0188	.0183
-1.9	.0287	.0281	.0274	.0268	.0262	.0256	.0250	.0244	.0239	.0233
-1.8	.0359	.0351	.0344	.0336	.0329	.0322	.0314	.0307	.0301	.0294
-1.7	.0446	.0436	.0427	.0418	.0409	.0401	.0392	.0384	.0375	.0367
-1.6	.0548	.0537	.0526	.0516	.0505	.0495	.0485	.0475	.0465	.0455
-1.5	.0668	.0655	.0643	.0630	.0618	.0606	.0594	.0582	.0571	.0559
-1.4	.0808	.0793	.0778	.0764	.0749	.0735	.0721	.0708	.0694	.0681
-1.3	.0968	.0951	.0934	.0918	.0901	.0885	.0869	.0853	.0838	.0823
-1.2	.1151	.1131	.1112	.1093	.1075	.1056	.1038	.1020	.1003	.0985
-1.1	.1357	.1335	.1314	.1292	.1271	.1251	.1230	.1210	.1190	.1170
-1.0	.1587	.1562	.1539	.1515	.1492	.1469	.1446	.1423	.1401	.1379
-0.9	.1841	.1814	.1788	.1762	.1736	.1711	.1685	.1660	.1635	.1611
-0.8	.2119	.2090	.2061	.2033	.2005	.1977	.1949	.1922	.1894	.1867
-0.7	.2420	.2389	.2358	.2327	.2296	.2266	.2236	.2206	.2177	.2148
-0.6	.2743	.2709	.2676	.2643	.2611	.2578	.2546	.2514	.2483	.2451
-0.5	.3085	.3050	.3015	.2981	.2946	.2912	.2877	.2843	.2810	.2776
-0.4	.3446	.3409	.3372	.3336	.3300	.3264	.3228	.3192	.3156	.3121
-0.3	.3821	.3783	.3745	.3707	.3669	.3632	.3594	.3557	.3520	.3483

Fuente: ProEva, Facultad de Ingeniería, Universidad de la República Uruguay, 2012.

Tabla 4.4: Tabla para cálculo de distribución normal.

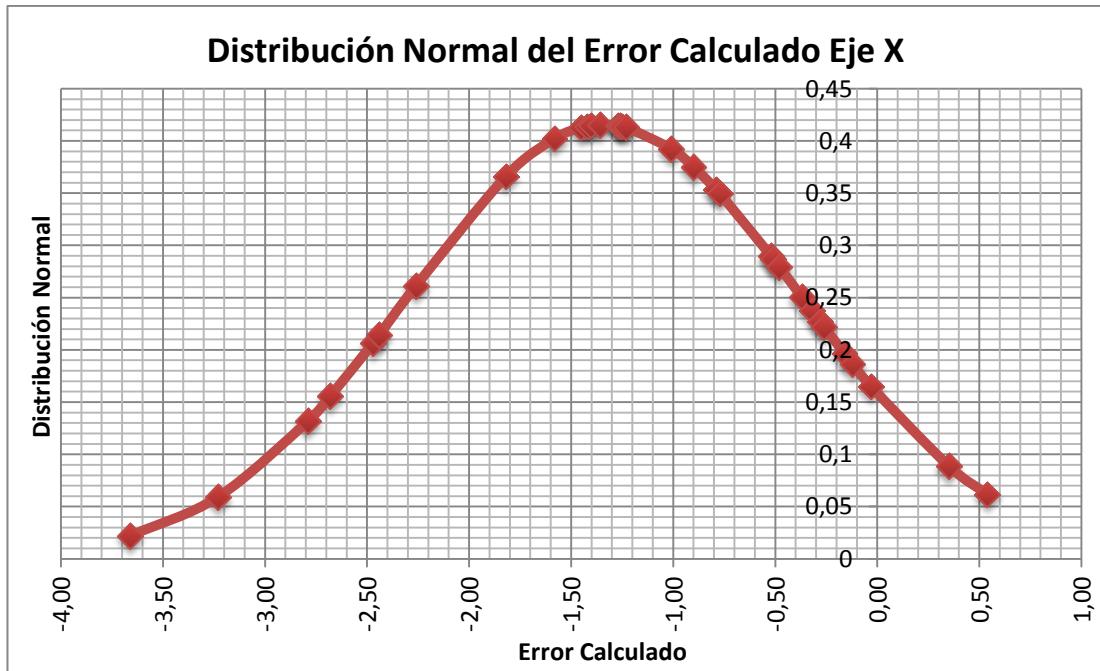


Figura 4.2: Distribución Normal del Error Calculado en Eje X.

