CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS
DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO
DEPARTAMENTO DE COMPUTACIÓN

Desarrollo de una interfaz hombre-máquina usando smartphone con aplicacion a brazo manipulador

Tesis que presenta

Carlos Román Parga Villalpando

Para Obtener el Grado de

Maestro en Ciencias

En

Computación

Director de la Tesis

Dra. Xiaoou Li Zhang

Codirector de Tesis

Dr. Wen Yu Liu

México, Distrito Federal **Octubre, 2013**

Resumen

La Interfaz Hombre-Máquina (IHM) es la parte de un sistema con la que el usuario puede interactuar con el. Estas interfases pueden ser desde simples interruptores y luces indicadoras hasta una sofisticada computadora diseñada a medida con interfases gráficas y sensores especializados. Las interfases pueden ser de monitoreo, de control o para almacenar los datos históricos que ocurren con la máquina dependiendo de la tarea para la cual el sistema esté diseñado. Debido a la complejidad de las tareas que realiza una interfaz es común utilizar un sistema informático como plataforma y la mayoría de las veces tanto el hardware como el software son sistemas hechos a medida.

En esta tesis se propone el uso de un smartphone como interfaz hombre-máquina para controlar un brazo robot manipulador. Se escoje el uso del smartphone por ser una computadora móvil alimentada por baterías e incluye una interfaz gráfica, sensores, comunicación inalámbrica y un sistema operativo al que pueden cargarse aplicaciones para tareas específicas. El sistema operativo Android se eligió sobre otras opciones principalmente por que permite acceder a todos los recursos de hardware y software por medio de las bibliotecas del sistema dándole a la aplicación transportabilidad entre diferentes marcas y modelos de smartphone.

En la tesis se implementa una aplicación para detectar y medir los gestos del brazo de un usuario. Para obtener éstos movimientos se hace uso los sensores inerciales del smartphone donde la información que se obtiene es filtrada y posteriormente un algoritmo de unidad de medición inercial (similar al utilizado en aviones) estima los desplazamientos y la dirección para que la aplicación sea capaz de detectar la magnitud de los gestos corporales del usuario. Este tipo de algoritmo a comparación de las unidades de medición inercial utilizado en vehículos, no requiere de información externa de referencia, como el GPS o balizas para corregir el error de estimación.

La información que la unidad de medición inercial ofrece permite controlar un brazo robot manipulador haciendo uso de cinemática inversa para calcular los ángulos de cada una de las articulaciones del brazo y para posicionar la pinza en las coordenadas que se obtuvieron durante la estimación de desplazamiento. Tanto el proceso de estimación como la cinemática inversa son procesadas dentro del smartphone sin necesidad de realizar procesos en un servidor de apoyo haciendo que la aplicación sea autónoma en sus procesos de cálculo. La información que se envía por Wi-Fi al brazo manipulador son los ángulos en la que los servos de las articulaciones del brazo

deben tener para llegar a la posición que se ha estimado. Aunque en ésta tesis se requiere de una PC para poder realizar la conexión entre el smartphone al robot, no realiza ningún proceso de cálculo y solo sirve para poder hacer la interfaz de Wi-Fi a la tarjeta controladora de los servos del brazo.

Palabras clave: Interfaz Hombre-Máquina, Sensores interciales, Smartphone, Robot manipulador.

Abstract

The Human Machine Interface (HMI) is the part of a system with which the user can interact with him. These interfaces may be from simple switches and indicator lights or a sophisticated custom-designed computer with graphical interfaces and sensors specialized. The interfaces may be for monitoring, control or for storing historical data occurring the machine depending on the task for which the system is designed. Due to the complexity of the tasks of an interface, a computer is an a common system as a platform and most of the time both hardware controller and software are customized systems.

This thesis proposes the use of a smartphone like a human-machine interface to control a robot manipulator arm. The use of a smartphone is due to be a mobile computer powered by batteries and includes a graphical interface, sensors, wireless communication and an operating system that can be loaded applications for specific tasks. The Android operating system was chosen over other options mainly because it allows access to all resources hardware and software through system libraries giving the application portability between different smartphone brands and models.

The thesis implements an application to detect and measure arm gestures of a user. For capture these movements uses inertial sensors of an smartphone where the obtained information is filtered and then the algorithm inertial measurement unit (similar to that used in aircraft) estimates the displacements and direction for the application to be able to detect the extent of user gestures. This type of algorithm, compared of the inertial measurement units used in vehicles, requires no external reference information, such as GPS or beacons to correct the error of estimation.

The inertial measurement unit provides information to control a robot arm manipulator using inverse kinematics to calculate the angles of each of the arm joints and positioning the clamp on the coordinates that were obtained during the displacement estimation. The estimation process and inverse kinematics are processed within the smartphone without the need of a external server support to make the processes due the application autonomous in the calculation processes. The information sent over Wi-Fi to the manipulator arm are the angles in which the joint servos arm must have to reach the position which has been estimated. Although in this thesis requires a PC to make the connection between the smartphone and the robot, does not perform any calculation process and serves only to make the Wi-Fi interface to the controller card servos arm.

keywords: Human-Machine Interfase, Inertial Sensors, Smartphone, Robot.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Actualmente,el uso de robots, es una necesidad en muchos ámbitos de la vida del ser humano. La tecnología tiene una característica bien conocida, esta consiste en que con el paso del tiempo su desarrollo continúa reflejándose en constantes mejoras y un incremento sostenido de su producción, haciendo que su costo de fabricación decrezca. Los robots al ser una aplicación tecnológica es de esperar que éstos tiendan a bajar su complejidad y su precio respecto a sus predecesores, pero a la vez aumente su confiabilidad y su versatilidad. Desde el punto de vista de un consumidor de productos tecnológicos el costo de los mismos es un factor muy importante. Por ello siempre se busca la manera de abaratar los desarrollos usando una estrategia ya conocida, que ofrece buenos resultados, es la de aprovechar el uso de tecnologías ya existentes y realizar solo pequeñas modificaciones para que cumplan con la tarea esperada.

En el desarrollo de los robots, la interfaz de usuario es uno de los componentes que puede ser implementado usando tecnologías existentes, como por ejemplo las computadoras comerciales y en últimos años los dispositivos smartphone. Tratándose específicamente de los smartphone, son dispositivos que han inundado el mercado con amplias y variables prestaciones y costos. Un smartphone es una computadora móvil por lo que puede ser usado como una plataforma integral de proceso, comunicación y adquisición de datos por medio de sensores de las magnitudes físicas que los rodean. Por tratarse de una computadora, los *smartphones* poseen un sistema operativo y se pueden adquirir con una variedad de los mismos. Uno de los sistemas operativos de mayor difusión es el sistema operativo Android y su estructura de software permite acceder a los recursos de hardware para controlarlos y/o obtener la información para realizar aplicaciones. La estructura del sistema operativo Android permite usar bibliotecas incorporadas para acceder a los recursos del smartphone sin necesidad de hacer una aplicación específica para cada modelo de dispositivo donde se instalará. En ésta tesis se plantea utilizar un smartphone con Android para desarrollar una interfaz usuario-máquina y controlar un brazo robot manipulador de forma inalámbrica usando el *Wi-Fi*. Una característica que se va a estudiar con esta interfaz es el uso de los sensores de acelerómetro y giroscopio para implementar una unidad de medición

inercial modificada que permita obtener los desplazamientos y rotaciones en 3 ejes simultáneamente del brazo del usuario aplicando los principios de navegación inercial. Los datos de movimiento calculados por la unidad de medición inercial se ingresarán al modelo de cinemática inversa para controlar las posiciones de las articulaciones de un brazo robot manipulador. La comunicación entre el smartphone con la interfaz hombre máquina y el brazo robot manipulador será inalámbrica usando el protocolo TCP-IP.

En éste capítulo se hará una revisión de los antecedentes, se explicará la motivación del trabajo de tesis, planteamiento del problema y contribuciones de la tesis.

1.1. Antecedentes

El control de un robot brazo manipulador necesita de 6 comandos (3 posiciones y 3 direcciones) para que sea capaz de manipular objetos como lo hace un brazo humano dentro de un área de acción. Existen varias maneras para obtener éstos comandos y cada una de ellas está diseñada para cubrir necesidades específicas y al mismo tiempo hacer que la interfaz sea lo más práctica posible.

El método más difundido es el uso de un *joystick* para enviar 2 direcciones de movimiento y un pedal o *joystick* auxiliar para enviar la tercera dirección. Para alcanzar la posición deseada, el usuario observa si las 3 posiciones son alcanzadas por el robot [11] [12] a forma de retroalimentación del sistema. La carencia que presenta este método consiste en que el *joystick* no puede enviar 6 comandos al mismo tiempo debido la construcción mecánica del mismo por lo que el usuario solo puede enviar dos de los 6 comandos simultáneamente. Otro inconveniente con los *joystick* es su precisión ya que si los sensores del mismo no son análogos no es posible controlar la magnitud de la dirección que se envía.

Otra interfaz consiste en utilizar un sensor de fuerza [17]. Este sensor puede enviar 6 comandos (3 fuerzas y 3 pares de torsión). Los problemas de este método son que el sensor de fuerza que permita 6-ejes simultáneamente es muy caro (aproximadamente 10,000 dolares EE.UU.) además de que no tiene comunicación inalámbrica por si misma obligando que su uso sea con cables a menos que se implemente adicionalmente un dispositivo de conexión inalámbrica.

El uso del smartphone en robótica para realizar la interfaz de usuario es un concepto relativamente nuevo ya que su introducción al mercado no tiene más de 8 años y han ido evolucionando desde entonces. El principal atractivo de estos dispositivos es que son plataformas de cómputo completas, es decir que cuentan con procesador, memoria, pantalla, sistema operativo y su programación es muy similar a como se programa una PC convencional. Una característica particular de éstos dispositivos es que se alimentan exclusivamente por baterías, siendo éstas de poco peso y per-

mitiendo que todo el smartphone no sea más grande y pesada que una libreta de notas. El avance tecnológico de la miniaturización ha permitido integrar más periféricos internos que amplían la funcionalidad además que la tecnología de dispositivos micro mecánicospermite la incorporación de sensores como por ejemplo el GPS, acelerómetro, giroscopio, termómetro, fotómetro y cámara. La conectividad también se ha diversificado con Wi-Fi, Bluetooth y 3G. Todas éstas características permiten que el smartphone se considere como una plataforma muy atractiva para desarrollar aplicaciones orientadas a la robótica como interfaz de usuario.

Un smartphone posee muchas ventajas como una interfaz hombre máquina y ya se han estado haciendo investigaciones con ellos. Prácticamente se han centrado en dos tipos de interfaz:

1. Monitoreo. El teléfono inteligente se utiliza como una computadora para registrar y guardar los datos del robot [2] [5] [6] [11].

2. Control. La pantalla táctil y teclado de los teléfonos inteligentes se utilizan para enviar comandos al robot [7] [9] [12] [13].

Uno de los primeros trabajos usando una computadora móvil se realizó utilizando una PDA como interfaz de usuario para controlar un robot móvil. El sistema consistía en enviar los comandos capturados en la *PDA* a un servidor PC el que interpretaba los datos y se encargaba de controlar el robot móvil. La comunicación de los tres dispositivos se hacía con la infraestructura *WiFi*. Los resultados expuestos de éste desarrollo son prometedores al haber logrado implementar la interfaz visual en el dispositivo y facilitando la representación de los datos de forma gráfica [21].

Otra propuesta analizada en el uso del smartphone es su utilización como interfaz alternativa a otra más especializada. Esta aplicación pretende que sea usada en situaciones donde no se requiere hacer uso de excesiva precisión del robot a controlar. Si bien puede un smartphone hacer función de interfaz existen limitantes en el diseño de la presentación de la información en pantalla debido a su reducido tamaño y no puede integrar el nivel de complejidad de una interfaz especializada para robots muy sofisticados. En este desarrollo se pudo demostrar que gracias a la pantalla *touch* y los sensores de un smartphone, este es capaz de hacer las funciones de una interfaz de control para robots que no realizan tareas muy complejas [20].

Las últimas generaciones de *smartphones* se han diseñado con diversos sensores micro mecánicos integrados a modo de periféricos. Estas prestaciones han permitido el desarrollo de más aplicaciones que hacen uso de dichos sensores. Se destaca el estudio que realizó la *NASA* al integrar un smartphone con sistema operativo Android a una plataforma de experimentación. El prototipo llamado SPHERES (Figura 1.1) cuenta con procesadores de alto rendimiento, procesadores gráficos, cámaras de color, sensores de temperatura, luz, sonido, y una comunicación de banda ancha inalámbrica *WiFi*. Dicho experimento a bordo de la *ISS (International Space Station)* sirve como plataforma de experimentación para sistemas de navegación y acoplamiento de naves orbitales en desarrollo [18].

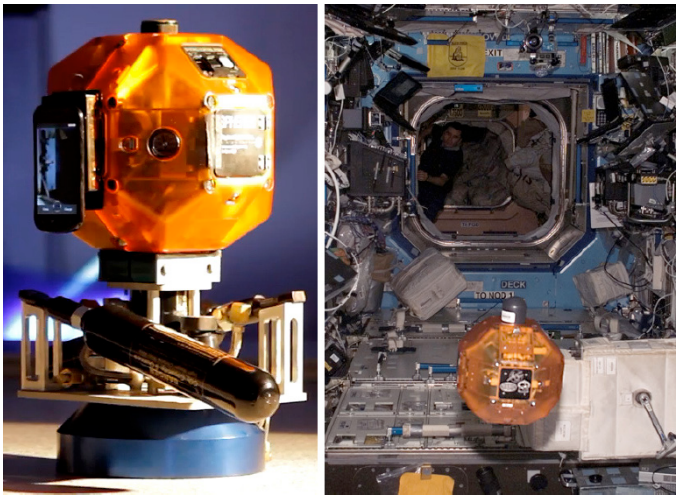

Figura 1.1: Prototipo SPHERES de la NASA

Los últimos trabajos se han enfocado a explotar los sensores como parte de la interfaz de usuario para aumentar la versatilidad y la facilidad en el control de robots. Un concepto que ha estado en desarrollo consiste en el uso del acelerómetro para capturar e identificar comandos por medio de gestos corporales. Se obtuvieron buenos resultados identificando tres comandos (pausa, continuar, detener) haciendo uso de los datos del acelerómetro para monitorear la los movimientos del smartphone e identificar los gestos que se realizan con el. Esto permitió una mayor versatilidad a la tradicional interfaz con botones en pantalla y se obtuvo mayor naturalidad en el uso, permitiendo que usuarios no expertos en robótica aprendieran a manejar robots móviles con un mínimo de instrucciones [22].

El interés por una interfaz hombre máquina en smartphone ha llegado a las áreas de uso militar para controlar robots en zonas de alto riesgo de manera simple [19] aprovechando la características de movilidad y reducido tamaño de los *smartphones*.

1.2. Motivación

La tendencia se ha enfocado al uso de los sensores del smartphone como entradas para identificar o generar comandos de control para un robot. Existen testimonios donde los usuarios declaran están más cómodos usando las manos con movimientos naturales, que botones en las reducidas pantallas del *touch*. Esta es la principal motivación para seguir con el desarrollo del concepto de la interfaz hombre máquina con *smartphones*.

En base de los logros obtenidos de los trabajos anteriores, se plantea que el siguiente paso a seguir es aumentar la naturalidad en la manera como se dan las ordenes a los robot haciendo uso de los recursos que ofrece actualmente un smartphone, enfocándose particularmente en el acelerómetro, giroscopio, *touch screen*, comunicación

WiFi y su alto desempeño de procesadores incorporados. Estos recursos no estaban tan avanzados en los *smartphones* hace pocos años y por consiguiente el campo de investigación es todavía muy nuevo. Esto da la posibilidad de crear innovaciones que tengan un impacto importante en el área de desarrollo de las interfaz hombre máquina.

Otro aspecto importante que se pretende modificar es el esquema tradicional de los sistemas para el uso de robots. En la práctica se tienen tres elementos Principales: interfaz, servidor y robot. El servidor se encarga del proceso numérico e interpretación de todos los datos recogidos tanto por el robot como de la interfaz y es necesario de disponer de una computadora exclusivamente para esta tarea. Si se logra prescindir del servidor incorporando sus tareas al smartphone, se lograra un sistema mucho mas sencillo, de comunicación directa y de mayor confiabilidad, aumentando la versatilidad, y reduciendo el costo del sistema entero.

El uso de la navegación inercial usando los sensores micro mecánicos ha sido un tema de interés en la práctica debido a que los sensores de éste tipo no cuentan con la precisión y calibración de sus homólogos de aplicación industrial y militar lo que compromete la precisión de la estimación de ésta técnica en la práctica. Debido a esta particularidad de los sensores micro mecánicos, se tienen una serie de problemas como por ejemplo el efecto de deriva, que es un error acumulativo causado por el ruido y falta de calibración de los sensores. Resolver este problema e implementar de forma satisfactoria la navegación inercial en una interfaz hombre-máquina permitirá la posibilidad de la aplicación de un smartphone como herramienta en la robótica.

Debido a todo lo mencionado en ésta sección se tiene que es necesario solventar varios problemas, como el filtrado de los ruidos de los sensores, su offset, los errores de estimación de la técnica de navegación inercial con sensores micro mecánicos, la precisión del calculo de la cinemática del robot, la comunicación y la interpretación de los comandos para lograr una interfaz práctica. Todos los procesos implicados se deben de realizar utilizando únicamente el hardware del smartphone sin apoyo de un servidor.

1.3. Planteamiento y Objetivos

El diseño de un smartphone es el de una plataforma de desarrollo para una gran diversidad de aplicaciones, como pueden ser juegos, redes sociales, negocios, comunicaciones e incluso cómputo de datos. Entonces es posible usar el hardware de manera muy diversa solo realizando una aplicación que se ejecute en el sistema operativo del dispositivo para que cumpla las tareas destinadas. Esto hace que la plataforma pueda ser usada como interfaz hombre-máquina sin necesidad de hacer modificaciones al hardware ni soporte de software adicional al que ya tiene disponible el sistema operativo.

Los actuales procesadores incluidos en los smartphone tienen una gran capacidad de cómputo pues están orientados al ambiente visual y al procesado de imágenes, audio

y video. Esta capacidad de cómputo es posible utilizarla para cálculos complejos tales como los que se requieren en el modelo de control de un robot en tiempo real. Actualmente en la interfaz hombre máquina propietaria se usan equipos de cómputo con menores prestaciones a la de un smartphone (procesadores no mayores a 200Mhz bajo una plataforma Windows 95 o microcontroladores a 20Mhz con software empotrado).

Los actuales *smartphones* contienen procesadores no menores a 1 Ghz y siendo incluso duales en los de alta gama. Esta característica permite que sea posible prescindir de un equipo de cálculo externo para controlar un robot.

La comunicación *WiFi* ya incluida en los *smartphones* permite que se realice la comunicación a otros dispositivos usando el protocolo *TCP-IP* y de igual forma se pueden usar las infraestructuras de comunicación *WiFi* existentes en el mercado.