### Distribución Normal del Error Calculado Eje Y

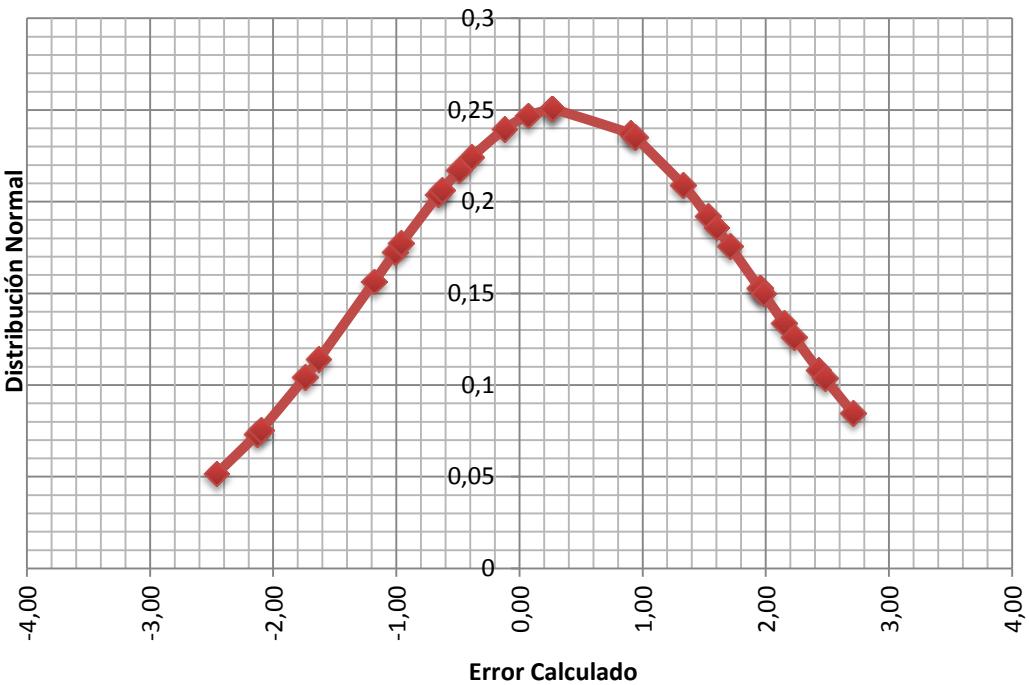


Figura 4.3: Distribución Normal del Error Calculado en Eje Y.

### Distribución Normal del Error Calculado Eje Z

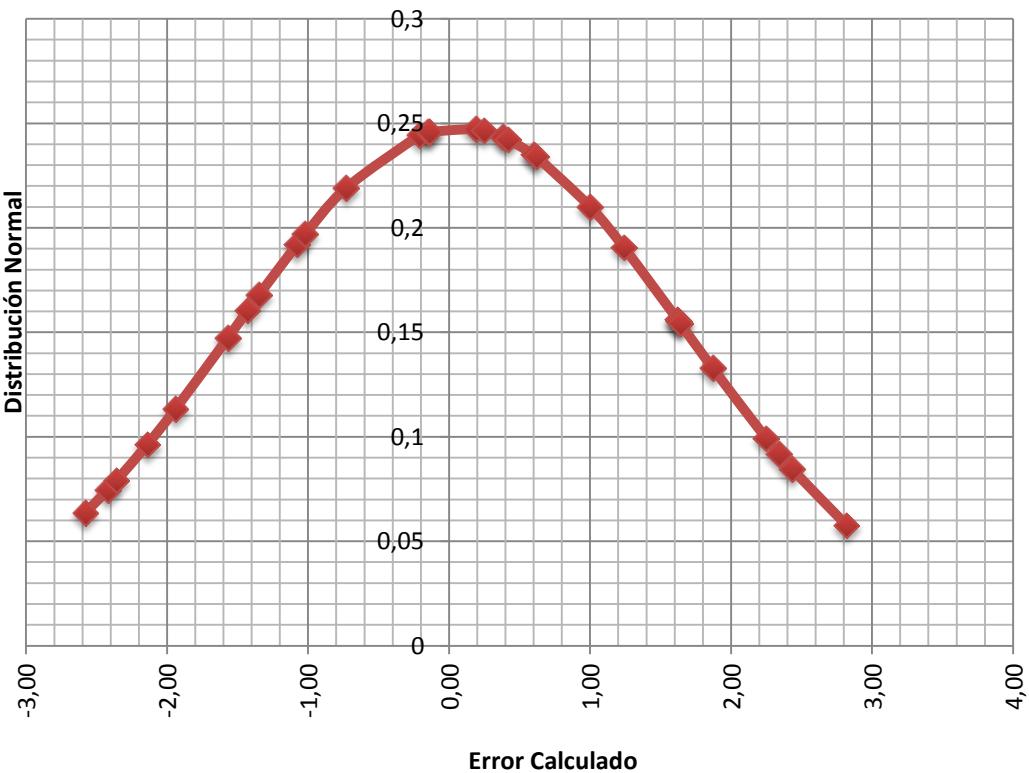


Figura 4.4: Distribución Normal del Error Calculado en Eje Z.

Como se observa la distribución normal del error de cada eje, tienen una media diferente de cero, lo que indica que el error calculado está compuesto de un error sistemático que puede presentarse en todas las medidas, y por supuesto el error aleatorio que se considera error blanco gaussiano.

El error promedio calculado de los ejes X, Y y Z es de  $-1.33^\circ$ ,  $0.37^\circ$  y  $0.07^\circ$  respectivamente.

Según la gráfica 4.2 y el valor calculado de desviación estándar podemos saber que existe un 95.4% de probabilidad que suceda una lectura con un error en el rango:  $-2\sigma \leq Error \leq +2\sigma$ ; es decir que existe un 95.4% de probabilidad que el error de una lectura sobre el eje X este en un rango de:  $-1.9194^\circ \leq Error \leq +1.9194^\circ$ , rango que es bastante aceptable.

Bajo el mismo concepto sobre el eje Y, existe un 95.4% de probabilidad que el error en una lectura este en un rango de  $-2.1176^\circ \leq Error \leq +2.1176^\circ$ .

De igual forma en el eje Z, existe un 95.4% de probabilidad que el error en una lectura este en un rango de  $-2.1216^\circ \leq Error \leq +2.1216^\circ$ .

Los valores máximos de las probabilidades dadas anteriormente son extremos, la realidad es que se tiene aproximadamente el 65% de probabilidades que el valor de error para cada medida en cada eje no sea mayor a  $1^\circ$ , un error que en realidad es muy despreciable.