Todas estas características benefician el desarrollo de un dispositivo que funcione como interfaz para su aplicación como interfaz hombre máquina con robots. En el caso de un robot manipulador que actualmente cuenta con una amplia aplicación industrial, este tipo de interfaz puede ser presentada como una solución. Pero el desarrollo plantea varios problemas.
Se pretende que la forma más natural para manipular un robot es usando los movimientos naturales del cuerpo sin la necesidad de aprender el uso de controles y botones que requieran de un conocimiento del funcionamiento del robot ni entrenamiento para adquirir una nueva habilidad, es decir que el robot sea lo más parecido a los movimientos del cuerpo humano.

En el caso particular de un brazo manipulador, se requiere conocer el ángulo de cada una de sus articulaciones para llegar a un punto dentro de su espacio de acción. Si éste robot mantiene las proporciones de un brazo humano, simplemente se necesita medir los ángulos de las articulaciones del operador y enviar las magnitudes a las articulaciones del robot.

Pero esto representa una desventaja desde el punto de vista de ingeniería debido a la estructura que debe ser usada para medir los movimientos de una extremidad humana. La manera más obvia de lograrlo es usando un exoesqueleto que se impulsa con la fuerza humana de las extremidades que se desean medir, integrando sensores de torsión y desplazamiento en cada una de las articulaciones. Toda ésta infraestructura física además debe de cumplir que sea ligera para que no requiera un esfuerzo considerable que evite que su uso sea cansado para el operador.

Otro método es consiste en conocer el punto deseado en el espacio para ubicar la herramienta del brazo y por medio de cinemática inversa calcular el ángulo de cada una de las articulaciones. El cálculo de la cinemática inversa tiene la ventaja que puede ser llevado por un programa de manera automática de manera que no se requiere intervención del usuario una vez determinadas las coordenadas en el espacio.

Una técnica que permite conocer el desplazamiento y dirección de un cuerpo cualquiera es la navegación inercial. Esta se ha usado ampliamente en naves para conocer su actitud (dirección y desplazamiento). Se propone el uso de ésta técnica para conocer los mismos datos de el brazo de un usuario por medio de los sensores micro mecánicos del smartphone. Pero esta técnica tiene los problemas de deriva ocasionada por errores acumulativos durante la estimación de la actitud debido a ruidos y errores de medición de los sensores. Los sensores micro mecánicos a pesar de sus bondades como tamaño, simplicidad y bajo coste, tienen grandes desventajas como el hecho de que no son calibrados en su manufactura y la falta de linealidad de las mediciones agravan el problema de la deriva en los resultados de la medición por estimación. Las técnicas de corrección de en la navegación inercial se apoyan en los datos provenientes de otro sistema como puede ser el GPS o algún otro sistema referencial.

El GPS comercial tiene una precisión de tres metros aproximadamente en condiciones óptimas, por lo que en distancias inferiores resultan inútiles los datos que pueda proporcionar. Para una interfaz hombre-máquina como el que se propone en esta tesis, donde el espacio de operación no supera en casos extremos a 1 metro cuadrado, el GPS resulta inaplicable como referencia para corregir errores de estimación. Debido a este motivo se deben buscar otras alternativas para solventar este problema.

El objetivo general de ésta tesis es desarrollar una interfaz hombre-máquina para smartphone usando sus sensores inerciales (acelerómetro y giroscopio) para implementar una unidad de medición inercial basada en la navegación inercial para calcular los movimientos de dirección y desplazamiento para detectar y medir gestos del brazo del usuario y así controlar remotamente a un brazo manipulador.

Para llevar a cabo el objetivo es necesario realizar lo siguiente:

1. Ensamblado de un brazo manipulador que servirá como hardware de pruebas de la interfaz.

2. Implementación en Java de la cinemática inversa del brazo manipulador de pruebas.

3. Implementación de algoritmos que permitan controlar el brazo robot manipulador con los datos de salida de la cinemática inversa traduciéndolos al lenguaje de operación del hardware.

4. Implementación de una unidad de medición inercial usando sensores micro mecánicos para determinar los problemas específicos relacionados a los sensores del smartphone.

5. Desarrollo de una variante de la unidad de medición inercial que permita solventar los problemas encontrados.

6. Implementación de una aplicación para Android que será la interfaz hombre máquina del brazo manipulador.

7. Obtener la precisión de estimación de ambas técnicas de navegación inercial para validar la propuesta.

1.4. Contribuciones

Un punto destacable de ésta investigación es la implementación de una interfaz de usuario para un robot, resolviendo los problemas relacionados en la estimación de la posición de la mano del usuario en el espacio haciendo uso de la información de los sensores inerciales para el uso de un brazo robot manipulador de forma inalámbrica.

La implementación de la interfaz haciendo uso solamente de la capacidad de cómputo del smartphone sin recurrir de una computadora adicional como apoyo, aportará la demostración de que es posible lograr que sea una interfaz móvil.

Durante el desarrollo de la unidad de medición inercial, los resultados de precisión deberán de demostrar que la interfaz es capaz de realizar tareas como lo hace un brazo humano, siendo posible captar los movimientos del brazo humano con un smartphone.

También se espera demostrar que la capacidad de cómputo de un smartphone y sus sensores son suficientes para cumplir con los objetivos de ser una interfaz barata, con alta disponibilidad, compacta y de un fácil manejo donde el usuario no requiere conocimientos técnicos sobre la operación de un brazo robot manipulador, pues todo el proceso lo realiza el software de manera transparente al usuario.

Adicionalmente se espera que con los resultados obtenidos del sistema se establezcan hipótesis que permitan el perfeccionamiento de ésta técnica en posteriores investigaciones.

1.5. Publicaciones

De las investigaciones realizadas en esta tesis se realizaron tres publicaciones a congresos los cuales fueron los siguientes:

- Carlos Parga, Xiaoou Li,Wen Yu *Tele-Manipulation of Robot Arm with Smartphone*, IEEE 6th International Symposium on Resilient Control Systems, Ago 13-15, 2013 in San Francisco, United States.
- Carlos Parga, Xiaoou Li, Wen Yu *Smartphone-based Human Machine Interface with Application to Remote Control of Robot Arm*, IEEE 10th International Conference on Electrical Engineering, Computing Science and Automatic Control, Sep 30 - Oct-4, 2013 in Mexico City, Mexico.
- Carlos Parga, Xiaoou Li, Wen Yu *Smartphone-based Human Machine with Application to Remote Control*, IEEE INTERNATIONAL CONFERENCE ON

SYSTEMS, MAN, AND CYBERNETICS Ago 13-16, 2013 in Manchester, UK.

1.6. Organización de la tesis

- Conceptos de interfaz hombre máquina y navegación inercial. Se establecen los conceptos de una interfaz hombre-máquina, navegación inercial y la revisión del estado del arte sobre su desarrollo en smartphones.
- Estructura de un HMI basado en smartphone. Se describe el problema de medir los movimientos del brazo humano usando una unidad de medición inercial como una interfaz hombre-máquina.
- Software y algoritmos para la realización de un HMI en un smartphone. Se describe la teoría y el desarrollo de la interfaz hombre-máquina, solucion al problema principal de la tesis y descripción de la implementación de todo el sistema.
- Resultados y análisis. Se exponen los experimentos y sus resultados que demuestran el funcionamiento de la interfaz además de obtener las características de funcionamiento.
- Conclusiones. Se expondrán las conclusiones de la investigación en base a los resultados obtenidos para posteriormente plantear algunas sugerencias en una futura continuación de la investigación.

Capítulo 2

Conceptos de interfaz hombre máquina y navegación inercial

El desarrollo de una interfaz hombre máquina que incorpora una unidad de medición inercial abarca diversos conceptos diferentes que operan en conjunto. En este capítulo únicamente se hará una revisión de los conceptos relacionados a la interfaz hombre máquina y la navegación inercial, así como los trabajos relacionados.

2.1. Sistema de navegación inercial

El sistema de navegación inercial es un dispositivo que se utiliza por las naves que viajan en el espacio aéreo, marítimo o en el vacío para conocer su ubicación y poder informar a la tripulación de a bordo, al control de tierra o a un sistema automatizado para poder navegar. En este capítulo se expondrá el principio teórico de su funcionamiento y los sistemas que se han estado utilizando haciendo énfasis en los sistemas basados en sensores micro mecánicos y los trabajos realizados con los smartphone como un sistema de navegación pues éstos últimos sistemas son los de mayor interés como antecedentes de información. La explicación del funcionamiento y su integración se detallarán en el capítulo que describe el software.

2.1.1. Concepto de navegación inercial

La navegación inercial es una técnica que usa la información de giroscopios y acelerómetros para estimar la velocidad y dirección de un objeto en movimiento. Los sensores están instalados en el objeto y son afectados por los movimientos del mismo. El principio de la inercia o primera ley de Newton es el fundamento de operación de los sensores inerciales y de ahí el nombre de los mismos y de la técnica que los emplea. La técnica requiere como entradas 6 datos que provienen de tres giroscopios ortogonales que miden velocidad angular y tres acelerómetros ortogonales que miden aceleración lineal. Esta configuración de los sensores permite así conocer aceleración

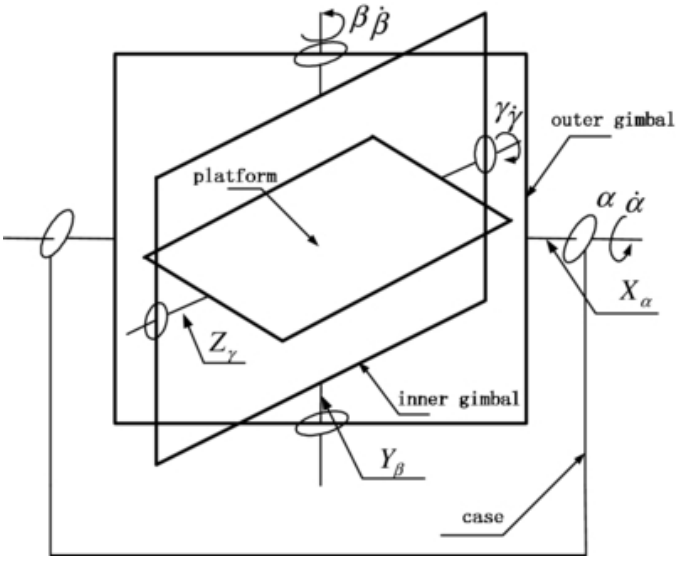

Figura 2.1: Sistema de cardanes de un sistema gimbaled

y cambios de dirección en los tres ejes del espacio aplicando cálculos de integración para conocer los desplazamientos y dirección.

Sistemas gimbaled y strapdown

En la práctica existen dos sistemas utilizados para llevar a cabo la técnica: el sistema gimbaled y strapdown.

El sistema gimbaled (Figura 2.1) tambien referido como plataforma estable aísla a los sensores de los movimientos rotacionales de la nave por medio de marcos sujetos por cardanes con que giran ortogonalmente entre sí manteniendo a los sensores siempre orientados paralelamente con el horizonte y ortogonalmente con la fuerza gravitacional haciendo uso de giroscopios para mantener la orientación. Este sistema permite leer directamente la dirección al medir los ángulos de los marcos entre sí y manteniendo el vector de fuerza gravitacional siempre en un solo eje haciendo simple su cancelación y calibración de las mediciones. El problema con este sistema es que aislar los movimientos rotacionales totalmente es imposible y eventualmente la fricción de los marcos en sus puntos de apoyo termina por desplazar el origen de la medición.

El sistema *strapdown* no tiene ningún sistema mecánico para aislar las fuerzas rotacionales y de aceleración ya que todos los sensores se fijan a la nave y son sometidas al mismo sistema de fuerzas. Sin embargo este método es más complicado ya que es necesario calcular la orientación de las fuerzas de aceleración en todo momento haciendo uso de la información de los giroscopios. Igualmente la información de los giroscopios necesita ser procesada para calcular la orientación. La ventaja de este sistema es la

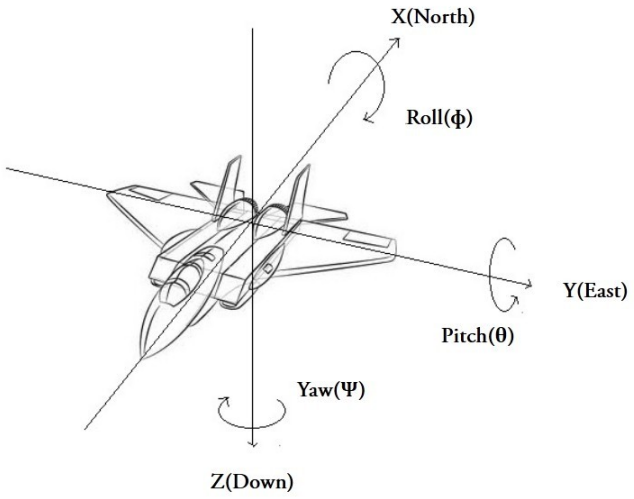

Figura 2.2: Sistemas de coordenadas de un sistema strapdown

simplicidad mecánica de sus sensores al no tener que estabilizar una plataforma por medios mecánicos.

Sistemas de coordenadas

Un sistema de coordenadas son vectores con un origen conocido para determinar las fuerzas que actúan sobre un cuerpo. Básicamente en la navegación se usan tres tipos de sistemas: origen en el centro de la tierra, origen en el centro de masa del cuerpo en movimiento. En los sistemas *strapdown* se toma como referencia el centro de masa de la nave (Figura 2.2), haciendo llamar al sistema de coordenadas como BCS(Body Coordinate System).

Ángulos de Euler

Para especificar la orientación de un sistemas de coordenadas se emplean los ángulos de Euler para indicar la actitud de un objeto. Los ángulos se definen como guiñada (*yaw* ϕ), cabeceo (*pitch* θ) y alabeo (*roll* ψ). Por convención se usarán los términos en ingles y es importante considerar que la referencia de las coordenadas corresponde al sistema *Body*, es decir, a los ejes de la nave y no del espacio donde se encuentra, correspondiendo así que el *pitch* es la rotación en el eje Y, *roll* es la rotación en el eje X y *yaw* en el eje Z. Todos estos giros pueden darse en sentido positivo o negativo.

Tipos de sensores

Se describirán brevemente los tipos de acelerómetros y giroscopio más comúnmente usados en los sistemas de navegación, pues si bien todos los acelerómetros o todos los giroscopios realizan la misma función, no es el mismo principio en el que se basan dando como resultado que cada tipo de sensor tenga características particulares en precisión, error, duración, peso, etc. Los sensores *MEMS* se detallarán por separado pues es el tipo de sensor en el que se basa en la investigación y se detallará mas extensamente.

- Tipos de giroscopios.
  - Mecánicos: Operan bajo el principio de conservación del momento angular de una masa circular que gira en su eje de masa y se resiste a los cambios de orientación. Se monta en un mecanismo que le permite mantener la orientación sin importar los cambios de dirección de la estructura que lo soporta mediante juntas giratorias o cardanes. Los sensores montados en los cardanes miden la posición angular directamente en el cual se encuentran. Si bien es un dispositivo confiable y preciso, tiene desventajas como es el peso y que prácticamente está compuesto por piezas mecánicas que genera fricción y requieren mantenimiento continuo, por lo cual los sensores más modernos lo han dejado en desuso, .
  - Opticos: Basados en el efecto Sagnac en donde haciendo uso de interferometría de anillo se obtiene un patrón de interferencia para medir la velocidad angular a la que es sometido el dispositivo. A estos sensores se les conoce como giroscopios de anillo láser y gozan de gran sensibilidad por lo que se utilizan en dispositivos que requieren resultados precisos. Este tipo de sensor es del tipo indirecto pues no miden la posición angular directamente, ya que su medición es la velocidad angular.
  - MEMS: Llamado giroscopio de estructura vibrante basan sus mediciones en las vibraciones de Coriolis. Este fenómeno se presenta en un elemento vibratorio que al someterlo a una rotación tiende a seguir vibrando en el mismo plano donde estaba originalmente ocasionando que aparezcan vibraciones ortogonales. La medición de este tipo de sensor como en los ópticos no es directa ya que miden velocidad angular. Este fenómeno es el que permite la fabricación de los sensores *MEMS* (Sistemas Micro Electro Mecánicos) ya que es posible la miniaturización del elemento vibratorio y haciéndolo muy barato de fabricar. Además es posible meter más de un solo sensor en un encapsulado de manera que se tienen tres giroscopios correspondientes a los tres ejes espaciales juntos.
- Tipos de acelerómetros.

- Mecánicos: Basados en la segunda ley de Newton, se componen de una masa suspendida por resortes en una estructura. La masa se desplaza por la fuerza causada por una aceleración aplicada en toda la estructura. Este desplazamiento es proporcional a la fuerza y en dirección opuesta a la aceleración. El desplazamiento es medido y la fuerza se calcula usando la ley de Hooke que se aplica en los resortes de suspensión.
- Piezoeléctricos: De amplia aplicación industrial, su principio se basa en el efecto piezoeléctrico de un elemento hecho de circonato de plomo. Una masa presiona con una fuerza al elemento piezoeléctrico creando una carga eléctrica proporcional. De la misma manera que en los acelerómetros mecánicos la fuerza ejercida es en dirección opuesta a la aceleración.
- MEMS: Su principio más difundido es el efecto capacitivo que varía con la distancia entre dos superficies. Este principio permite crear estructuras microscópicas con una masa de prueba que se desplaza por la fuerza ejercida debido a una aceleración como en el acelerómetro mecánico. La distancia de desplazamiento es medida por la variación de capacitancia.

2.1.2. Sistemas de navegación inercial con *MEMS*

Los sensores *MEMS* han tomado una importancia creciente por sus características de bajo costo y tamaño reducido logrando hacer diseños de sistemas de navegación más ligeros y baratos. Pero no solo las características a su favor contribuyen a sus diseños ya que también existen características que bajo ciertas circunstancias sea complicada la implementación de estos sensores en el diseño.

Ventajas

En general los sensores MEMS reúnen una cantidad de ventajas que los vuelven una atractiva opción en diseño de dispositivos de una variada aplicación. De las ventajas se pueden enumerar las siguientes:

- Tamaño reducido.
- Ligeros.
- Bajo consumo energético.
- Tiempo de encendido reducido.
- Fáciles y baratos de producir en grandes cantidades.
- Confiables.
- No requieren mantenimiento como los sensores mecánicos.
- Se pueden construir para condiciones extremas.

Desventajas

De los sensores inerciales existentes, los *MEMS* reúnen una serie de desventajas que deben de considerarse al diseñar un sistema basado en ellos. La desventaja principal es la falta de precisión, comparado con sistemas ópticos en el caso del giroscopio, o los sistemas piezoeléctricos en los acelerómetros. Dependiendo el tipo de sensor que se trate hay que tomar en cuenta los errores causados por sus deficiencias, pero en general se pueden mencionar las siguientes:

- Fluctuaciones en el *offset*. En el caso del acelerómetro, la aceleración gravitacional se detecta como un *offset* variable dependiendo de su orientación con el mismo. Este problema se trata con más detalle en el capítulo de implementación.
- Ruido blanco. También conocido como ruido blanco gaussiano es una señal aleatoria que no tiene una correlación estadística en su comportamiento y su función de densidad es una distribución normal. Esta señal se suma a la magnitud de aceleración original y provoca una desviación estándar en los cálculos de posición contribuyendo al efecto *drift* o barrido en la estimación de una posición.
- Termosensibilidad. La temperatura afecta directamente al sensor manifestándose como fluctuaciones en el *offset* en la señal de salida.
- Error de calibración. Los errores en la calibración pueden afectar de tres maneras al sensor: escala, alineación y linealidad.