Esto significa que bajo condiciones normales, tomando un error igual a  $\sigma$  y una extremidad con una longitud promedio de 60cm de longitud el error de la posición real en el espacio contra la posición medida en el espacio tridimensional puede ser apenas de 10.04mm, algo que puede ser despreciado si tomamos en cuenta el análisis de riesgo ergonómico o cualquier otro tipo de análisis biomecánico.

Una vez conociendo al detalle el módulo detector de movimiento, se debe pasar a probar el sistema con los 7 módulos funcionando a la vez y colocados en las partes del cuerpo específicas. A continuación se muestran algunas pruebas realizadas sobre un sujeto.

## 4.2 PRUEBAS COMPARATIVAS MODELO 3D VS POSICIÓN REAL

### 4.2.1 PRUEBA #1

La primera prueba realizada es la medición de ángulos iniciales, como se puede ver en la figura 4.3 la fotografía muestra la posición inicial para el proceso de representación en 3D.



Figura 4.3: Posición de prueba #1.

Las gráficas correspondientes a la lectura de los sensores inerciales en la posición mostrada en la figura 4.3 se pueden apreciar a continuación.

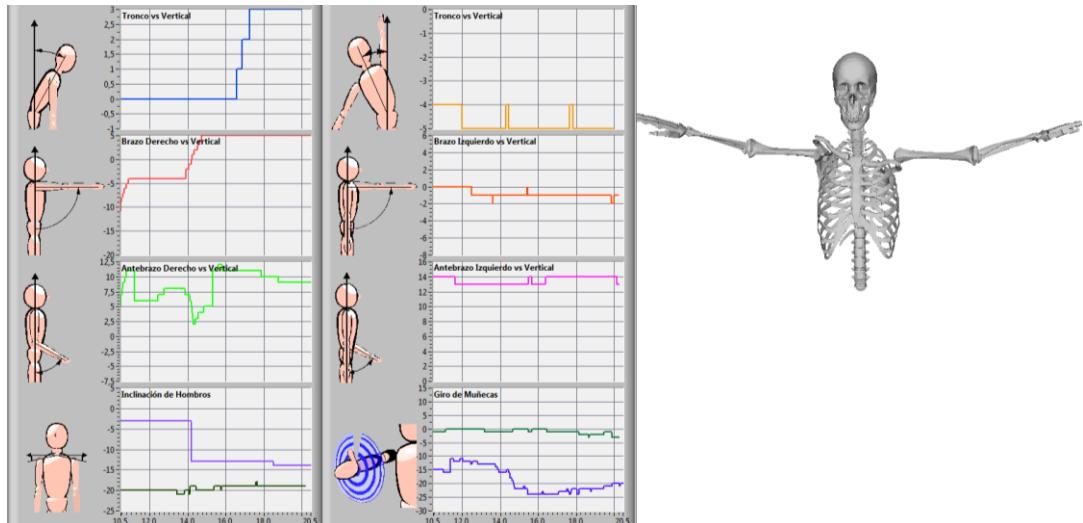


Figura 4.4: Imagen 3D construida y datos obtenidos a partir de los datos de sensores inerciales en prueba #1.

Como se puede apreciar la imagen 3D es muy precisa en posición inicial, coincide inclusive el ángulo en el que se ven giradas las manos, gracias al sensor de la muñeca. Pero en una posición como esa es muy difícil medir ángulos ya que depende del punto de donde se tome la fotografía.

#### 4.2.2 PRUEBA #2

La siguiente prueba se realizará para comprobar los datos de medición de posición y rotación del tronco.



Figura 4.5: Posición de prueba #2.

Las gráficas correspondientes a la lectura de los sensores inerciales en la posición mostrada en la figura 4.5 se pueden apreciar a continuación.

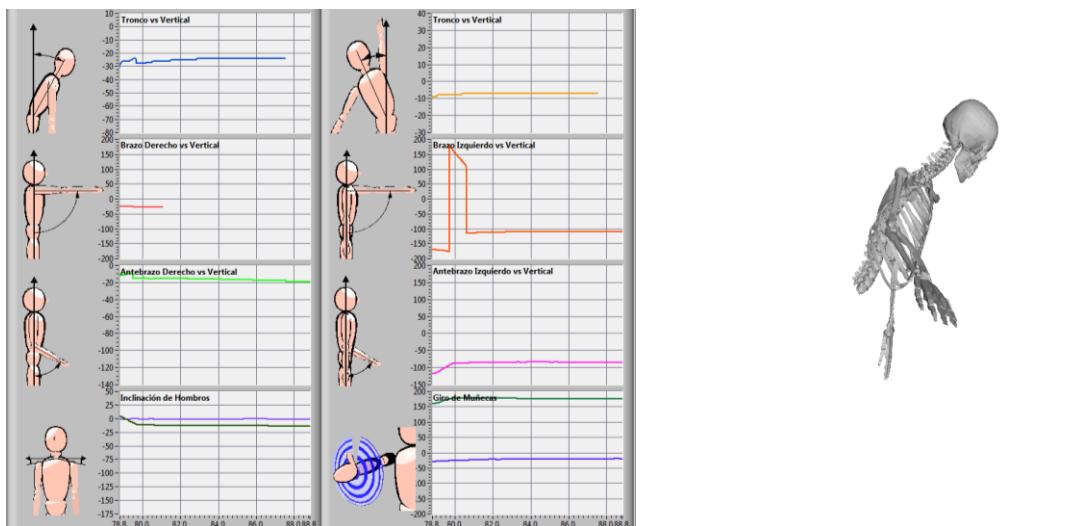


Figura 4.6: Imagen 3D construida y datos obtenidos a partir de los datos de sensores inerciales en prueba #2.