Deriva o *drift*

La deriva es el desvío de la estimación de la actitud y desplazamiento de una nave en referencia a los valores reales causado por la suma sistemática de los errores de medición conforme transcurre el tiempo. Este es un problema al que siempre se han presentados los sistemas de navegación inercial, siendo los sensores la causa principal y donde más se ha trabajado la solución. También los modelos de estimación y los métodos de cálculo contribuyen a los errores en menor escala. Un ejemplo de un problema de cálculo relacionado con un sistema computacional es la resolución de los bits. Se han prepuesto diversas técnicas, algunas enfocándose en el tipo de sensores pero a pesar de todos los esfuerzos por hacer lo más preciso a un sistema, siempre se ha tenido la necesidad de basarse en una referencia para ir corrigiendo la deriva de tiempo en tiempo. En caso de las naves que viajan en el sistema solar usan el sol y las estrellas como referencias para la corrección. Con el desarrollo del GPS, la navegación en tierra utiliza este sistema para corregir la deriva.

2.1.3. Desarrollos actuales de navegación inercial con MEMS

Los sistemas de navegación inercial no es un concepto nuevo ya que la necesidad de un sistema de navegación y guiado automatizado nació junto con los cohetes en la

segunda guerra mundial. Estos primeros sistemas eran totalmente mecánicos logrando resultados prometedores. Sin embargo su perfeccionamiento se logró durante los 60´s con la necesidad de la automatización de la navegación de robots en el espacio y el control de los cohetes durante su lanzamiento. Los sensores inerciales mecánicos siguieron utilizándose pero se integró un sistema computacional para el cálculo de las trayectorias y la estimación de la orientación y posición de las naves logrando resultados que fueron lo suficientemente precisos para poder llegar a otros astros del sistema solar. Después de comprobar la robustez de éstos sistemas, se empezaron a desarrollar otros sistemas para la aviación militar y comercial que dieron la posibilidad de incorporar pilotos automáticos para el vuelo y guiado de misiles. Los sensores inerciales también fueron sofisticándose naciendo los sensores piezoeléctricos y ópticos ya descritos anteriormente que aumentaron más su precisión y durabilidad al no tener piezas mecánicas que sufrieran desgaste. Con la miniaturización de los sistemas mecánicos y electrónicos fue posible crear los sensores *MEMS*, los cuales si bien gozan de bondades únicas, también tienen problemas que hacen que su implementación sea limitada a las aplicaciones donde la precisión no es un factor estrictamente necesario.

El estudio de las aplicaciones de los sensores *MEMS* en sistemas inerciales ha sido amplio y se han diseñado varias estrategias para solucionar los problemas relacionados a sus desventajas. Para solventar el problema de los giroscopios *MEMS* y su baja precisión, Ching-Woo [24] propuso la utilización de 6 acelerómetros *MEMS* para diseñar un sistema de navegación para calcular los movimientos lineares y angulares de un cuerpo rígido. Los resultados del sistema indican que los errores son mayores que en un sistema convencional usando giroscopios y para corregir estos errores se sugiere utilizar el apoyo de un sistema GPS como una solución. El *GPS* es un sistema de navegación confiable ya que se basa en la posición de satélites para determinar las coordenadas y por consiguiente su uso como apoyo en la navegación ha aparecido recurrentemente para complementar la navegación inercial. Jan Wendel realizó un estudio de la aplicación de este tipo de sistemas en un helicóptero no tripulado operado a baterías [25]. Su trabajo plantea que sin los datos del GPS, los errores de la dinámica de la trayectoria no pueden ser eliminados resultando una imprecisa estimación del sistema. Incluso con la señal del GPS los datos son usados, según define el autor de una forma sub-optima y los desvíos de los datos del acelerómetro no pueden ser estimados y dando por resultado errores sistemáticos en las estimaciones del vehículo. Esta afirmación fue comprobada en los resultados de investigación donde se reporta que no es posible mantener la precisión de los datos de posición y velocidad debido a la pobre calidad en la precisión de los sensores inerciales MEMS. Especifica que los errores que se acumulados durante los primeros 10 segundos dan como resultado un error de 30 metros o mas.

Estos problemas relacionados con los *MEMS* en su aplicación en estimación de actitud ha sido motivo de realizar investigaciones sobre métodos para solventar los inconvenientes y aumentar la precisión de los sistemas. Uno de los problemas que han sido motivo de este tipo de investigaciones es la calibración de los sensores ya que por su costo no son calibrados en su fabricación. Isaac Skog [26] hace un estudio sobre

este problema y propone un método para llegar a una solución. Los resultados de la simulación del método fueron considerados aceptables para aplicaciones de bajo costo pero en la aplicación física de un dispositivo casero se detectó una desalineación de la estructura interna del sensor acelerómetro en los ejes X y Y que se reflejaron en los datos de las pruebas.

2.2. Interfaz Hombre-Máquina (HMI)

Una interfaz hombre-máquina o *HMI* (*Human Machine Interface en inglés*) se puede definir como una ventana de un proceso donde el usuario y el proceso interactúan. Esta interfaz o ventana puede encontrarse en un dispositivo a medida o en una computadora comercial.

2.2.1. Tipos de *HMI*

Existen dos tipos principales de *HMI*, los desarrollados a medida y los empotrados.

- *HMI* empotrados. Es el software que usan las funciones Standard de los sistemas SCADA. Están diseñados para realizar funciones específicas en sistemas especializados.
- *HMI* a medida. Es el software que se desarrolla en un entorno de programación gráfica. Se desarrollan generalmente para ser usados en un sistema operativo y hardware comercial pero controlan un hardware específico del fabricante.

Funciones de un *HMI*

Un *HMI* ejecuta varias funciones dentro de su marco de operación los cuales se definen como monitoreo, supervisión, alarmas, control y registros históricos.

- Monitoreo. Es la habilidad de obtener y mostrar datos de la planta en tiempo real. Estos datos se pueden mostrar como números, texto o gráficos que permitan una lectura más fácil de interpretar.
- Supervisión. Esta función permite junto con el monitoreo la posibilidad de ajustar las condiciones de trabajo del proceso directamente desde la computadora.
- Alarmas. Es la capacidad de reconocer eventos excepcionales dentro del proceso y reportarlos. Las alarmas son basadas en límites de control pre-establecidos.

- Control. Es la capacidad de aplicar algunos algoritmos que ajustan los valores del proceso y así mantener éstos valores dentro de ciertos límites. Control va más allá del control de supervisión removiendo la necesidad de la interacción humana. Sin embargo la aplicación de ésta función desde un software corriendo en una PC puede quedar limitada por la confiabilidad que quiera obtenerse del sistema.

- Históricos. Es la capacidad de muestrear y almacenar en archivos, datos del proceso a una determinada frecuencia. Este almacenamiento de datos es una poderosa herramienta para la optimización y corrección de procesos.

Historia de las *HMI*

El uso de las computadoras como *HMI* en sus inicios era bastante limitado y a medida que se han hecho progresos en la tecnología, éstas han proliferado en una diversidad de tareas hasta el punto de estar en todos lados y parecería que cualquiera es capaz de manejarlas (al menos, en sus funciones básicas).

Inicios

Si se considera como punto de partida a la primer computadora electrónica ENIAC en 1943, se puede decir que estas computadoras no tenían nada que hoy se reconoce como *HMI*. Tanto instrucciones como datos eran introducidos directamente a las ubicaciones de memoria al iniciar la ejecución a través de tarjetas perforadas, y eran leídos de los registros del procesador, mostrándolos directamente en un volcado binario, hacia tarjetas o cintas perforadas, que debían ser traducidas a algo legible empleando dispositivos mecánicos independientes.

Teletipos y terminales

El primer avance resultó en una dirección obvia, pero facilitó tremendamente tanto el uso como el aprovechamiento de los resultados: La interfaz textual. No se trata aún de una pantalla, sino de la adecuación del teletipo, híbrido de teclado e impresora, que comenzó su existencia como un reemplazo más ágil y confiable que el código Morse para la comunicación a larga distancia. El teletipo permitía ingresar programas mucho más complejos a memoria, lo cual llevó a que naciera y popularizara un concepto: el de los archivos. Aparecieron los primeros editores (obviamente, mucho más espartanos de lo que conocemos hoy, y orientados a trabajo línea por línea), y como consecuencia directa, los programas pudieron comenzar a presentar una mayor complejidad, llevando a la introducción de bibliotecas de código y a las diversas abstracciones y estrategias para manejar la complejidad.
La transición del teletipo a la pantalla no es tan simple como podría parecer. Dejando de lado la mera complejidad técnica (relativa al estado del arte de la época) de crear

dispositivos independientes y de relativo bajo costo capaces de mantener comunicación con la computadora central generando la imagen en pantalla del texto que iban recibiendo, lo cual implicaba que tuvieran una memoria interna, aunque mínima para estándares modernos. Las ventajas de tener terminales con cierto grado de inteligencia se hicieron obvias y comenzaron a aparecer terminales con diferentes estándares capaces de reposicionar el cursor o de desplegar texto con atributos (negritas, subrayado, colores), caracteres semi-gráficos, hasta verdaderas capacidades de formularios como las que manejaban las IBM 3270, que comenzaron a permitir desacoplar la lógica de un programa de su presentación tal como hoy lo vemos en los navegadores Web.
Las terminales además fueron centrales para la aparición de computadoras multitarea/multiusuario. Quienes prefieren utilizar sistemas Unix utilizamos como una de nuestras herramientas más poderosas al emulador de terminal. Si bien las terminales como producto de hardware hace mucho tiempo que ya no existen para propósitos prácticos, la interfaz de línea de comandos programable permite un grado de expresividad tan rico que no ha podido ser reemplazado por ningún otro medio.

WIMP: Window, Icon, Menu, Pointer

En diciembre de 1968, en los laboratorios de Palo Alto de Xerox, Douglas Engelbart presentó la que al día de hoy se conoce como la madre de todas las demos: La introducción de la interfaz gráfica básica que se sigue utilizando hoy en día, manejada a través de un apuntador controlado por un *Mouse*, presentando ventanas para la visualización de las diferentes aplicaciones en uso o vistas de una misma aplicación, iconos representando atajos a las acciones de sistema disponibles y con un menú como elemento estándar presentando las acciones relacionadas con cada aplicación activa. También para la entrada y manipulación de datos dentro de cada aplicación el dispositivo primario seguirá siendo el teclado.
La demostración de Engelbart incluía ejemplos de aplicaciones verdaderamente revolucionarias en esa época, como la videoconferencia, el correo electrónico, el hipertexto o un editor colaborativo de tiempo real.
Aunque ha habido refinamientos sucesivos y grandes cambios en la parte estética de esta propuesta para las computadoras de uso general seguimos utilizando este esquema de interacción con más de cuarenta años de antigüedad.

Interfaces de propósito acotado

Posiblemente el mayor cambio en las interfaces de usuario viene de que, cada vez con mayor fuerza, tenemos dispositivos con gran poder de proceso de cómputo sin un formato de computadora de propósito general. No es casualidad que hoy existan las interfaces innovadoras presentes en teléfonos celulares o consolas de videojuego, se esta llegando al punto en que se van encontrando formas muy convenientes de

interactuar con computadoras de propósito acotado, aunque éstas no sean adecuadas para aquellas de propósito general.

Con la generación actual de consolas de videojuegos, Nintendo se anotó el mayor éxito al introducir su Wii: Una consola de relativamente bajas prestaciones y muy inferior a su competencia en las áreas en que típicamente competían, la capacidad gráfica. Sin embargo, su apuesta más fuerte fue hacia una interfaz novedosa: Los controles basados en acelerómetros, que permitieron modelar diferentes actividades como nunca antes se habían presentado en videojuegos.
Por otro lado, el iPod de Apple introdujo una interfaz largamente prometida, basada en una pantalla táctil de tamaño reducido, y orientada a equipos destinados al entretenimiento, a la consulta rápida de información, y especialmente popularizada a través del teléfono aparecido poco tiempo después. Sin embargo, si bien esta interfaz ha resultado natural para una gran cantidad de personas, resultaría indudablemente impráctica y antiergonómica para una computadora de propósito general.

La tendencia es que se sigan creando nuevas interfases dedicadas a tareas específicas, las más exitosas serán sin duda las más transparentes mejorando nuestra vida sin requerir que los usuarios estén conscientes siquiera en su existencia.

Otras ideas

Se han propuesto muchas otras propuestas de interfaces para computadoras de uso general, sin embargo todavía no han madurado suficiente o sólo son aplicables en contextos muy específicos.

- Pantallas táctiles. Desde mediados de los 1980, Hewlett-Packard introdujo su línea de computadoras HP110, con una pantalla sensible al tacto. Esta interfaz prometía ser más ágil y natural que el *Mouse* (que requiere un nivel de coordinación no trivial). Y si bien esta interfaz tuvo un moderado éxito en áreas como los kioscos (cajeros automáticos, estaciones de servicio), nunca fue del todo aceptada para uso en computadoras de propósito general por lo poco ergonómico que resulta tener que levantar la mano constantemente para apuntar la pantalla.

- Reconocimiento de voz. La ciencia ficción de los 1970 (Computadora HAL de Odisea 2001, Arthur C. Clark) presentó a la voz como la principal forma de interacción hacia la computadora. Se han ensayado interfaces de reconocimiento de voz, pero su uso todavía es limitado, principalmente por la dificultad que presenta el lenguaje humano. Además de esto, fuera de dar comandos puntuales, "dictar" un texto a la computadora no es una tarea trivial. Al redactar un texto, el proceso normal que seguimos implica ir hacia atrás y hacia adelante,

corrigiendo el texto, reemplazando y reformulando las frases. Una interfaz de dictado debe distinguir el texto de las órdenes, lo cual requerirá un entrenamiento complejo.

- Manipulación 3D. Presentar la información como objetos del mundo real manipulables a través de guantes o gestos parece muy atractivo. Un ejemplo muy cuidadosamente desarrollado de una interfaz basada en estas ideas se aprecia en la película Minority Report. El poder de procesamiento y el hardware especializado para hacer este tipo de manipulaciones sin embargo no justifica, al día de hoy, el costo que significaría. Hay aplicaciones para las que este costo sí se justifica. En México la Dirección General de Servicios de Cómputo Académico de la UNAM cuenta con la computadora especializada Ixtli para simulación y visualización. Existe muchas más ideas en el tintero y en los años por venir seguro se acumularán. Sin embargo a pesar de las innovaciones que se han dado en dispositivos de uso específico, particularmente en el caso de las interfaces para interactuar con computadoras de uso general no ha habido cambios substantivos en los últimos 40 años.

2.2.2. Smartphone, sensores inerciales y HMI

Los smartphone son dispositivos de muy reciente introducción al mercado, y vinieron a revolucionar el cómputo en muchas de sus áreas. Pero los HMI no aparecieron de forma inmediata ya que primero tuvieron que madurar los smartphone como tecnología antes de que fueran plataformas atractivas para desarrollos más especializados.

Primeros smartphone

El primer iPhone tenía como propósito el de comunicaciones interpersonales como telefonía y mensajería y ya contaban con la arquitectura completa de una computadora (Sistema operativo, aplicaciones, procesador, memoria, periféricos y almacenamiento). Aunque su sistema operativo era monoproceso, ya era programable y se podían crear aplicaciones adicionales que se podían instalar en él. Aunque esta arquitectura era bastante primitiva si se comparaban con las PC disponibles, su éxito radicó en las bondades que el hardware ofrecía y una GUI que operaba con la pantalla táctil.
Así fue como el iPhone introdujo el concepto de smartphone y cómputo móvil. El equipo debe ser pequeño y liviano para transportarse, pero en el momento de usarlo debe ser suficientemente grande para poder operarlo con facilidad. A medida que fue evolucionando, otras compañías introdujeron al mercado sus propias líneas de smartphone. El hardware de cada producto era diferente a el primer iPhone, pero sus características eran muy similares. En cuanto al software, iPhone manejaba su propio sistema operativo de Apple, mientras que Nokia introdujo el Symbian en sus

productos, y el resto decidió probar con un nuevo sistema operativo desarrollado por Google, el Android.

El hardware y el software siguieron una rápida evolución. El hardware ha ganado más poder de cálculo y almacenamiento, sin sacrificar su consumo, mientras el software aprovechando los adelantos del hardware se han desarrollado sistemas operativos multiproceso y aplicaciones más complejas ya no solo destinadas a la comunicación. En la actualidad un smartphone de alta gama tiene el mismo poder de cómputo que una PC de hace 10 años.

Revisión de desarrollos

Desde la aparición del smartphone se pudo apreciar su potencial para la aplicación en robótica, ya que como se ha mencionado, es una plataforma de cómputo integral, es decir que incluye procesador, memoria, almacenamiento y monitor, la cual no es necesario modificar su hardware debido a que incluye conectividad WiFi.

- Desarrollos de HMI con PDA. Aunque las PDA no son consideradas como un smartphone, si comparten muchas características, y fueron de las primeras en usarse para experimentar la funcionalidad de éstos dispositivos como HMI.
  Un caso concreto es el de Jong Hyun Park cuando en el 2008 fig2.3 [21] con una PDA desarrollo un HMI para un robot móvil. Estos dispositivos no poseían un significativo poder de proceso, de tal forma que se planteó usar el PDA como una simple HMI donde se mostraba la imagen captada por la cámara del robot móvil e información adicional por medio de colores y números sobre la imagen y también se tenían botones para controlar el avance y dirección. Pero para el procesado de la información se tenía una PC a modo de servidor que se encargaba de los complejos cálculos que se requerían para que las órdenes e información entre el robot y la PDA se realizaran.
  Este caso se menciona como un antecedente de desarrollo a la intención de simplificar las HMI y las PDA ya permitían explorar éstas posibilidades, pero las PDA no prosperaron ya que dieron paso a los *smartphones*.

- Desarrollos de HMI con smartphone para vehículos de transporte. Los vehículos de transporte también se encuentran dentro de las áreas que son de interés para demostrar la funcionalidad de las HMI basadas en smartphone. Cristiano Suelta en el 2010 [2] propuso un sistema de monitoreo para motocicletas usando el sintetizador de voz integrado en el sistema operativo para enviar y recibir mensajes hablados del manos libres via bluetooth al conductor sobre el estado de el vehículo y otras órdenes.
  El smartphone, como parte central del sistema, se comunica con la motocicleta

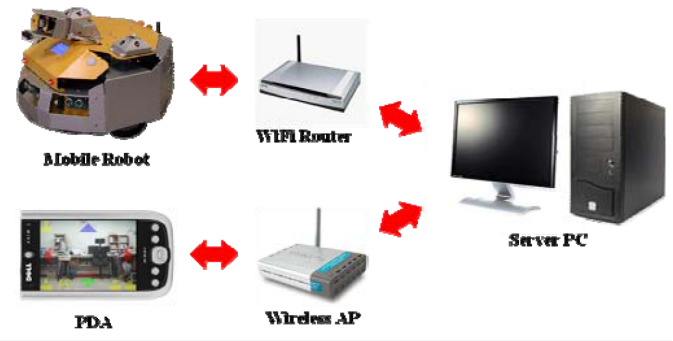

Figura 2.3: HMI basado en PDA de Jong Hyun Park

vía bluetooth a una interfase para el protocolo CAN del sistema propietario de la motocicleta.
Una novedad con este sistema es aprovechar la comunicación de datos 3G para conectarse a un servidor, no para procesar datos sino, para enviar datos sobre la posición y estado del vehículo en tiempo real.
Es denotable que ésta propuesta agrega una funcionalidad mas a las HMI basadas en smartphone, como es la comunicación con la nube.
La nube en últimos años ha demostrado su valía para muchos de las tareas cotidianas, y otra de las aplicaciones propuestas por Byung-Hyung Lee en el 2011 [9] es el de una HMI que permita no solo controlar en tiempo real un vehículo de motor, si no que tambien permite conocer la posición GPS del mismo y mostrarla en el mapa.
El vehículo es operado por la plataforma OSGi (Open Service Gateway initiative) la cual se comunica por TCP hacia el smartphone el cual es usado como HMI principal hacia el usuario.

- Desarrollos de HMI con smartphone para discapacitados. Otra demostración de los alcances de las HMI basadas en *smartphones* es relacionada al área biomédica. En 2011 Jeonghee Kim desarrolló una interfaz para operar una silla eléctrica con un smartphone [7].
  La HMI consistía básicamente de un detector magnético de diadema, el cual era sensible al movimiento de algún objeto magnético. Dichos sensores se ubicaban a cada lado de la boca del usuario para detectar el movimiento de un piercing magnético incrustado en la lengua, con la cual el usuario realiza movimientos específicados que posteriormente se convertirán en órdenes. Los sensores acoplados a la diadema envían las señales generadas vía inalámbrica hacia un iPhone.
  El iPhone es la parte central del diseño, ya que a parte de ser la HMI de todo el sistema, se encarga del proceso de interpretación de las órdenes y comunicación con la silla eléctrica. En su pantalla se muestra información sobre los sensores y opciones de calibración.
  Para la comunicación con la silla el iPhone utilizaba el protocolo UART hacia

la interfase propia de la silla donde se envían los comandos para hacer que ésta se mueva.
Cabe mencionar que para este desarrollo no se requirió un servidor adicional para el proceso de los datos, si no que se aprovecharon los propios recursos del iPhone, pues su poder de cálculo era suficiente.

- Desarrollos de HMI con smartphone para robots móviles. El interés del desarrollo de HMI en *smartphones* se ha enfocado más para el control de robots móviles. Ya se han desarrollado ideas experimentales para demostrar la funcionalidad de los *smartphones* como HMI remotos de diferentes formas.

  Hui-Kyung Oh [13] propone un HMI trabajando en la plataforma Android para controlar un robot móvil Lego Mindstorms NTX. Básicamente el sistema se centra en un servidor de control que se comunica vía Bluetooth con el robot y vía WiFi con el smartphone. El HMI del smartphone recibe el video de una webcam empotrada en el robot para tener una vista en la perspectiva del robot para poder ser controlado por un usuario remoto el cual envía comandos para controlar los movimientos del mismo por medio de botones programados en pantalla. La actitud resultante del robot es un una operación híbrida pues el robot puede detectar obstáculos por medio de un sensor ultrasónico y con esta información al encontrarse el robot con un obstáculo, automáticamente el servidor retira el robot del mismo 10cm hacia atrás.

  Otro trabajo muy similar es el propuesto por Saso Koceski [22], Fig 2.4 donde de igual manera utilizan un smartphone con Android para controlar un robot Lego Mindstorms NTX con dos servidores para procesar los datos, uno que ejecuta el programa de control del robot y procesa los datos visuales de las cámaras para enviarse al HMI del smartphone, y otro que reconoce guiños recogidos por el smarphone con el acelerómetro y los traduce a comandos para el primer servidor de control del robot. Esta propuesta va más allá del primer trabajo pues no solo se limita a botones en una GUI en pantalla ya que el trabajo explota el sensor acelerómetro incluido en el smartphone para poder controlar el robot. Se proponen tres movimientos, uno de pausa, uno de continuar y otro de parada. El prototipo terminado fue sometido a usuarios no familiarizados con el mismo, para observar la facilidad con que pudieron operar el robot, se les permitio intentarlo algunas veces y la conclusión del trabajo fue que a los usuarios les fue mas natural el HMI para interactuar con el robot.

  Una característica de los trabajos anteriores es el hecho de que se requiere un servidor dedicado para interpretar los comandos del HMI y calcular los movimientos del robot, es decir que el HMI del smartphone es simplemente una terminal. Esta característica se consideraría trivial en prototipos y ambientes de laboratorio, pero en la práctica orientada para un usuario final cada elemento

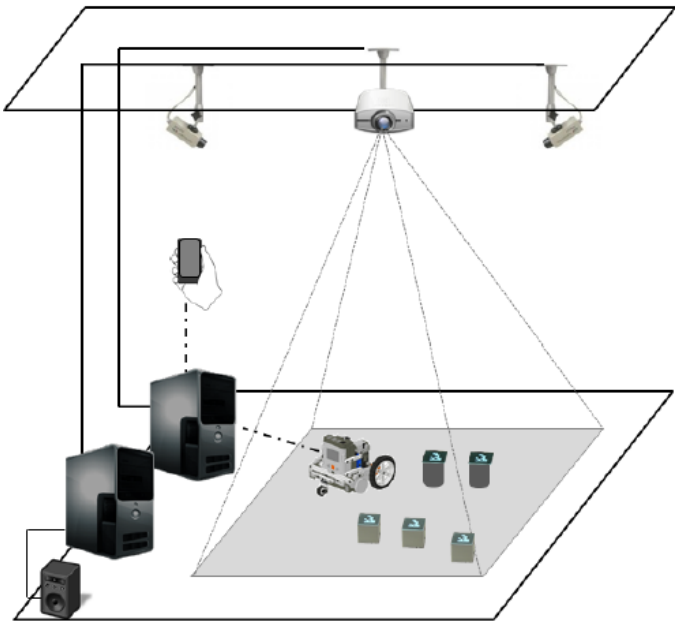

Figura 2.4: HMI con reconocimiento de guiños de Saso Koceski

extra implica costo y conocimiento extra.

Eliminando el servidor de control distribuyendo las tareas del mismo entre el robot móvil y el smartphone, el sistema smartphone-robot será tan práctico como un juguete de radiocontrol donde todo es totalmente portátil ya que pueden operar con baterías sin una infraestructura dedicada para su operación en un ambiente controlado.
Fausto Ferreira en el artículo publicado en el 2011 [20] Figura 2.5 propone varias tendencias para aumentar las capacidades de los USV (*Unmanned Surface Vehicles*) y una de sus propuestas es el uso de los smartphone como una HMI que sustituya a las HMI de consola para situaciones particulares donde no es requerido un control muy complejo de un USV. Su uso es limitado respecto a una consola debido a las limitadas capacidades de visualización de datos con la reducida pantalla de un smartphone, pero la idea no deja de ser atractiva debido a que dichos dispositivos incluyen una pantalla *touch*, es de bajo costo y opera con baterías. La *HMI* se conecta al robot por la *WiFi* incluida en el smartphone. Otro detalle mencionado es la portabilidad de la aplicación ya que al ser desarrollada para Android es posible utilizarla en diferentes *smartphones* que usen la misma plataforma, haciendo posible que pueda ser enfocado para usuarios no profesionales con el sistema.

Otros dos trabajos, aunque no tan sofisticados como el de Fausto Ferreira, son los de Sung Wook Moon [11] y Hou-Tsan Lee [12] donde proponen un

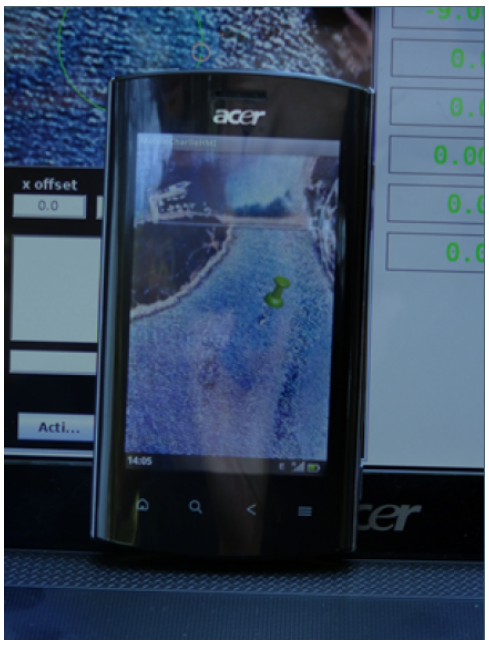

Figura 2.5: HMI con smartphone de Fausto Ferreira

sistema *smarphone* HMI-robot sin un servidor intermediario. En ambos casos la comunicación se realiza vía *WiFi* donde el robot envía imágenes desde su perspectiva y recibe comandos para avanzar, retroceder o cambiar de dirección. En el caso del trabajo de Sung, se enfocan sobre el control y monitoreo en tiempo real y su retardo en todo el proceso. En cambio Hou-Tsan se enfoca en la facilidad de uso del robot con solo la información enviada por la IPCam empotrada en el robot y la facilidad de uso de los comandos mostrados en pantalla.
Ambos trabajos demuestran que es posible prescindir de un servidor de control pero es posible hacer uso de mas recursos incluidos en un smartphone comercial como es el caso de los sensores como lo demuestra el trabajo de Saso Koceski.

Capítulo 3

Estructura de un HMI basado en smartphone

Para implementar la *IMU* y realizar pruebas controladas sobre el funcionamiento de la solución propuesta, se ensambló un sistema que permita controlar un brazo robot manipulador con el smartphone.

El sistema está compuesto por diferentes elementos físicos que realizan una función específica. Estos componentes son el smartphone, el robot, servidor y la red inalámbrica que pueden definirse como subsistemas.

En la Figura. 3.2 se muestran físicamente cada uno de los componentes del sistema, o subsistemas los cuales son cuatro:

- smartphone. Esta es la parte principal del *HMI*. Actualmente estos dispositivos tienen una variada cantidad de sensores que pueden ser leídos por las aplicaciones, pero para esta aplicación en específico solamente se usará el acelerómetro y el giroscopio para obtener el comportamiento de los movimientos del usuario. Con el uso de algunos algoritmos se obtienen tres posiciones (x,y,z) y tres rotaciones, guiñada (yaw), cabeceo $(pitch)$ y de alabeo $(roll)$. Este conjunto de datos son trasformados en ángulos que corresponden a cada junta del robot por una cinemática inversa simplificada donde finalmente son enviados al servidor.

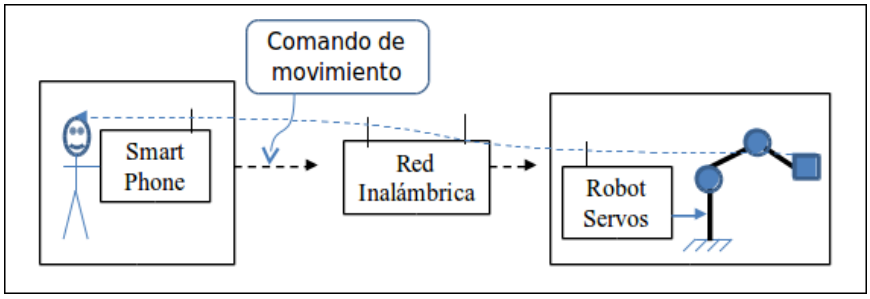

Figura 3.1: Estructura principal del HMI aplicado en un brazo manipulador

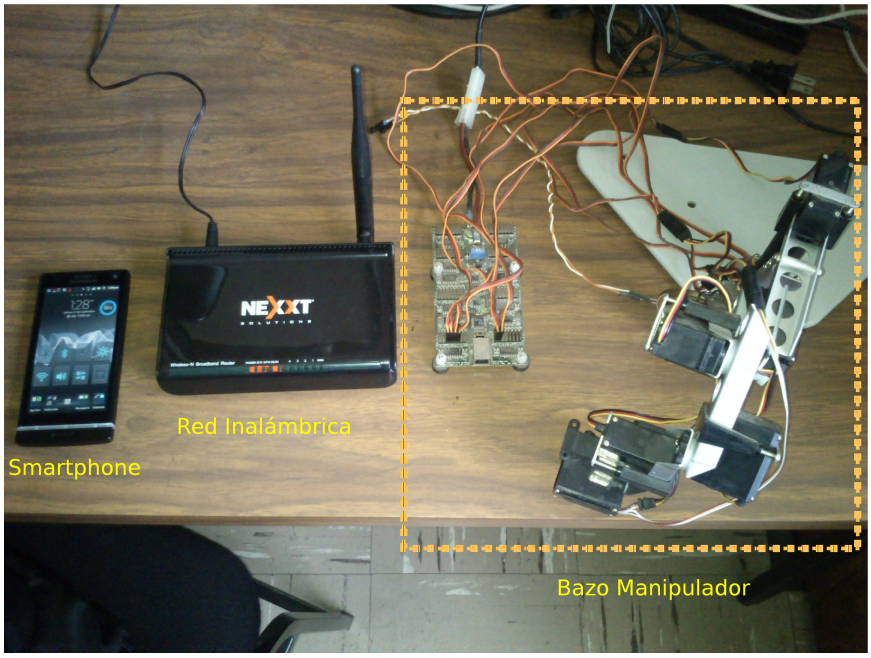

Figura 3.2: Vista de la infraestructura

También el smartphone contiene la interfaz gráfica de usuario *GUI* con la que el usuario puede controlar algunos comandos de reinicio y monitorear los datos que se generan en algunas etapas del proceso.

- Brazo manipulador. Este subsistema tiene un controlador para los servos que mueven la estructura del brazo. El sistema tiene un algoritmo proporcional integral derivativo *PID* estándar embebido para controlar el movimiento de los servos y mantenerlos en el ángulo indicado por la aplicación en el smartphone. Como el brazo tiene 5 juntas con sus servos se define entonces que es un robot con 5 grados de libertad (5-DOF).

- Servidor. Es una *PC* convencional donde está instalado el software del controlador del robot y hace la comunicación del mismo con la red inalámbrica. No realiza ningún post-procesado de la información ni contiene algún algoritmo exceptuando los que pertenecen a los controladores del hardware.

- Red inalámbrica. Este subsistema está compuesto por el ruteador inalámbrico, la interfase *Wi-Fi* en el smartphone y la interfase *Wi-Fi* en el servidor. El protocolo de comunicación es *TCP (Transmisión Control Protocol)* bajo el estándar de comunicación inalámbrica IEEE802.11n.

El flujo de datos a través de los subsistemas se muestra en la Figura. 3.3

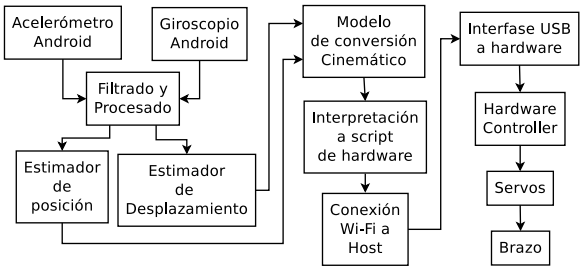

Figura 3.3: Flujo de datos en el sistema

3.1. Brazo Manipulador

Un robot es una entidad virtual o mecánica artificial. En la práctica, esto es por lo general un sistema electromecánico que, por su apariencia o sus movimientos, ofrece la sensación de tener un propósito propio. La independencia creada en sus movimientos hace que sus acciones sean la razón de un estudio razonable y profundo en el área de la ciencia y tecnología. La palabra robot puede referirse tanto a mecanismos físicos como a sistemas virtuales de software, aunque suele aludirse a los segundos con el término de *bots*. No hay un consenso sobre qué máquinas pueden ser consideradas robots, pero sí existe un acuerdo general entre los expertos y el público sobre que los robots tienden a hacer parte o todo lo que sigue: moverse, hacer funcionar un brazo mecánico, sentir y manipular su entorno y mostrar un comportamiento inteligente, especialmente si ese comportamiento imita al de los humanos o a otros animales. Actualmente podría considerarse que un robot es una computadora con la capacidad y el propósito de movimiento que en general es capaz de desarrollar múltiples tareas de manera flexible según su programación; así que podría diferenciarse de algún electrodoméstico específico.

Un brazo manipulador o brazo robótico se puede definir como el conjunto de elementos electromecánicos que propician el movimiento de un elemento terminal (*gripper* o herramienta). La constitución física de la mayor parte de estos manipuladores guarda cierta similitud con la anatomía de las extremidades superiores del cuerpo humano, por lo que, en ocasiones, para hacer referencia a los distintos elementos que componen al robot, se usan términos como: cintura, hombro, brazo, codo, muñeca, etc.

Una especificación general de un brazo robótico comprende: sus grados de libertad, su configuración y su cinemática directa e inversa [7,9]. Estas especificaciones son dadas desde el diseño propio de cada robot y su aplicación. Hay una clasificación de robots manipuladores la cual presenta las diferencias de diseño, precisión, precio, etc. [23]

Para éste trabajo se empleó un robot comercial Robix mostrado en la Figura 3.4, el cual contiene un kit de servos, base, eslabones, controlador y accesorios. La estructura que se diseñó fue un brazo robot manipulador con 5 grados de libertad (5-DOF) Figura. con una pinza de sujeción como herramienta. Además tiene dos direcciones, rotación y cabeceo.

Figura 3.4: Brazo Robot Manipulador

Los servos están conectados a un controlador el cual se encarga de hacer la conexión con el software del servidor a través de algún puerto USB y además proporciona la corriente de trabajo de los mismos. El controlador puede ser configurado desde software (número de servos, velocidad, posición máxima y mínima) para el funcionamiento del mismo.

3.2. Servidor

Formalmente un servidor es un nodo que provee servicios en la red. En éste caso específico hospeda el software que se comunica con el controlador de los servos del robot y además permite comunicación por red a los clientes que ejecutan la biblioteca provista por el fabricante para Java. Esta biblioteca es un acceso remoto a las funciones del software para poder mandar instrucciones por medio de líneas *string* de *script* propio del software. Adicionalmente el software permite la configuración del controlador de parámetros específicos de los servos. Por último también éste software se encarga de la gestión del puerto USB para la comunicación con el controlador. El servidor en sí mismo no realiza tareas adicionales como post-procesado de la información ni ejecuta algún control adicional.

La ventaja del uso de este software es que se usa una computadora que ejecute Windows y permite una configuración fácil realizando un proceso de instalación estándar de una aplicación Windows.

3.3. Red Inalámbrica

La red inalámbrica permite la comunicación entre el smartphone y el servidor del robot. La ventaja de esto es que ambos componentes se comunican con el protocolo TCP sin importar de la infraestructura física en la que se encuentre ($LAN\ o\ WLAN$) de tal forma que es transparente para los equipos y no se requiere ningun cambio en el

software de los mismos. Esto se traduce a que fue posible usar un equipo comercial *Wi-Fi* de bajo costo y altas tazas de transferencias de datos sin mayores especificaciones en sus características siempre que cumplan con el standard IEE802.11 para permitir la comunicación con el smartphone.