Como se muestra en la figura 4.5 y 4.6, la medida del ángulo de rotación del tronco con respecto a la columna baja son muy similares, el sensor mide alrededor de 18 grados, mientras que la medida en la fotografía muestra una lectura menos precisa debido a la ropa del usuario de alrededor de 20 grados.

#### 4.2.3 PRUEBA #3

Los módulos detectores de movimiento tienen la capacidad de ubicar el norte magnético, de esta manera todos los sensores estarán orientados en su rotación al eje Z general gracias al norte magnético, al igual que una brújula.

La prueba para verificar esta medición se la puede realizar con una vista de los ángulos desde arriba en el modelo 3D y compararlo con la medición real, en la figura a continuación se muestra la fotografía en posición real.



Figura 4.7: Posición de prueba #3.

Las gráficas correspondientes a la lectura de los sensores inerciales en la posición mostrada en la figura 4.7 se pueden apreciar a continuación.

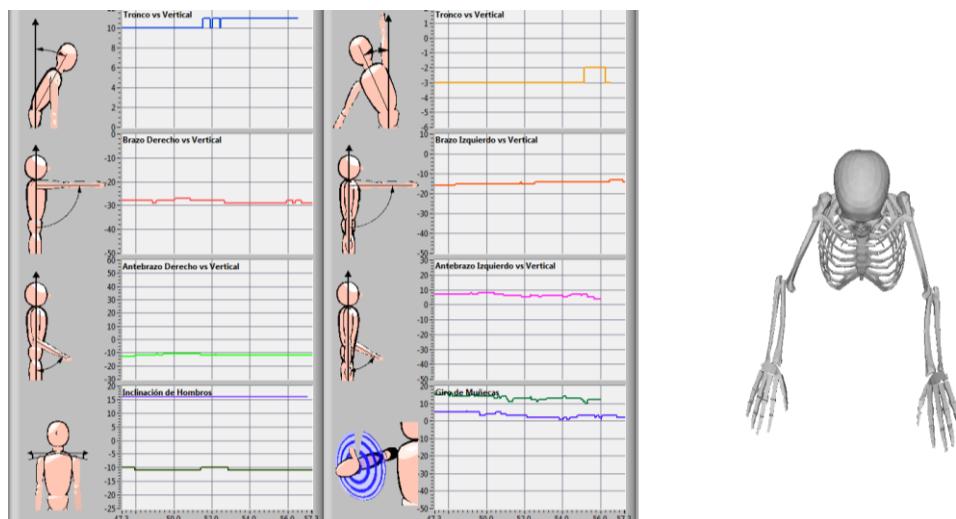


Figura 4.8: Imagen 3D construida y datos obtenidos a partir de los datos de sensores inerciales en prueba #3.

#### 4.2.4 PRUEBA #4

La siguiente prueba se realizará sobre el movimiento de los hombros, es otra medición necesaria para el análisis ergonómico, como se puede ver en la figura 4.9.



Figura 4.9: Posición de prueba #4.

Las gráficas correspondientes a la lectura de los sensores inerciales en la posición mostrada en la figura 4.7 se pueden apreciar a continuación.

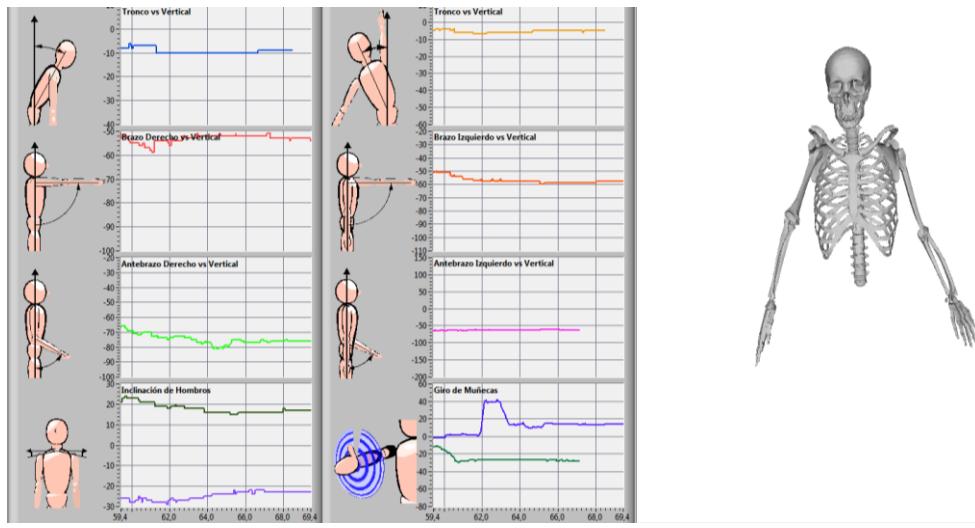


Figura 4.10: Imagen 3D construida y datos obtenidos a partir de los datos de sensores inerciales en prueba #4.

Como se puede apreciar, las medidas de los hombros de la fotografía comparada con los sensores inerciales son muy precisas, al momento de la fotografía los hombros se encuentran inclinados alrededor de 20 grados cada uno, el hombro derecho presenta más inclinación que el izquierdo, como se ve en la fotografía.

#### 4.2.5 PRUEBA #5

La siguiente prueba se realizará tomando las medidas del brazo y el antebrazo combinadas y compararlas contra la posición real.