3.4. Smartphone HMI con Unidad de Medición Inercial

Smartphone

Actualmente un smartphone es una plataforma de cómputo móvil completa, pues tiene memoria, almacenamiento, sistema operativo, dispositivos de entrada y salida y periféricos. Adicionalmente éstos periféricos embebidos son sensores (luz, *touch screen*, micrófono, acelerómetro, giroscopio, etc) y dependiendo de cada fabricante incluye más o menos de éstos sensores pero en su mayoría comparten los mismos. También en sus periféricos están los de comunicación los cuales son la red de datos (2G y 3G), los de telefonía celular, *Wi-Fi* y *bluetooth*. En los dispositivos de salida se tienen una o más bocinas, una pantalla a color y un motor para generar vibraciones. Además por definición un smartphone opera a baterías y no requiere conexiones físicas exceptuando para alguna transferencia de datos ocasional vía USB o para recargar su batería.

En sí la plataforma con todas éstas características ofrece muchas ventajas pues actualmente tienen procesadores con una velocidad promedio de 1GHz, memorias mayores de 256MB y almacenamientos de 16GB en promedio. Todo esto permite que el smartphone ejecute tareas al igual que una computadora personal simplemente instalando o programando las aplicaciones compiladas para el mismo.

El software que se desarrolló para ésta aplicación usa dos sensores: acelerómetro y giroscopio. El acelerómetro es utilizado para obtener las magnitudes (en m/s^2) del vector de aceleración causado por el movimiento que el usuario da al smartphone. Por lo tanto se tienen tres medidas (x, y, z) como se muestra en la Figura. 3.5 El eje X es paralelo a los lados horizontales de la pantalla, y un movimiento hacia la derecha del smartphone da lugar a una magnitud positiva, mientras un movimiento hacia la izquierda da una magnitud negativa. El eje Y es paralelo al lados verticales de la pantalla, y un movimiento hacia arriba del smartphone da lugar a una magnitud positiva, mientras que un movimiento hacia abajo da una magnitud negativa. Por último el eje Z es perpendicular a la pantalla de forma que si se sostiene el smartphone en vertical y de frente, se tiene que un movimiento hacia adentro da lugar a una magnitud positiva, y un movimiento hacia fuera da una magnitud negativa. Se entiende entonces con la descripción anterior que el origen de los ejes del acelerómetro está en el centro de la pantalla.

El giroscopio del smartphone tiene de forma análoga al acelerómetro tres ejes pero

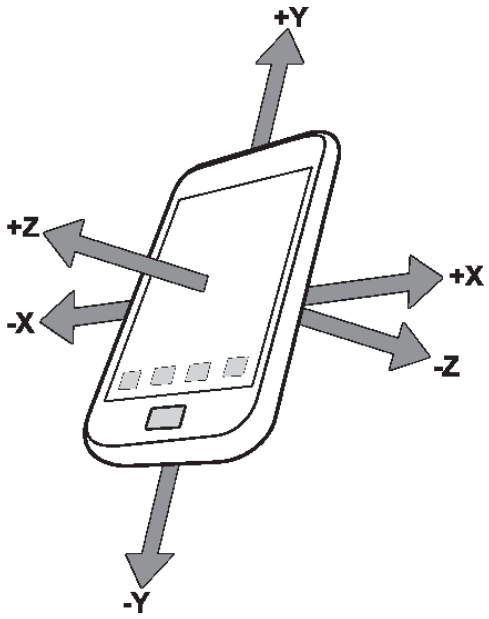

Figura 3.5: Disposición de los ejes del acelerómetro

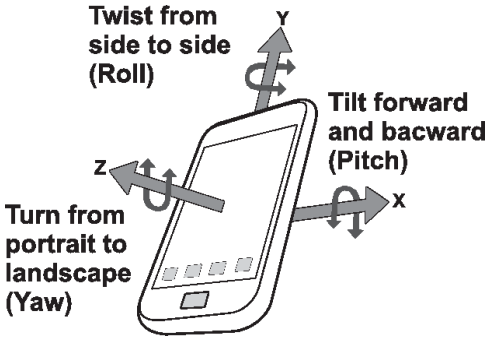

Figura 3.6: Disposición de los ejes del giroscopio

éstos son de rotación en vez de aceleración, por lo tanto sus magnitudes son de velocidad angular (rad/s) en cada uno de sus ejes (x,y,z) Figura. 3.6

Un giro en sentido inverso del reloj da lugar a una magnitud positiva, mientras que un giro en sentido de del reloj da una magnitud negativa. Los ejes están dispuestos de la misma forma que el del acelerómetro.

Estos dos sensores son usados en conjunto para estimar el desplazamiento y la dirección relativa del smartphone con el cual el software tendrá los datos de dirección y posición para ser procesados por la cinemática inversa.

Otra de las funciones que realiza el smartphone es el de monitoreo y control del programa con comandos como iniciar el proceso o reiniciar los parámetros. También la información mostrada proporciona información sobre los datos que se van calculando en cada una de las etapas del proceso. Todo esto se detalla con más detalle en el

capítulo 4.

Unidad de Medicion Inercial

La unidad de medición inercial o *IMU (Inertial Measurement Unit)* es un dispositivo que aplica la técnica de navegación inercial descrita anteriormente en el capítulo 2 para medir la aceleración, velocidad, posición y orientación de un aparato.

La *IMU* es el componente principal de los sistemas de navegación inercial usados en los aviones, naves espaciales, buques y misiles guiados entre otros. Los datos recolectados por los sensores de una *IMU* permiten a un computador seguir la posición del aparato usando un método conocido como navegación por estima.

El problema más notable que se presenta en todas las *IMU* es que son afectadas por un error acumulativo, esto es causado por que el sistema de guía está continuamente agregando los cambios de posición a las posiciones previamente estimadas provocando que cualquier error en la medición sea acumulando. Este error acumulativo crea un efecto llamado 'deriva' *(drift)*. Debido a éste problema las *IMU* no son componentes únicos en los sistemas de navegación y operan conjuntamente con otros sistemas para corregir los errores por deriva que se tiene invariablemente. Los sistemas auxiliares son el GPS, sensores de gravedad, sensores de velocidad externos, etc.

Requerimientos de una *IMU* para capturar movimientos corporales

La captura de movimientos corporales usando una *IMU* tiene como objetivo, como su nombre lo indica, de capturar los movimientos de alguna parte del cuerpo de forma tridimensional, además de la orientación de igual forma tridimensional. Existen ventajas de utilizar una *IMU* para capturar gestos corporales sobre la captura de gestos solo detectando cambios específicos en los sensores. La *IMU* está diseñada para poder medir velocidades y posiciones en el espacio a partir de la aceleración dando la ventaja de poder conocer la magnitud y dirección exacta de algún gesto. De forma ideal es posible hacer el seguimiento tridimensional de alguna extremidad sin necesidad de sensores mas complicados reduciendo la complejidad y costo de desarrollo.

Pero una *IMU* destinado a medir movimientos corporales tiene requerimientos diferentes al de una *IMU* utilizada en la navegación. En esta tesis la extremidad de interés es el brazo humano, por lo que a continuación solo se va a tratar únicamente de las particularidades relacionadas al comportamiento de dicha extremidad A continuación se explicarán en que consisten:

- Espacio de operación. El espacio de operación de una *IMU* para detectar los movimientos de un brazo está limitado por el espacio de operación del mismo. Esto significa que el espacio esta claramente delimitado y los movimientos tienen un máximo, al igual que los cambios de dirección debido a la cinemática del brazo humano que depende de su estructura mecánica.

- Corrección de errores. La corrección de errores no puede ser gestionada como en los sistemas de navegación convencionales usando el GPS. La causa de ésta limitación es por el espacio de operación de la *IMU* ya que, el *GPS* tiene una resolución de 3 metros en el mejor de los casos, mientras que un brazo humano requiere resolución al menos de centímetros. Adicionalmente el GPS no realiza la ubicación de forma tridimensional. Para esta tesis se plantea el uso del sistema *IMU* sin un sistema adicional de corrección de errores.

Teniendo en cuenta los anteriores requerimientos se usará un smartphone como hardware de una *IMU* para detectar los gestos realizados por los movimientos del brazo humano para controlar un brazo robot manipulador y demostrar que es posible su implementación sin necesidad de hardware adicional.

Análisis de una *IMU* con un smartphone

Para poder determinar las consideraciones que se deben tener para que sea posible detectar y medir los movimientos de un brazo humano con un smartphone operando como una *IMU* es necesario conocer los problemas que se presentan con el mismo.

El algoritmo para una *IMU* esta ya establecido pero el smartphone cuenta con sensores *MEMS* que como se explicó anteriormente, tienen consideraciones particulares que el algoritmo convencional *IMU* probablemente no solucione completamente.

Para poder obtener el comportamiento de una *IMU* con datos reales, se capturaron datos reales del acelerómetro de un smartphone para hacer una simulación del algoritmo en *MATLAB* y hacer posible el análisis gráfico.

El comportamiento teórico ideal de una *IMU* captando el movimiento el movimiento de un brazo está representado en la figura 3.7. Se representa primero un estado de reposo donde en determinado momento se tiene una aceleración donde el brazo empieza su movimiento. El algoritmo de la *IMU* inmediatamente empieza a estimar la velocidad que se ve incrementada por la aceleración y a su vez la posición del mismo empieza a cambiar. Posteriormente la aceleración decrece hasta que la aceleración se vuelve negativa provocando que la velocidad empiece a decrecer hasta que tanto la velocidad y la aceleración convergen en cero y esto detiene el movimiento totalmente por lo que la posición deja de cambiar ahora siendo una línea recta, lo que representa que el brazo llegó a su destino y se ha detenido en una posición concreta estimada por la *IMU*. A partir de ese momento no existe ninguna variación la aceleración, velocidad y posición.

Con los datos capturados del acelerómetro de un smartphone se obtiene la aceleración para distintas condiciones de operación de la *IMU* en la simulación en *MATLAB*. El primer caso que se estudia es el comportamiento del *IMU* con el smartphone en reposo. En la Figura 3.7 se ve la grafica de la *IMU* y puede verse claramente como el *drift* se manifiesta por el ruido residual del filtro que se aplica a la señal y el offset del acelerómetro. Este es un fenómeno que en la práctica representa un problema ya que no se puede determinar si el smartphone se encuentra en posición estática.

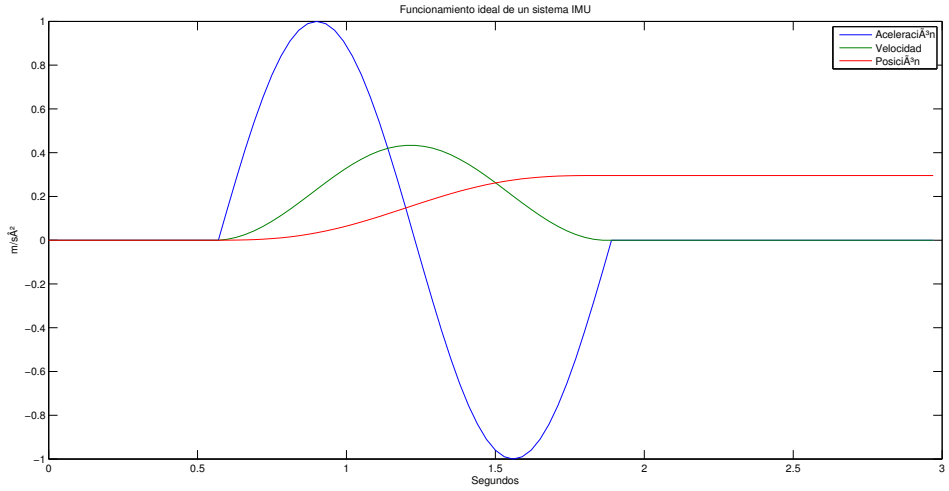

Figura 3.7: Gráfica mostrando el funcionamiento ideal de un algoritmo *IMU*

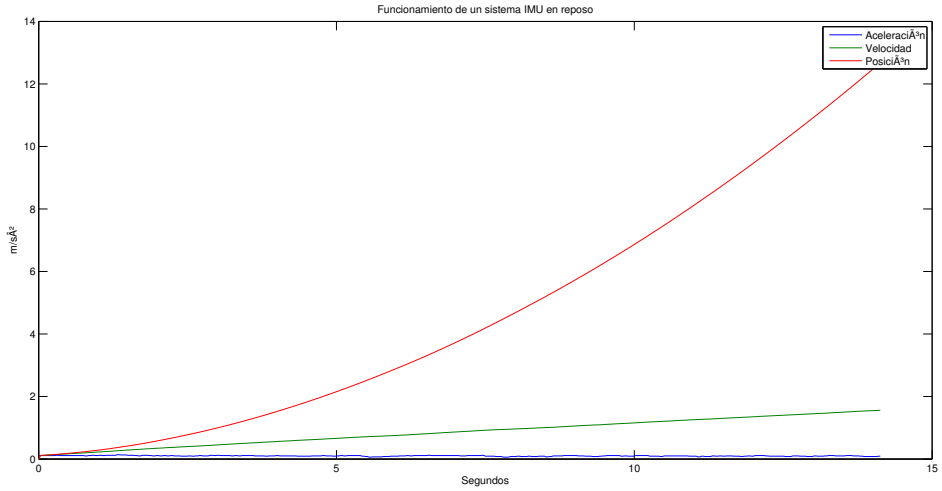

Figura 3.8: Gráfica mostrando el funcionamiento real de un algoritmo *IMU* convencional en posición estática

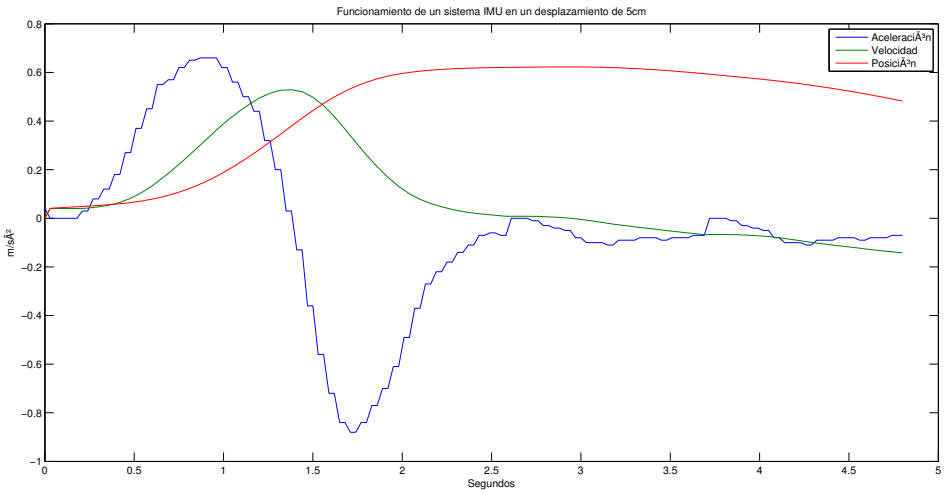

Figura 3.9: Gráfica mostrando el funcionamiento real de un algoritmo *IMU* convencional en un desplazamiento muy corto

Otro caso a tener en cuenta es la sensibilidad de la *IMU* y su resolución, pues como se comentó en los requerimientos de la *IMU* que se pretende implementar, se debe poder detectar y medir movimientos pequeños con una resolución de centímetros. En la Figura 3.9 se muestra el comportamiento de la *IMU* durante un desplazamiento real de 5cm. Es claramente visible el comportamiento de la *IMU* respondiendo al movimiento, pero el problema causado por el *drift* observado durante el reposo sigue presente en este caso. Este fenómeno hace que al final del trayecto en vez de tener líneas rectas como en el caso ideal de funcionamiento, la *IMU* presenta un corrimiento gradual en la estimación del desplazamiento. Sin embargo el algoritmo si fue capaz de detectar el movimiento y estimarlo.

Por último se tiene la estimación de un movimiento máximo aproximado. El comportamiento observado en la Figura 3.10 presenta el mismo problema observado anteriormente de error acumulativo causando el *drift* en las estimaciones.

Captura de gestos corporales

Los gestos son parte de la comunicación corporal no verbal y son movimientos voluntarios o involuntarios del cuerpo humano, como movimiento de ojos, manos y piernas. Para capturar el movimiento de las manos y poder medir estos desplazamientos se requiere de sensores que puedan capturar el tipo de movimiento y su magnitud. En el caso concreto de los gestos utilizados para mover un robot que simule los movimientos de un brazo humano, estos deben poder comunicar los comandos necesarios que permitan realizar todos los movimientos necesarios para manipular objetos por medio del robot.

Estos gestos pueden definirse como:

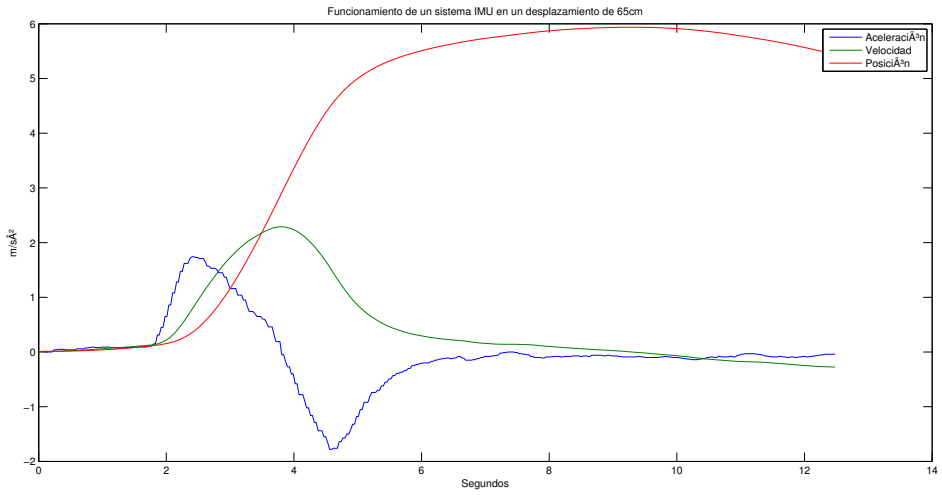

Figura 3.10: Gráfica mostrando el funcionamiento real de un algoritmo *IMU* convencional en un desplazamiento de 65cm

- Desplazamiento en X
- Desplazamiento en Y
- Desplazamiento en Z
- Giro en X
- Giro en Y
- Giro en Z

Para capturar y medir estos movimientos es necesario el uso de sensores inerciales, siendo el caso del acelerómetro en los desplazamientos y el giroscopio para los giros.

Capítulo 4

Software y algoritmos para la realización de un HMI en un smartphone

Debido a la poca información referente a la precisión que se puede obtener implementando una unidad de medición inercial en un smartphone, es necesario primero recopilar información sobre el comportamiento de sus sensores. Para ésta tarea se capturara información de los sensores, filtros y modelos de navegación inercial para examinarla mediante graficas hechas en MATLAB. Se explicara el desarrollo de una interfaz hombre máquina para ejecutarse en un smartphone con sistema operativo Android donde se implementará un algoritmo alternativo para la estimación de actitud con el objetivo de mejorar la precisión, y junto con la cinemática inversa que controlará a un brazo manipulador vía Wi-Fi con se obtendrán los datos de precisión para validar la aplicación de la navegación inercial en una interfaz hombre máquina.

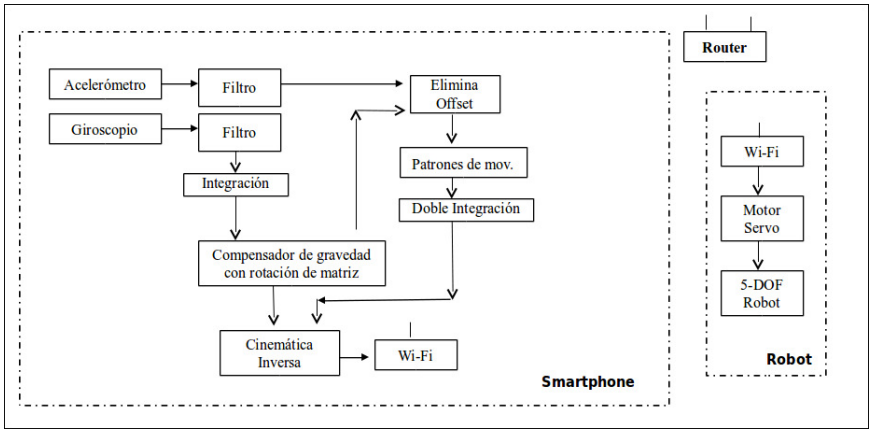

Figura 4.1: Estructura principal del software del HMI

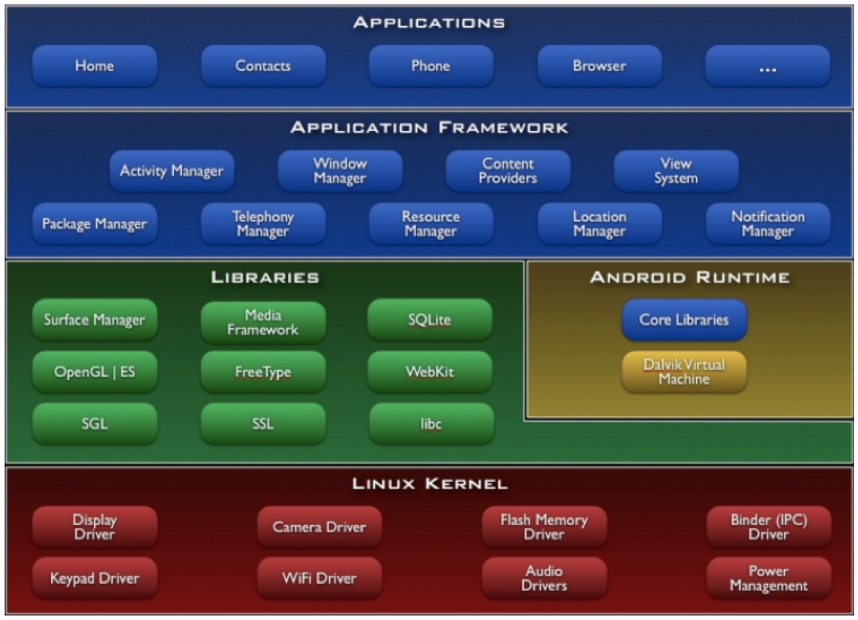

Figura 4.2: Estructura del sistema operativo Android

4.1. Sistema operativo Android

Android es un sistema operativo para dispositivos con pantalla táctil basado en Linux desarrollado por Open Handset Alliance. Su principal característica distintiva con otros sistemas operativos es que su código es abierto y se encuentra disponible bajo licencia de Apache. Su estructura básicamente se compone de aplicaciones ejecutadas en un *framework* de *Java*.

4.1.1. Estructura del S.O. Android

Debido a que los smartphones son prácticamente computadoras, es decir que tienen entradas, salidas, almacenamiento y periféricos, es posible usarlas como plataformas de cómputo multipropósito. Para poder gestionar el hardware y los recursos es necesario contar con un sistema operativo tal como ocurre en una computadora de escritorio, por lo que Android no difiere mucho en lo que estructura se refiere, a otros sistemas. Se tratará de explicar brevemente a continuación la estructura de Android.

Núcleo

Android está basado en la versión 2.4.16 del *kernel* de Linux. La tarea básica del *kernel* es administrar el acceso al hardware del sistema actuando como una capa de abstracción. Para tener el correcto acceso a los periféricos se encuentran implementados todos los controladores primarios (cámara, USB, teclado, sensores, micrófono, gps, Wi-Fi...).

Bibliotecas

El siguiente nivel en la estructura de Android son las bibliotecas. Estas se encargan de las funcionalidades básicas de las aplicaciones. Estas están escritas en C/C++ y pueden ser accedidas directamente desde los programas de la plataforma. Las bibliotecas más importantes son:

- Librerías en C del sistema: Implementaciones de las librerías estándar de C, preparadas para un mejor funcionamiento en dispositivos embebidos que usan Linux.
- Librerías multimedia: Proporcionan soporte para grabación y reproducción de audio y vídeo, así como de imágenes.
- Gestor de superficies: Maneja el acceso al dispositivo de pantalla.

Librería del núcleo Web: Moderno motor de navegación Web que posibilita usar el navegador de Android así como vistas incrustadas de los sitios Web.

- SGL: El motor gráfico 2D.
- Bibliotecas para 3D: Implementación basada en las API de OpenGL ES 1.0.
- FreeType: Renderización de mapas de *bit* y gráficos vectoriales.
- SQLite: Proporciona soporte para base de datos relacionales.

Máquina virtual Dalvik

Es la máquina virtual encargada de ejecutar los programas. Optimizada para requerir poca memoria y está diseñada para permitir ejecutar varias instancias de la máquina virtual simultáneamente, delegando en el sistema operativo subyacente el soporte de aislamiento de procesos, gestión de memoria e hilos.

La máquina virtual ejecuta dos tipos de archivos:

- APK: Comprimido con ejecutables y datos del programa.
- DEX: De Dalvik ejecutable, es un archivo similar a .class de Java pero post procesado para conseguir un mejor rendimiento en la máquina Dalvik.

La máquina virtual tiene acceso a los *Core Libraries* las cuales son simplemente las bibliotecas con las clases estándar de Java.

Application Framework

Módulos escritos en Java que permiten interactuar con la agenda de contactos, la mensajería, llamadas, etc.

Aplicaciones

Todas están escritas en Java pero tienen acceso directo a bajo nivel a través de la API. Están armadas en base a cuatro bloques de construcción:

- *Activity*: Parte visual de la aplicación.
- *Intent Reciver*: Un intento, traducción al español de *intent*, sería algo así como un evento genérico que sería independiente de los programas, por lo que en caso de pedirse Android buscaría la aplicación adecuada para servir dicho intento. Por ejemplo podría ser algo así como enviar sms.
- *Service*: El concepto de demonio en Linux aplicado a Android.
- *Content provider*: Proporciona a la aplicación capacidad de interactuar de manera sencilla con la parte interna de Android.

4.1.2. Android frente a otros S.O.

Directamente se definirá la razón por que se ha escogido Android para desarrollar la aplicación de los algoritmos de estimación de actitud de la navegación inercial.

Al principio del capítulo quedó establecido que Android es un sistema operativo que pertenece a la tecnología abierta en el cual una gran cantidad de fabricantes diseñan sus productos para que cumplan con el estándar de arquitectura para usar éste sistema operativo, y el software esta hecho por una gran cantidad de colaboradores, un fenómeno muy parecido al que ocurre con Linux.

Las ventajas que ofrece la tecnología abierta es que su uso es gratuito y su código está disponible al público para modificaciones y aportaciones con aplicaciones. Esta característica sumada a opción de escoger entre varios fabricantes de hardware fue el motivo por el cual se decidió utilizar ésta plataforma para realizar el desarrollo del sistema de HMI para el robot ya que se escogió el mejor hardware que ofrecía todas las características necesarias. Adicionalmente la información sobre la programación del software es ampliamente difundida y discutida en una gran cantidad de foros y *blogs*, lo que permite hacer una búsqueda sobre problemas específicos en programación y hardware.

4.1.3. Biblioteca para Java Rascal

El robot incluye software para ser instalado en Windows, junto con *drivers* y una biblioteca para java (.jar) que puede ser agregada a proyectos para controlar el robot (Código B.1. Esta biblioteca ya incluye conectividad TCP/IP lo cual permite la posibilidad de comunicarse con el *host* de forma remota con aplicaciones java. Esto significa que Android puede reconocer aplicaciones donde se agregue ésta biblioteca para conectarse al *host*.

Para mover los servos del brazo es necesario interpretar a un *script* en formato de *string* los ángulos de los servos, y a través de métodos de la biblioteca de Rascal este *script* es interpretado por la controladora del robot. Debido a que el formato de los datos es incompatible con las instrucciones propias de java con la cual está desarrollada la aplicación es necesario hacer la traducción de las instrucciones utilizando estructuras predefinidas de cadenas con una variable que se traduce a *string* desde una variable entera calculada desde la aplicación.

Se debe tomar en cuenta que los parametros de los servos no son en unidades de grados o radianes por lo que es necesario hacer la conversión de unidades de los valores.

4.2. Sensores inerciales del smartphone

Los sensores en un smartphone son como los periféricos en una PC y pueden ser accedidos haciendo uso del administrador de sensores ya existente en el sistema operativo. Específicamente se hace una instancia del Sensor Manager que se encuentra ubicado en el *Application Framework* el cual se encarga de la gestión de los sensores y entregando los datos de forma periódica de los mismos en un valor numérico flotante en unidades específicas según el sensor para ser utilizadas. El uso de sensores con Android presenta la ventaja sobre un sistema hecho a medida, que no es necesario desarrollar la electrónica para instrumentar los sensores ni es necesario hardware adicional ya que todo se encuentra dentro del diseño del smartphone, es decir que el smartphone se puede trabajar como si de una plataforma de desarrollo se tratara. Igualmente la programación presenta la ventaja que el software puede ser usando en diferentes smartphone de forma transparente gracias al framework sin necesidad de hacer modificaciones. Pero también se presenta una desventaja frente al hecho de que el hardware no puede ser reemplazado por piezas específicas, siendo de interés los sensores los cuales ya vienen desde su diseño y no pueden ser reemplazados por otros de diferentes características para aplicaciones concretas. En ésta tesis se trata de resolver los problemas de deriva en las estimaciones de actitud debido a la imprecisión de los sensores MEMS del smartphone por medio de algoritmos. Tratar de caracterizar el sensor específico de un smartphone por medio de su hoja de especificaciones es inútil pues cada modelo de cada fabricante incluye sensores diferentes y en su mayoría de los casos se desconoce el modelo específico del sensor. El método a seguir consiste

en caracterizar el sensor del smartphone elegido para pruebas y tomarlo como base. Para poder mostrar como se van tratando los datos del sensor y analizar los detalles de cómo se obtiene la estimación de actitud, se realizaron 4 tipos diferentes de movimientos y se graficaron. A continuación se describen los movimientos:

- Estático: El smartphone se encuentra sobre una superficie firme sin que sea afectado por ningún tipo de movimiento.
- Movimiento controlado: El smartphone está sobre un riel donde se puede controlar la dirección de su desplazamiento asegurando que es solamente en un eje.
- Movimiento natural: El smartphone es movido por la mano del usuario, tal y como es sometido en condiciones normales de uso.
- Giro: El smartphone es girado en uno de sus ejes por la mano del usuario, tal y como es sometido en condiciones normales de uso.

Los datos presentados son solo la información de un solo sensor pues se estudia en esta sección específicamente el comportamiento del mismo y no el funcionamiento del sistema inercial completo en tres ejes.

4.2.1. Acelerómetro

Por definición, el acelerómetro es un instrumento para medir aceleración. En otras palabras el acelerómetro mide los cambios de velocidad del cuerpo donde está instalado. Se definirá primero el funcionamiento teórico del acelerómetro para posteriormente tratar el comportamiento del acelerómetro MEMS específico del smartphone.

Planteamiento teórico

El acelerómetro más simple mide la fuerza de una masa que aplica a un dinamómetro cuando cambia su velocidad y de acuerdo a la segunda ley de newton se puede conocer la aceleración del cuerpo.

$$F = ma$$

Donde F es la fuerza medida por el dinamómetro, m es la masa del cuerpo y a es la aceleración.

Si se despeja la aceleración, entonces un acelerómetro puede ser descrito como:

$$a = F/m$$

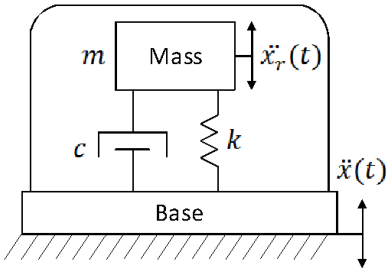

Figura 4.3: Modelo mecánico de un acelerómetro

Pero en un acelerómetro no solo se mide la aceleración de la masa de prueba en el dinamómetro pues también se mide la aceleración del cuerpo al que está empotrado, tal como el modelo de la Figura 4.3

De ésta manera se expresa que la fuerza inercial de la masa de prueba está dada por

$$F = m(\ddot{x}(t) + \ddot{x}_r(t))$$

Donde $\ddot{x}(t)$ es la aceleración actuando en el acelerómetro y $\ddot{x}_r(t)$ es la aceleración relativa de la masa de prueba respecto a la base o cuerpo donde está empotrado. Se tiene entonces que la dinámica del acelerómetro (fig) sujeta a la aceleración $\ddot{x}(t)$ se obtiene con la segunda ley de newton

$$m\ddot{x}_r(t) + c\dot{x}_r(t) + kx_r(t) = -m\ddot{x}(t)$$

Donde k es el coeficiente de rigidez del resorte, c es el coeficiente de amortiguamiento y m es la masa de prueba. La deformación debido a la aceleración es sensado y convertido a un equivalente en señal eléctrica usando diferentes métodos como es el capacitivo o inductivo teniéndose que

$$\ddot{x}_r(t) + 2\zeta\omega\dot{x}_r(t) + \omega^2 x_r(t) = k_a\ddot{x}(t)$$

Donde k_a es la ganancia del acelerómetro, ζ is la constante de amortiguación y ω es la frecuencia de resonancia. Despejando ζ y ω se tiene que $\zeta = \frac{c}{2\sqrt{km}}$ y $\omega = \sqrt{\frac{k}{m}}$. En resumen la expresión de la salida del acelerómetro dice que

$$a(t) = k_a\ddot{x}(t) + w(t) + d$$

Donde $w(t)$ es el ruido, d es el offset de la aceleración gravitacional. El ruido $w(t)$ debe ser tratado y reducido con filtros posteriormente dependiendo de la aplicación.

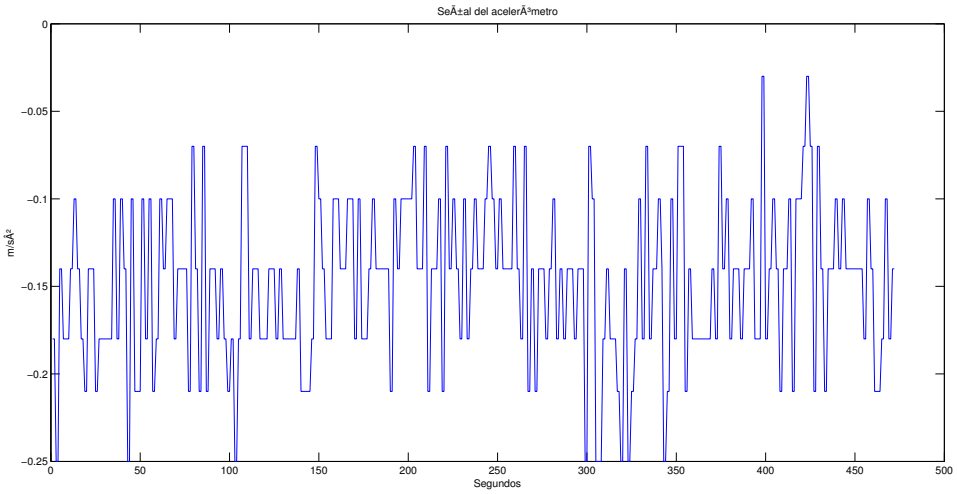

Figura 4.4: Señal del acelerómetro en posición estática

El acelerómetro MEMS del smartphone

En la Figura 4.4 se muestra la señal de salida del acelerómetro del smartphone en posición estática. Se aprecia claramente que existe una gran cantidad de ruido blanco gaussiano en la señal de la que se debería esperar una línea recta sin variaciones. Este un grave problema pues el algoritmo de estimación de actitud integra éste ruido como movimiento creando el efecto de deriva, ya que con el tiempo se va incrementando el error de manera progresiva aunque no exista un movimiento que afecte mecánicamente al acelerómetro.

El ruido que se presenta en el acelerómetro en posición estática igualmente se presenta durante el movimiento y suma a la señal de aceleración alterando los datos provocando errores de estimación. En la Figura 4.5 se observan los datos de un movimiento controlado y es claramente visible como el ruido está presente durante todo el trayecto del movimiento.

Las dos observaciones anteriores están hechas en una situación controlada, pero cuando se somete el smartphone a un movimiento natural también existe otro tipo de perturbación que se añade al ruido blanco gaussiano del acelerómetro. Los humanos no hacemos movimientos perfectos y tendemos a temblar de manera natural, un fenómeno muy conocido por los cirujanos. Durante un movimiento éstas variaciones provocadas por el temblor natural de la mano son percibidas de igual forma por el acelerómetro y se suman al ruido blanco gaussiano del sensor, causando que la información de un movimiento sea muy distorsionada. En la Figura 4.6 se observa un movimiento natural. Se puede comparar que la distorsión del movimiento es claramente mayor que el movimiento controlado en un riel.

Es evidente que os datos del acelerómetro que se obtienen directamente de él deben de ser tratados por un filtro antes de poder ser usados para una estimación de actitud.

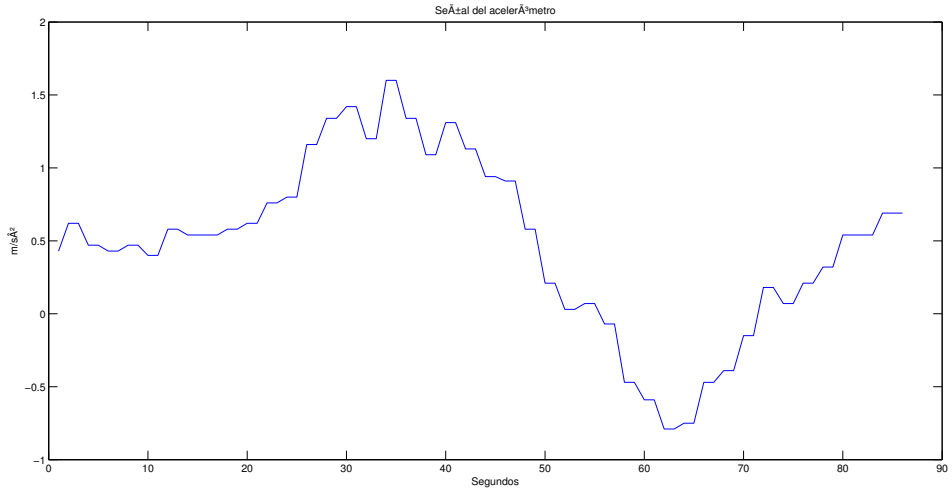

Figura 4.5: Señal del acelerómetro en movimiento controlado

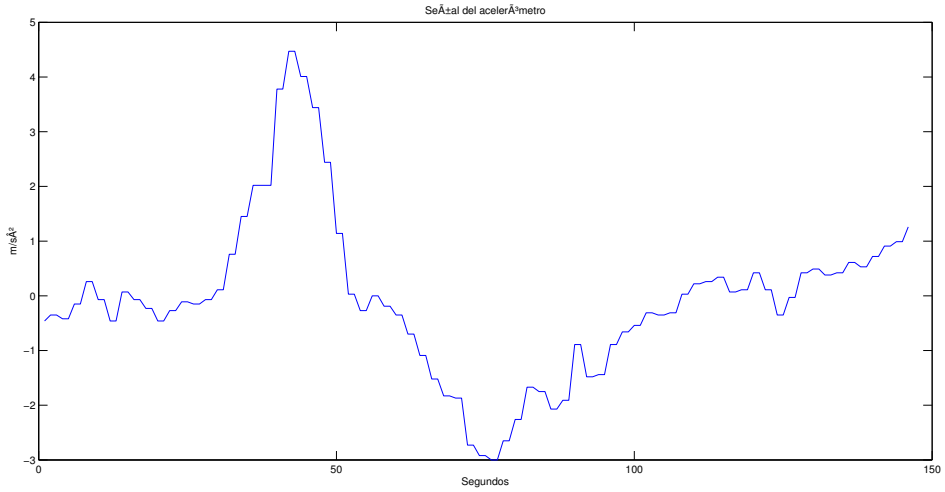

Figura 4.6: Señal del acelerómetro en movimiento natural

4.2.2. Giroscopio

El acelerómetro es un instrumento que permite medir desplazamientos angulares o cambios de dirección. Existen diversos tipos de giroscopios, pero el giroscopio vibrante es el tipo de giroscopio que se usa en los sensores MEMS y es el que se va a estudiar. Como en el acelerómetro se definirá primero el funcionamiento teórico del giroscopio y después el comportamiento del giroscopio MEMS específico del smartphone.