Figura 4.11: Posición de prueba #5.

Las gráficas correspondientes a la lectura de los sensores inerciales en la posición mostrada en la figura 4.11 se pueden apreciar a continuación. Como se puede ver en la gráfica la posición del brazo y del antebrazo coinciden con la lectura de los sensores inerciales y la gráfica 3D.

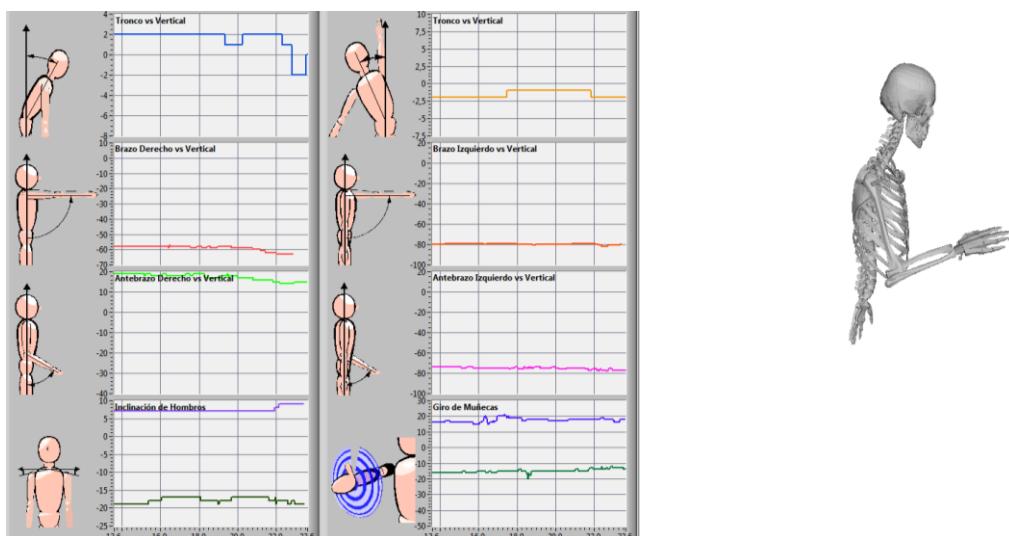
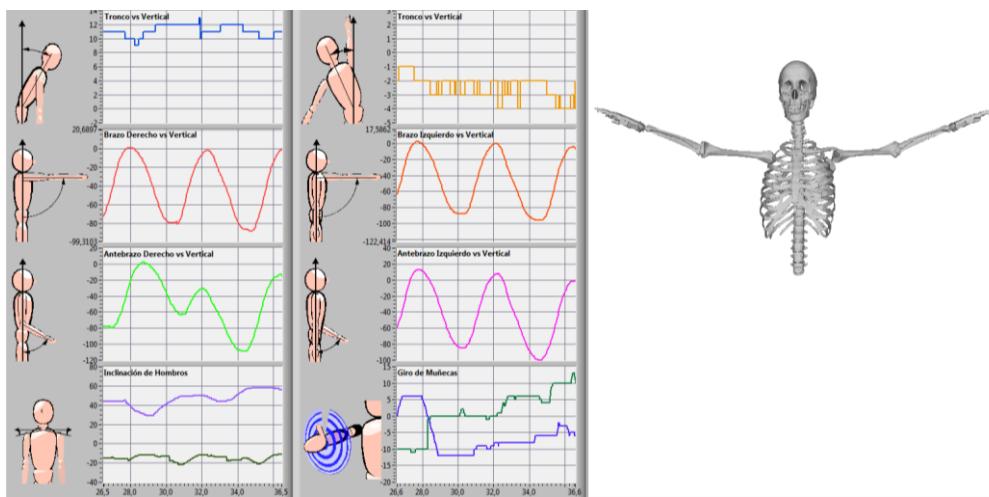


Figura 4.12: Imagen 3D construida y datos obtenidos a partir de los datos de sensores inerciales en prueba #5.

Como se puede apreciar en la figura 4.11 y 4.12, las medidas reales y las tomadas son iguales, lo que demuestra la precisión de la medición.

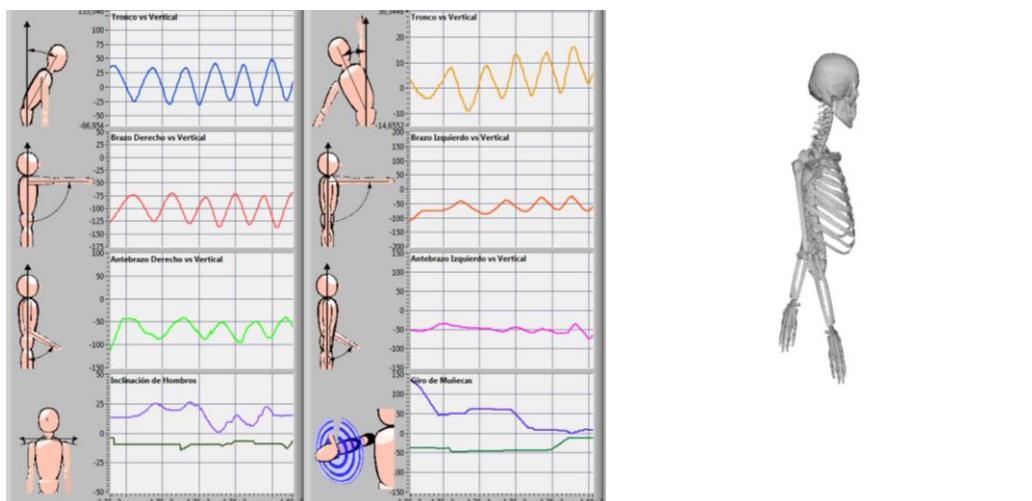
#### 4.2.6 PRUEBA #6

Esta prueba se basa en mover brazos y antebrazos izquierdo y derecho en igual cantidad de manera continua en un intervalo aproximado de 80 a 90 grados, las gráficas de la figura 4.13 muestran en la historia de los gráficos de los ángulos lo dicho anteriormente.



**Figura 4.13:** Imagen 3D construida y datos obtenidos a partir de los datos de sensores iniciales en prueba #6.1.

De la misma manera en la figura 4.14, se realiza la misma prueba con la medición del torso, nótese que por condiciones de equilibrio el usuario a movido los brazos de la misma manera en ángulos parecidos lo que causa lecturas similares en los brazos, estas fueron arbitrarias ya que dependen en su mayoría de la forma de equilibrarse del usuario.



**Figura 4.14:** Imagen 3D construida y datos obtenidos a partir de los datos de sensores iniciales en prueba #6.2.

#### **4.2.7 FALLAS DEL SISTEMA**

El sistema diseñado tiene una buena exactitud en el cálculo de ángulos de inclinación de las extremidades pero al ser un prototipo inicial tiene aún algunos errores.

##### **4.2.7.1 RETARDO DE REPRESENTACIÓN 3D**

La representación 3D tiene un retardo con respecto a los datos recibidos, esto se debe a que se necesita más tiempo de máquina para graficar el modelo del esqueleto en 3 dimensiones y moverlo con los datos adquiridos de los módulos detectores de movimiento.

El retardo en la llegada de los datos desde los sensores inerciales hasta el PC es de 250ms, a partir de ahí y tras todos los cálculos el tiempo total desde que sucede el movimiento hasta representarlo en 3D es de 1250ms a 1750ms según la velocidad del CPU, se tiene que considerar que una PC tiene muchísimos procesos funcionando al mismo tiempo para mantener el sistema operativo funcionando, se debe tratar de disminuir al máximo el consumo de recursos del CPU para aumentar la velocidad de la gráfica 3D.

##### **4.2.7.2 ÁNGULOS RECTOS**

Cuando se va a calcular los ángulos rectos, es decir los que son iguales a +90 o -90 grados, el módulo detector de movimiento experimenta cambios en las magnitudes de los ejes del acelerómetro y del magnetómetro, causando que se pueda perder la orientación con respecto al norte magnético.

La solución para este problema es evitar corregir ángulos de con respecto a referencias generales si el ángulo de uno de los ejes se encuentra en el intervalo de 85 a 90 grados, o de -85 a -90 grados. Sin embargo al confiar solamente en referencias internas como el cálculo que realiza el giroscopio puede incurrir en errores que saltan a la vista en la representación 3D.

##### **4.2.7.3 ALGORITMOS INCOMPLETOS**

A pesar de haber realizado un sistema confiable en muchos sentidos, aún no está listo para representar el movimiento de una parte del cuerpo con movimiento complejo al 100%. Los brazos o piernas son extremidades que pueden moverse a grandes velocidades tanto lineales como angulares, pero sobre todo angulares.

Los algoritmos utilizados en este sistema pueden mostrar grandes falencias a la hora de mostrar movimientos rápidos con precisión, debido a dos principales razones, el *drift* de los sensores iniciales y los algoritmos incompletos que podrían no estar considerando algunas fases del movimiento real de la parte del cuerpo analizada.

#### **4.2.7.4 SENsoRES INERcIALES POCO CONFIABLES**

Lastimosamente dadas la experiencia con los sensores utilizados, cabe decir que son sensores poco confiables para la labor requerida, pueden servir perfectamente para navegación sobre aeronaves o vehículos en tierra o mar, pero debido a la gran complejidad de movimientos de una parte del cuerpo como brazos, antebrazos o piernas, resulta muy difícil que el sensor, sobre todo el giroscopio no cometa errores que pueden significar mucho en una gráfica en 3D.

El *drift* es una de las características de sensores iniciales de este tipo, debido a su precio económico este fenómeno traducido al español como derrape, se causa ante cambios en la aceleración del sensor, que debido a la inercia del cuerpo y a sus partes móviles causan lecturas incorrectas en los sensores y en el proceso final, causando pérdida de datos por el efecto derrape e incluso aparición de datos que no existen.

Tanto el acelerómetro, giroscopio y magnetómetro pueden ser influenciados por condiciones externas.

El acelerómetro y giroscopio pueden variar sus mediciones en función de la temperatura en la que se encuentren trabajando, según el fabricante recomienda una temperatura ambiente de alrededor de 25 grados centígrados para conseguir el óptimo funcionamiento de estos.

#### **4.2.7.5 PÉRDIDA DE DATOS EN TRANSMISIÓN HASTA EL PC**

Se pueden perder datos en la transmisión desde el módulo detector de movimiento la PC, las razones pueden ser debido a la distancia de los módulos hasta el enrutador o simplemente por pérdida de estos por errores de transmisión. Debido a que el módulo realiza una conversión de Serial a IEEE802.11, puede pasar que un paquete de estos se pierda, se han incluido algoritmos que evitan estas pérdidas pero en ocasiones son inevitables. Sin embargo, se ha podido registrar un pérdida de menos del 0.1% en 20 minutos continuos de transmisión, pero esto podría afectar al efecto visual de la gráfica 3D, al igual que a los valores posteriores de los sensores iniciales, debido a

que cada uno calcula datos en función de los datos recibidos en el tiempo, por lo menos hasta poder igualarse a una referencia global.