Planteamiento teórico

Ya se ha definido que el giroscopio mecánico es una masa simétrica que gira alrededor de su eje de simetría. Cuando existe alguna momento de fuerza que tiende a cambiar la orientación del eje de rotación, éste opone resistencia cambiando de orientación en una dirección perpendicular a la dirección que lo haría un cuerpo normal en las mismas circunstancias. Este fenómeno hace que si el giroscopio es montado sobre cardanes en tres ejes, pueda conservar su dirección sin importar que el cuerpo que soporta el sistema cambie su dirección permitiendo medir la dirección relativa entre el giroscopio y la base. Sin embargo un giroscopio MEMS tiene un principio de funcionamiento distinto, ya que se basa en el efecto Coriolis para medir los cambios de dirección.

El efecto Coriolis consiste en la existencia de una aceleración relativa de un cuerpo en un sistema de referencia en rotación. Esta aceleración es siempre perpendicular al eje de rotación del sistema y a la velocidad del cuerpo. El efecto Coriolis hace que un objeto que se mueve sobre el radio de un disco en rotación tienda a acelerarse con respecto a ese disco según si el movimiento es hacia el eje de giro o alejándose de éste. Debido a que el objeto sufre una aceleración desde el punto de vista del observador en rotación, es como si para éste existiera una fuerza sobre el objeto que lo acelera. A esta fuerza se la llama fuerza de Coriolis, y no es una fuerza real en el sentido de que no hay nada que la produzca. Se trata pues de una fuerza inercial o ficticia, que se introduce para explicar, desde el punto de vista del sistema en rotación, la aceleración del cuerpo, cuyo origen está en realidad, en el hecho de que el sistema de observación está rotando.

La fuerza de Coriolis es una fuerza ficticia que aparece cuando un cuerpo está en movimiento con respecto a un sistema en rotación y se describe su movimiento en ese referencial. La fuerza de Coriolis es diferente de la fuerza centrífuga. La fuerza de Coriolis siempre es perpendicular a la dirección del eje de rotación del sistema y a la dirección del movimiento del cuerpo vista desde el sistema en rotación. La fuerza de Coriolis tiene dos componentes:

- Una componente tangencial: debido a la componente radial del movimiento del Cuerpo.
- Una componente radial: debida a la componente tangencial del movimiento del cuerpo.

La componente del movimiento del cuerpo paralela al eje de rotación no engendra fuerza de Coriolis. El valor de la fuerza de Coriolis Fc es:

$$Fc = -2m(wxv)$$

Donde:

- m, es la masa del cuerpo.
- v, es la velocidad del cuerpo en el sistema en rotación.
- w, es la velocidad angular del sistema en rotación vista desde un sistema inercial.
- x, indica producto vectorial.

En los sensores *MEMS* se disponen de dos placas que son sometidas al efecto Coriolis al ser sometidas al giro del sensor. Estas placas también forman parte de un capacitor que permite medir la distancia que se forma por la deformación de las mismas y al variar la capacitancia por la fuerza es posible medir la velocidad del giro. Como en el acelerómetro también existen variaciones por ruido blanco gaussiano que afectan los cálculos de actitud.

El giroscopio *MEMS* del smartphone

En la Figura 4.7 se muestra la señal de salida del giroscopio del smartphone en posición estática. A comparación del acelerómetro, el giroscopio tiene una cantidad de ruido mucho menor dentro de una amplitud fija, pero a pesar de ser mucho menor sigue causando problemas en la estimación de la dirección.

Sin embargo durante un movimiento lineal controlado, donde solo hay desplazamiento y no cambio de dirección del smartphone el ruido se aminora como se aprecia en la Figura 4.8, contrario a lo que se esperaría. Un fenómeno no esperado que presenta el sensor pues como se menciono anteriormente, el ruido debería continuar presente en todo momento como con el acelerómetro. En ésta tesis no se estudian los sensores a profundidad, si no su aplicación y metodología para eliminar el deslizamiento en las estimaciones, por lo que solo se hace mención de éste fenómeno para un estudio posterior ya que no afecta directamente al funcionamiento del sistema, pues incluso resulta benéfico.

Otro caso es el movimiento natural del smartphone que se aprecia en la Figura 4.9. Se puede observar que existe ruido en la señal del giroscopio, y por las pruebas anteriores realizadas se sabe que no es un ruido del mismo sensor, si no por el temblor natural de la mano del usuario y sobre todo a la incapacidad del cuerpo humano de mantener en una dirección la mano de forma precisa. Estos deslizamientos son perfectamente cuantificables por el giroscopio y presenta una ventaja para la técnica de estimación de actitud.

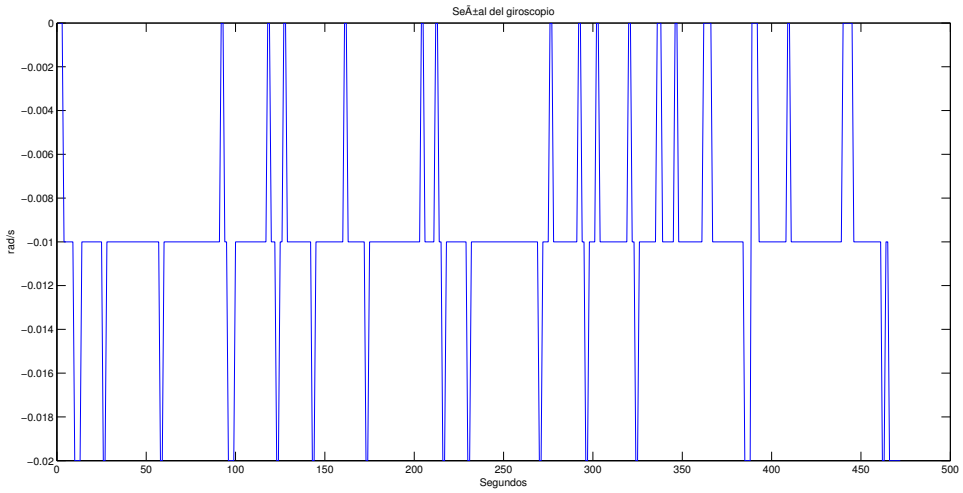

Figura 4.7: Señal del giroscopio en posición estática

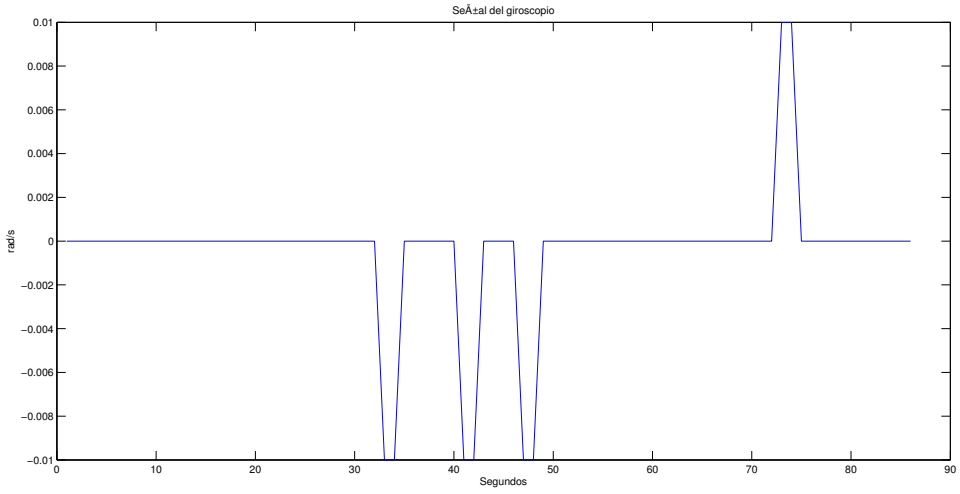

Figura 4.8: Señal del giroscopio en movimiento controlado

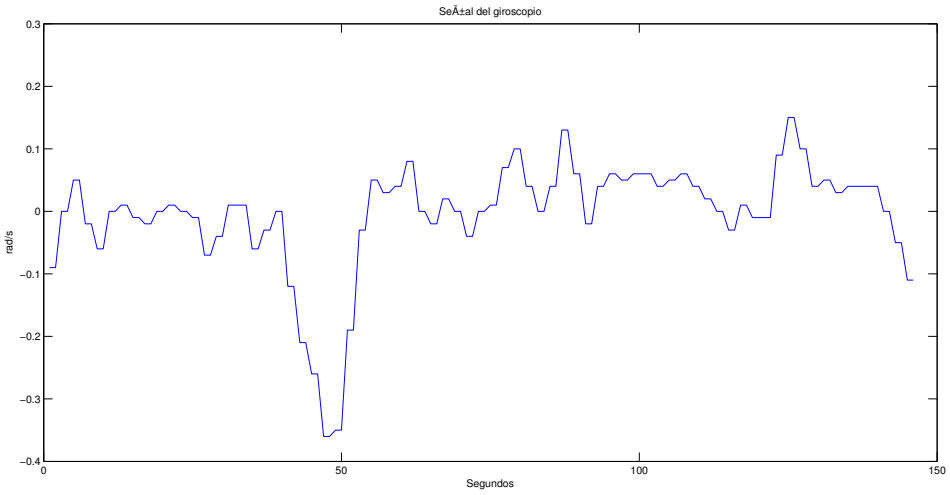

Figura 4.9: Señal del giroscopio en movimiento natural

Los datos adquiridos anteriormente son desplazamientos y el giroscopio mide específicamente cambios de dirección o giros. La Figura 4.10 es un movimiento de giro capturado por el sensor. Se aprecia claramente el momento del movimiento de giro y el término por la velocidad angular registrada. Se observa que los datos igualmente contienen una componente de ruido, y gracias al análisis anterior del giroscopio se deduce que éste movimiento es causado de nuevo por el temblor natural de la mano. Estas perturbaciones o ruido, aunque no pertenezcan al ruido blanco gaussiano provocado por la electrónica del sensor, deben de ser aminoradas para reducir el efecto de deslizamiento en las estimaciones de dirección.

4.2.3. Uso de los sensores de Android

En android todos los sensores se manipulan de forma homogénea. Android permite acceder a los sensores internos del dispositivo a través de las clases *Sensor,SensorEvent, SensorManager*, y la interfaz *SensorEventListener*, del paquete *android.hardware*. La clase Sensor acepta ocho tipos de sensores aunque los sensores disponibles varían en función del dispositivo utilizado. En la tabla4.1 se muestran los sensores soportados por Android y la constante del sistema con la cual la clase Sensor los identifica.

Es necesario registrar cada tipo de sensor por separado para poder obtener información de todos ellos. El método *registerListener* toma como primer parámetro la instancia de la clase que implementa el *SensorEventListener*. El segundo parámetro indica el sensor el cual se desea registrar. El tercer parámetro acepta cuatro posibles valores, que indican al sistema con qué frecuencia en la que se van a recibir actualizaciones del sensor. Esta indicación sirve para que el sistema estime cuánta atención necesitan los sensores, pero no garantiza una frecuencia concreta. Para la aplicación solamente se usan dos sensores que son el acelerómetro y el giroscopio de manera que

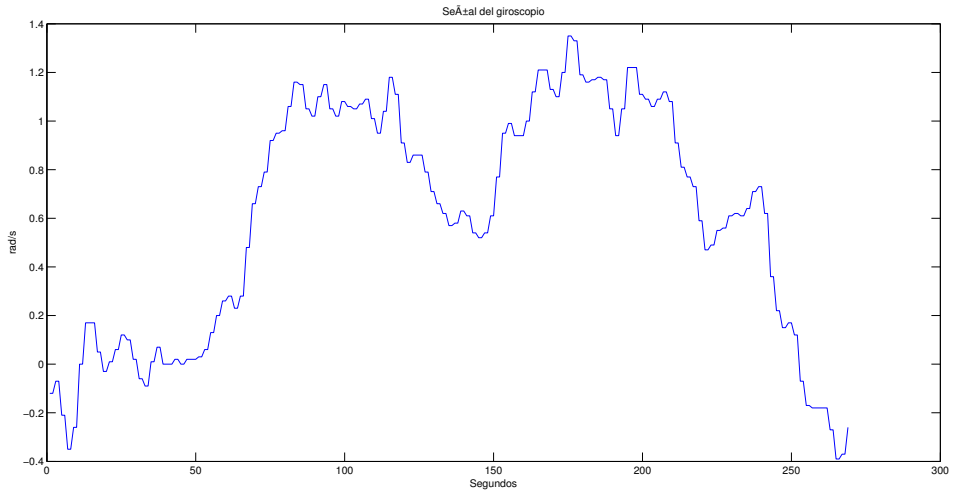

Figura 4.10: Señal del giroscopio sometido a un giro en el eje de medición

el código implementado queda como sigue en la declaración del código 4.1.

Listing 4.1: Declaración de sensores

```
sm.registerListener(this, sm.getDefaultSensor(Sensor.TYPE_ACCELEROMETER), ↩
    SensorManager.SENSOR_DELAY_GAME);
sm.registerListener(this, sm.getDefaultSensor(Sensor.TYPE_GYROSCOPE), SensorManager↩
    .SENSOR_DELAY_GAME);
```

Cuando un sensor tiene un dato listo crea un evento que se dispara en el método *onSensorChanged* y se implementa el código para buscar cual sensor lo ha causado y posteriormente leer los datos. Los valores devueltos por el giroscopio y el acelerómetro son un *array* de 3 valores que corresponden a los tres ejes x,y,z (Codigo 4.2).

Listing 4.2: Método para obtener datos de sensores

```
public void onSensorChanged(SensorEvent event) {
        synchronized (this) {


            switch (event.sensor.getType())
            {
                case Sensor.TYPE_ACCELEROMETER:
                Ax = event.values[0];
                Ay = event.values[1];
                Az = event.values[2];
                Break;

            case Sensor.TYPE_GYROSCOPE:
                Gx = event.values[0];
                Gy = event.values[1];
                Gz = event.values[2];
                Break;

        }
```

Tipo	Constante	Dim
acelerómetro	TYPE_ACCELEROMETER	3
campo magnético	TYPE_MAGNETIC_FIELD	3
giroscopio	TYPE_GYROSCOPE	3
orientación	TYPE_ORIENTATION	3
luz ambiental	TYPE_LIGHT	1
proximidad	TYPE_PROXIMITY	1
presión atmosférica	TYPE_PRESSURE	1
temperatura interna	TYPE_TEMPERATURE	1
gravedad	TYPE_GRAVITY	3
acelerómetro lineal	TYPE_LINEAR_ACCELERATION	3
vector de rotación	TYPE_ROTATION_VECTOR	3
temperatura ambiental	TYPE_AMBIENT_TEMPERATURE	1
humedad relativa	TYPE_RELATIVE_HUMIDITY	1

Tabla 4.1: Tabla de sensores

```
    }
}
```

4.2.4. Tecnica convencional de navegación inercial

El algoritmo para estimar la actitud y rumbo (posición y dirección) se basa en los principios de física de aceleración, velocidad y posición.

Matemáticamente la velocidad $\dot{x}(t)$ y la posición $x(t)$ son calculados integrando la aceleración $\ddot{x}(t)$

$$\dot{x}(t) = \int_0^t \ddot{x}(\tau)d\tau + \dot{x}(0)$$

$$x(t) = \int_0^t \int_0^\tau \ddot{x}(\tau)d\tau dt + \dot{x}(0)t + x(0)$$

Cuando $\dot{x}(0)$ y $x(0)$ son la velocidad y la posición inicial respectivamente.

Para controlar el brazo robot se requiere calcular la posición en relación al origen del mismo pero la posición absoluta del smartphone no es necesaria de manera que lo único que se necesita es calcular los desplazamientos del smartphone para ir calculando la posición a la que se desea desplazar el robot en relación a su origen facilitando el cálculo. Al desplazar el smartphone desde una posición A hasta una posición B y se asume que su velocidad inicial es $\dot{x}(0) = 0$ se realiza el siguiente cálculo

$$x_B(t) - x_A(t) = \int_0^{t_B}\int_0^{\tau} \ddot{x}(\tau)d\tau dt - \int_0^{t_A}\int_0^{\tau} \ddot{x}(\tau)d\tau dt$$

$$= \int_{t_A}^{t_B}\int_0^{\tau} \ddot{x}(\tau)d\tau dt$$

Este desplazamiento es utilizado entonces para calcular la nueva posición que se le va a indicar al robot moverse. En otras palabras se calculan tres incrementos de posición (x,y,z)

$$\Delta x = \int_{t_A}^{t_B}\int_0^{\tau} \ddot{x}_x(\tau)d\tau dt$$

$$\Delta y = \int_{t_A}^{t_B}\int_0^{\tau} \ddot{x}_y(\tau)d\tau dt$$

$$\Delta z = \int_{t_A}^{t_B}\int_0^{\tau} \ddot{x}_z(\tau)d\tau dt$$

Se debe tener en cuenta que el smartphone por ser un sistema digital no es un sistema continuo y los cálculos deben ser de manera discreta y por lo tanto se utilizan integraciones numéricas para obtener una aproximación y para eso se aplica una interpolación numérica

$$\int_{t_0}^{t_n} \ddot{x}(t)dt \approx \sum_{i=1}^{n} \left[\frac{\ddot{x}(i-1) + \ddot{x}(i)}{2}\right] \Delta t$$

El algoritmo completo de estimación de dirección y posición se muestra en la Figura 4.11.

4.2.5. Técnica propuesta de navegación inercial

La técnica de navegación inercial que se propone está diseñado para eliminar el problema de la deriva en el cálculo de la posición. El principio teórico se basa en la reconstrucción de señales con un patrón conociendo solo datos específicos sobre el mismo como es la amplitud y duración del mismo.