## CAPÍTULO 5

### PRESUPUESTO

Tras haber terminado la construcción del sistema, el presupuesto se lo ha realizado en base a lo gastado, sin considerar costos de pérdida, como fallas en la construcción, o elementos electrónicos quemados, en la tabla 5.1 se muestran los costos parciales y el costo final del sistema.

Elemento	Costo (USD)
Elementos Electrónicos Comunes	210.00
PCBs	64.00
Módulos Wifi	336.00
Módulos de Sensores Inerciales	160.00
Prenda para usuario	40.00
Enrutador	35.00
Baterías Recargables	140.00
<b>COSTO TOTAL</b>	<b>985.00</b>

**Tabla. 5.1: Costo de cada elemento del sistema construido.**

## **CONCLUSIONES Y RECOMENDACIONES**

### **1. Acerca del Sistema Construido:**

Se ha podido desarrollar un sistema que puede medir la orientación y posición de las partes del cuerpo y representarlas en un entorno 3D. El sistema si bien puede tener falencias evidenciables en ciertas situaciones, demanda una gran labor en pruebas y desarrollo de software y hardware, logrando conseguir gran exactitud en otras condiciones, como se puede ver en la obtención de errores en el cálculo de los ángulos, donde mediante el cálculo de la distribución normal se puede decir que una medida tomada en un eje del módulo detector de movimiento tiene un 95.4% de probabilidad de tener un error con una diferencia de  $0^\circ$  a  $\pm 1.91^\circ$  con respecto al valor real, y tiene alrededor del 65% de probabilidad de que el error sea de  $0^\circ$  a  $\pm 1^\circ$ , errores que son despreciables cuando hablamos de rotaciones de articulaciones que se miden en decenas de grados.

Es importante recalcar que sobre condiciones de ángulos no cercanos a 90 grados el sistema tiene una respuesta muy acertada, así como en cualquier posición en estado estático.

Puede mostrar resultados erróneos en condiciones de movimiento muy acelerado.

### **2. Sensores Inerciales:**

Se ha demostrado que los sensores inerciales son capaces de representar el movimiento de una persona en tiempo real con la utilización de los algoritmos correctos y un sistema eficaz de corrección de errores.

Sería mucho más eficaz la utilización de sensores inerciales de mejor calidad, sobre todo enfocándose al error en respuesta lineal, recomiendo que sea menor al 0.05%, debido a que al tomar una cantidad de 100 muestras por segundo el error puede incrementarse muy rápidamente para tener medidas alejadas de la realidad.

Cabe recalcar también que es imposible confiar en un solo sensor inercial para el cálculo de orientación en 3D, el uso de sensores con referencias externas como el acelerómetro con la referencia de la gravedad o el magnetómetro con referencia al campo magnético de la Tierra son necesarios para ubicación, pero se necesita de

sensores sin referencias externas, que aunque sean poco confiables en el tiempo, en corto plazo se apegan mucho a la realidad y combinados con correcciones de otros sensores de referencia externa pueden brindar un sistema totalmente confiable.

### **3. Desarrollo de Algoritmos:**

El sistema construido puede tener deficiencias debido a que se pueden estar obviando fases del movimiento en los algoritmos de cálculo de ángulos, que a la larga van a representar el movimiento del objeto estudiado.

Según mi experiencia de estudio del área de detección de movimiento usando sensores inerciales, es recomendable la utilización un filtro especial de carácter estadístico, llamado filtro Kalman, lamentablemente el uso del filtro comprende uso de matemáticas de alto nivel lo que me hubiese demandado mucho tiempo de aprendizaje, algo complicado dadas las diferentes ramas de este proyecto y el tiempo limitado que tenía para desarrollar el mismo.

### **4. Sensores Inalámbricos:**

Cabe recalcar la exitosa implementación de módulos inalámbricos, fáciles de usar, poco invasivos para la persona y funcionales. Tienen un alcance de hasta 30 metros en condiciones perfectas y su batería está diseñada para que funcionen de 2 a 3 horas continuas, recomiendo la comunicación Wifi, debido al alcance de esta, otros protocolos como el Bluetooth pueden cubrir las necesidades de velocidad, pero en alcance son muy limitados.

Los módulos diseñados envían 100 veces por segundo los datos de cada uno de los sensores inerciales a bordo.

Sería sumamente interesante poder diseñar los mismos módulos detectores de movimiento con un micro procesador que funcione con 3.3V, de montaje superficial y menor potencia, esto reduciría significativamente el tamaño de la tarjeta electrónica y de la batería necesaria para el funcionamiento.

### **5. Resultados Obtenidos:**

Los resultados obtenidos al construir este sistema son alentadores.

En el proceso del desarrollo, se han encontrado grandes dificultades, dadas todas las ramas del sistema, como la comunicación de 7 módulos a la vez, para luego evitar la pérdida de datos, los algoritmos para obtener los datos de los sensores iniciales, la representación 3D, la sincronización de todos los datos, etc, me han llevado a pensar que debí enfocarme en el desarrollo de un solo sensor y todos los algoritmos para que este sea lo más eficiente posible ya sea en reposo o movimiento.

Sin embargo el sistema diseñado cumple los objetivos propuestos, mide la posición del cuerpo del usuario, permite que un profesional de la salud pueda analizar los resultados leídos por los sensores y calificar estos, ya sea buscando riesgos ergonómicos, patrones de movimiento, desempeño deportivo u otras ramas en las que son usados los sensores iniciales como representación de movimiento humano en 3D.

## BIBLIOGRAFÍA

- [1] MERUANE NARANJO, Claudua, *Analysis And Modeling Of Mems Based Inertial Sensors*, Signal Processing School Of Electrical Engineering, Kungliga Tekniska Hgskolan, Stockholm, 2008.
- [2] QUIROGA GARCÍA, Pablo Esteban y WENJIE, Li, *On Indoor Positioning form Mobile Devices*, Tesis de Masterado, Departamento de Señales y Sistemas, Universidad de Chalmers de Tecnología, Göteborg, Sverige, 2011.
- [3] PARK, Minha, *Error Analysis And Stochastic Modeling Of Mems Based Inertial Sensors For Land Vehicle Navigation Applications*, Department Of Geomatics Engineering Galgary, Alberta, Abril 2004.
- [4] MARGOTH FREIRE, Nancy, *Proyecto De Análisis Con El Método Rula Al Departamento De Mantenimiento De Equipo Pesado Del Campo Sacha, Rio Napo, De La Provincia De Orellana En El Año 2010*, Tesis de Ingeniería, Universidad Estatal De Milagro, Milagro, 2010.
- [5] LOPEZ OSORIO, Leidy, *Implementación de un Filtro Kalman para la Localización de un Robot Movil tipo LEGO NXT en LabView*, Tecnura, vol. 16, octubre, 2012, pp. 68-75.
- [6] FATEMEH, Abyarjoo, BARRETO, Armando, COFINO, Jonathan y ORTEGA, Francisco, *Implementing a Sensor Fusion Algorithm for 3D Orientation Detection with Inertial/Magnetic Sensors*, Electrical and Computer Engineering Department, Florida International University, Miami, Florida, USA.
- [7] A. M. Khan y T. Kim, *Accelerometer Signal-Based Human,Activity Recognition using Augmented Autoregressive Model Coefficients, and Artificial Neural Nets*, IEEE EMBC 2008, pp. 5172-5175.
- [8] RAMASWAMY, Bharath, *Kalman Filter Based Estimation of Inertial Measurement Unit Parameters in a Portable Biomechanical Assessment Suite*, Tesis de Ingeniería Eléctrica, The Pennsylvania State University, Mayo de 2011.
- [9] FERRER, Gonzalo, *Integración Kalman de sensores iniciales INS con GPS en un UAV*, Tesis de Ingeniería Electrónica, RSLab TSC UPC, Abril de 2009.
- [10] LIGORIO, Gabriel y SABATINI, Angelo, *Extended Kalman Filter-Based Methods for Pose Estimation Using Visual, Inertial and Magnetic Sensors: Comparative Analysis and Performance Evaluation*, The Institute of BioRobotics, Scuola Superiore Sant'Anna, Piazza Martiri della Libertá 33, 56124 Pisa, Italia, 4 de Febrero de 2013.
- [11] CASTILLA, Alejandro, *Seguimiento Virtual en Tiempo Real de Maniobras de Estabilización de un Simulador de Vuelo Satelital*, Tesis para obtener título de Ingeniero Mecatrónico, Facultad de Ingeniería, Universidad Nacional Autónoma de México, México D.F., México, 2012.
- [12] NATIONAL INSTRUMENTS, HALVORSEN, Hans, *System Identification and Estimation in LabView*, 16 de Agosto de 2011.
- [13] MICROCHIP, *PIC18F2455/2550/4455/4550 Datasheet*, Microchip Technology Inc., 2006, 430 páginas.
- [14] HONEYWELL, *3-Axis Digital Compass IC HMC5883L*, Honeywell Inc., 20 páginas.
- [15] INVENSENSE, *MPU-6000 and MPU-6050 Product Specification*, Revisión 3.4, Invensense Inc., California, USA, 19 de Agosto de 2013, 52 páginas.
- [16] INVENSENSE, *MPU-6000 and MPU-6050 Register Map and Descriptions*, Revisión 4.0, Invensense Inc., California, USA, 3 de Septiembre de 2012, 47 páginas.

- [17] JINAN USR IOT Technology Co., Ltd., *USR-WIFI232-X-V4.3 Embedded WiFi Module User Manual*, Shenshen, China, 65 páginas.
- [18] “Ángulos de Navegación”, Wikipedia La Enciclopedia Libre, [http://es.wikipedia.org/wiki/%C3%81ngulos\\_de\\_navegaci%C3%B3n](http://es.wikipedia.org/wiki/%C3%81ngulos_de_navegaci%C3%B3n).
- [19] JARA DÍAZ, Oswaldo, *Ergonomía: Métodos de Análisis de Riesgo Ergonómico Laboral*, Ponencia presentada en el Curso: Curso para Auditores del Sistema de Gestión de Riesgos del Trabajo, Cuenca, 16 de Junio de 2012.
- [20] VÁSQUEZ, Luis, Fundamento de Ley: *Métodos de Análisis de Riesgo Ergonómico Laboral*, Ponencia presentada en el Curso: Curso para Auditores del Sistema de Gestión de Riesgos del Trabajo, Cuenca, 16 de Junio de 2012.
- [21] IEES, *SEGURO GENERAL DE RIESGOS DEL TRABAJO, NORMATIVAS. DECRETO 2393. RESOLUCIÓN CSNº 741. CONVENIO Nº 121 DE LA OIT. PÁG.: 8 – 78*, Quito, Ecuador, 2009.
- [22] IEES, *RESOLUCIÓN CONSEJO DIRECTIVO CD 333, REGLAMENTO PARA EL SISTEMA DE AUDITORIA DE RIESGOS DE TRABAJO "SART"*, Quito, Ecuador, 2009.