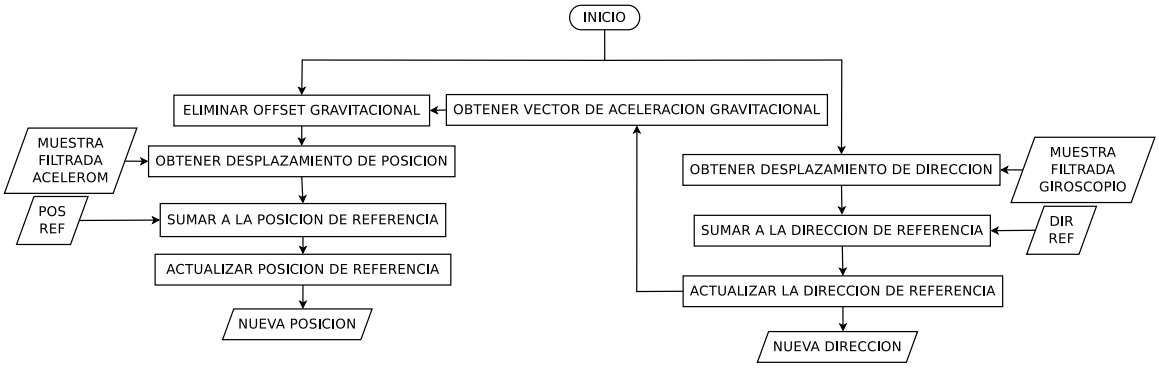

Figura 4.11: Proceso de estimación de posición y dirección

Patrones de movimiento

Los movimientos que se esperan capturar con un smartphone que está siendo manipulado por un usuario, están confinados a una zona específica del espacio. Esto quiere decir que un movimiento hecho por el brazo humano no puede tener una distancia ni duración infinita, ya que está limitado al tamaño del brazo de una persona, al contrario de un vehículo que se desplaza en un espacio teóricamente infinito y hace cambios de dirección y posición que pueden ser igualmente infinitos, como por ejemplo un avión volando en círculos, o un satélite en órbita de un astro. En la práctica no se espera que un usuario realice este tipo de movimientos al manipular un brazo robótico.

Para validar el planteamiento de ésta hipótesis se registraron y graficaron movimientos para estudiar el comportamiento de los movimientos a los que se somete el smartphone al manipular el robot usando la navegación inercial convencional. Los movimientos registrados se pudieron clasificar por sus características de aceleración y en la Figura 4.12 se han graficado 6 de los patrones mas representativos. Sin embargo la información de los acelerómetros resulta más fácil de apreciar cuando el filtro ha eliminado el ruido (Figura 4.13).

Los movimientos se clasificaron en base a las siguientes características:

- Tiempo de aceleración
- Tiempo de frenado (aceleración negativa)
- Máxima aceleración positiva y negativa.

En base a las anteriores características se pueden clasificar éstos 6 diferentes movimientos donde se describirán a continuación.

- Curva de aceleración 1. Es el tipo de movimiento esperado por similitud a una onda senoidal. Tiene un tiempo muy similar de aceleración y frenado con un máxima aceleración en su fase positiva y negativa.

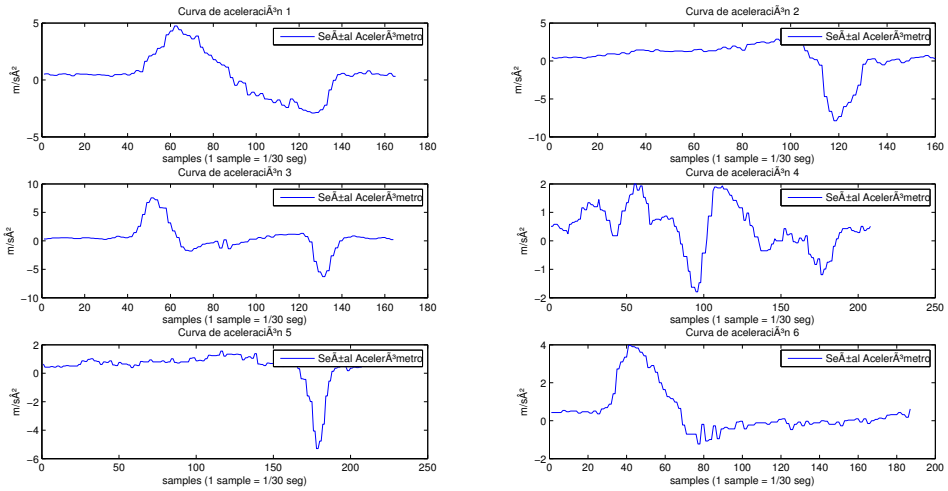

Figura 4.12: Patrones de movimiento captados por el acelerómetro

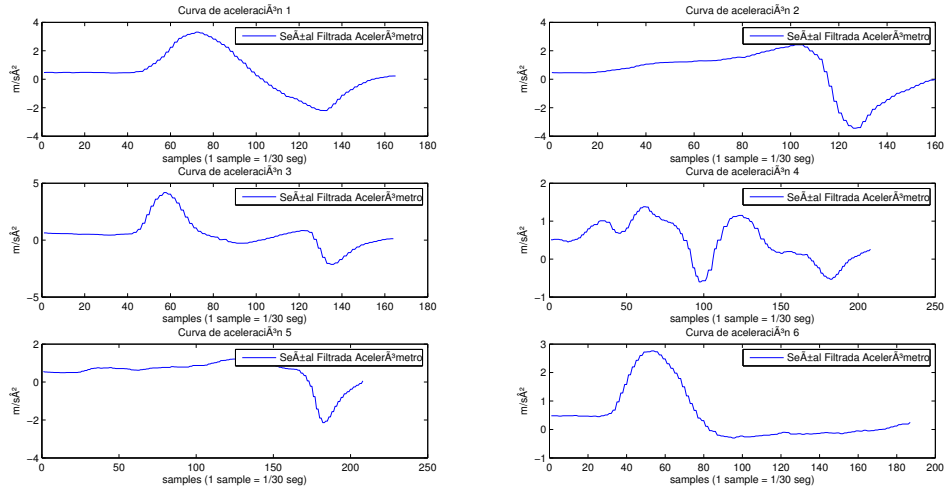

Figura 4.13: Patrones de movimiento después del proceso de filtrado

- Curva de aceleración 2. se tiene una prolongada aceleración pero de menor magnitud que el frenado. El frenado es repentino y de una amplitud mayor que la aceleración.

- Curva de aceleración 3. Se tiene una breve aceleración manteniendo una velocidad constante durante la mayor parte del movimiento hasta el frenado de igual corta duración.

- Curva de aceleración 4. Movimiento irregular. Se tiene una aceleración irregular, con un repentino frenado y una reanudación de la aceleración que termina con un breve frenado.

- Curva de aceleración 5. De características muy similares a la curva 2 con la diferencia de que la aceleración tiene un máximo de 1 m/s^2 mientras el frenado supera los 2 m/s^2. Los máximos en ambas fases son muy asimétricos.

- Curva de aceleración 6. Es un caso invertido a la curva 5. Se tiene una breve y pronunciada aceleración y un suave pero prolongado frenado.

A pesar de la diferencia existente entre los movimientos en las características descritas, se pueden delimitar los movimientos y considerarlos como eventos separados de corta duración.

Planteamiento del modelo de estimación

Un movimiento puede definirse como el cambio de posición de un objeto que tiene una velocidad inicial de cero y al cual se le aplica una aceleración que aumenta gradualmente la velocidad del objeto y posteriormente una desaceleración gradual de igual magnitud donde la velocidad se reduce a cero terminando así el movimiento.

En base a la definición anterior se espera que un movimiento tenga un comportamiento similar a una onda sinusoidal. El algoritmo propone que todos los movimientos pueden ser interpretados y reconstruidos con una senoide conociendo los datos de tiempo del movimiento y amplitud máxima.

Se propone entonces que la aceleración un movimiento puede ser representado por la siguiente expresión:

$$a(t) = M \sin(wt)$$

Donde $a(t)$ es la función de aceleración en el tiempo t M es la amplitud determinada por la aceleración máxima w velocidad angular del movimiento

Esta función garantiza que el desplazamiento tiene una aceleración y un frenado de igual magnitud evitando la deriva ocasionada por el ruido y las irregularidades en el movimiento como se observo en las graficas.

Para asegurar que lo anterior se cumpla sin que el ruido residual del filtro o por movimientos involuntarios afecte la estimación es necesario definir unas reglas para determinar el momento en que se realiza el movimiento. Teóricamente un desplazamiento inicia cuando la aceleración no es cero. Pero debido al ruido observado en la Figura 4.4 en la práctica esa regla no es aplicable, y es necesario discriminar el ruido usando fronteras. Estas fronteras se definieron en +/- 0.5 m/s^2 y cualquier aceleración dentro de ese rango es considerado cero. Así puede definirse que la función de aceleración es una función por segmentos.

$$a(t) = \begin{cases} 0 & si \quad -0.5 < M < 0.5 \\ \\ M\sin(wt) & si \quad -0.5 \geq M \geq 0.5 \end{cases}$$

Asumiendo que esta expresión es la función de aceleración de un movimiento capturado por el acelerómetro puede entonces estimarse el desplazamiento como se ha descrito anteriormente integrando la aceleración para obtener velocidad y posteriormente integrándola a su vez para conocer la posición a la que se ha desplazado el smartphone. Así matemáticamente el algoritmo se puede expresar de la siguiente forma:

$$\ddot{x} = \begin{cases} 0 & si \quad -0.5 < M < 0.5 \\ \\ M\sin(wt) & si \quad -0.5 \geq M \geq 0.5 \end{cases}$$

$$x(t) = \int_{t_0}^{t_f}\int_{t_0}^{t_f} \ddot{x}(t)dtdt$$

La estimación de la dirección sigue usando el algoritmo convencional de integración de la velocidad angular obtenida por el giroscopio donde:

$$\theta = \int_0^t \omega(t)dt + \theta(0)$$

Donde θ es la posición angular estimada $\omega(t)$ es la velocidad angular obtenida del giroscopio

4.3. Filtrado

Ya se ha estudiado el hecho de que las señales de los sensores a parte de la información que propiamente miden según su tipo, agregan ruido blanco gaussiano causado por el mismo sistema electrónico del smartphone y la vibración causada por el ambiente, principalmente por el temblor natural de las manos del usuario. Esta información

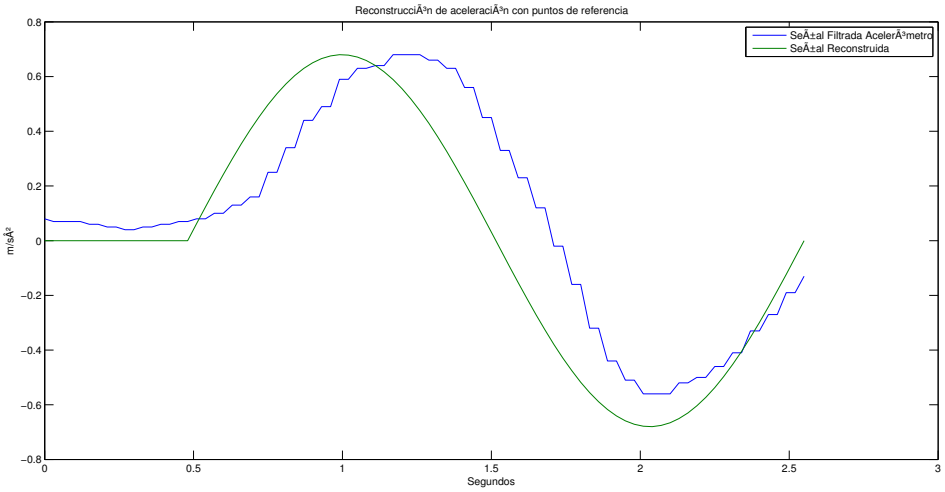

Figura 4.14: Comparación de aceleración real y aceleración sinusoidal en movimiento controlado

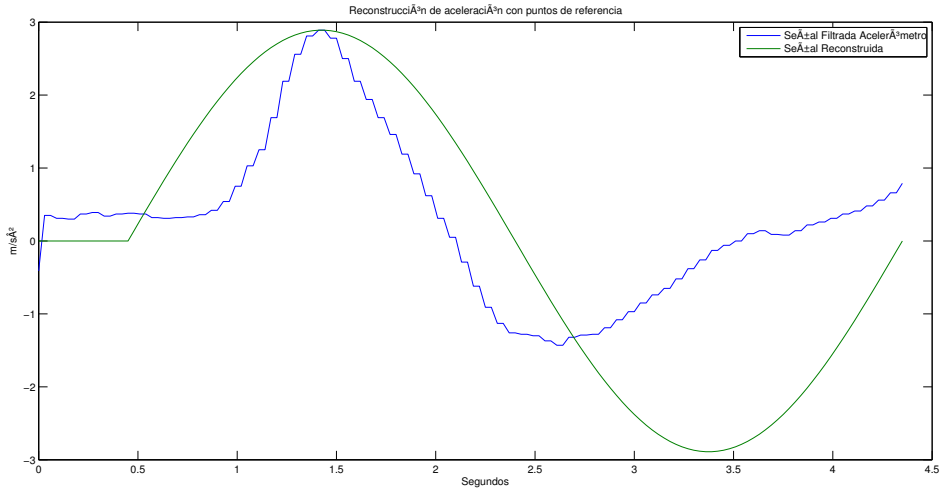

Figura 4.15: Comparación de aceleración real y aceleración sinusoidal en movimiento natural

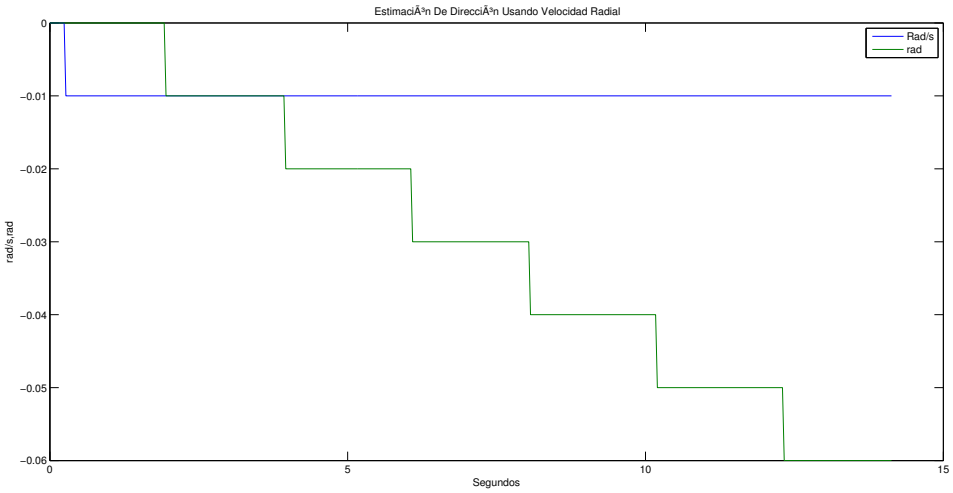

Figura 4.16: Velocidad angular y estimación de posición angular en posición estática

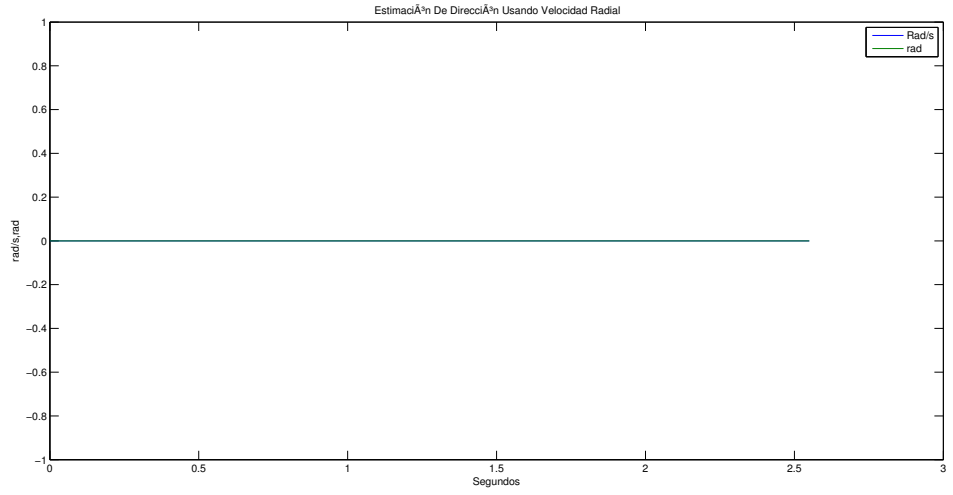

Figura 4.17: Velocidad angular y estimación de posición angular movimiento controlado

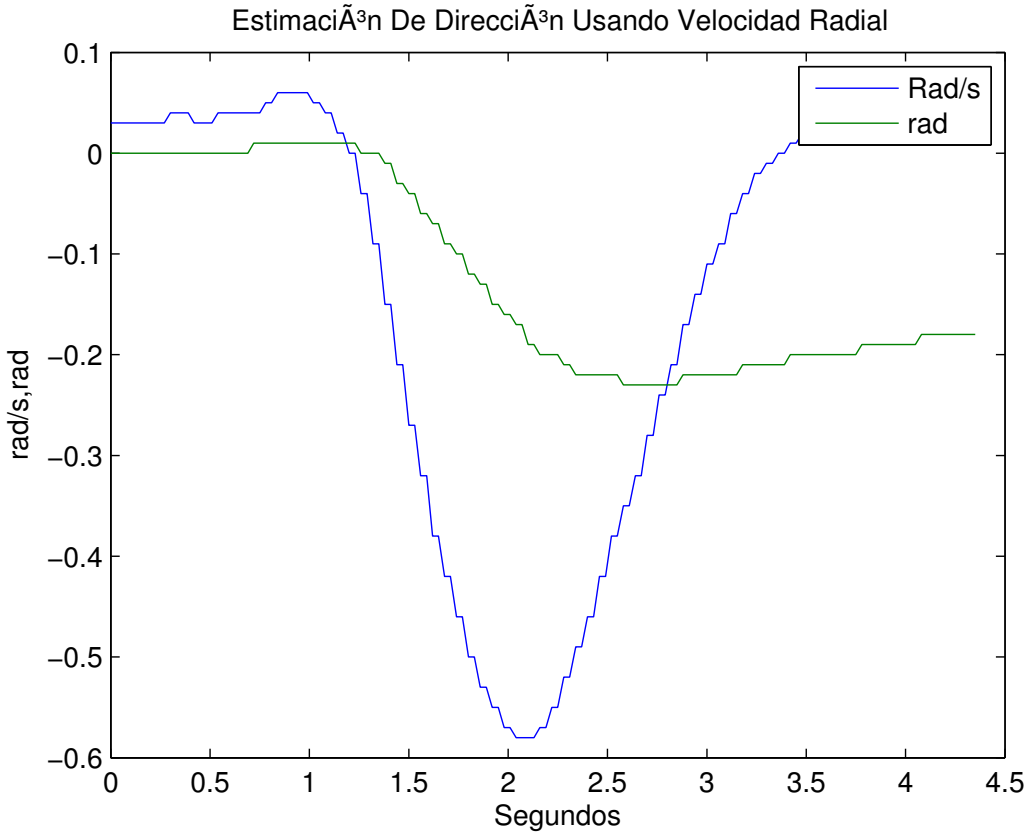

Figura 4.18: Velocidad angular y estimación de posición angular movimiento natural

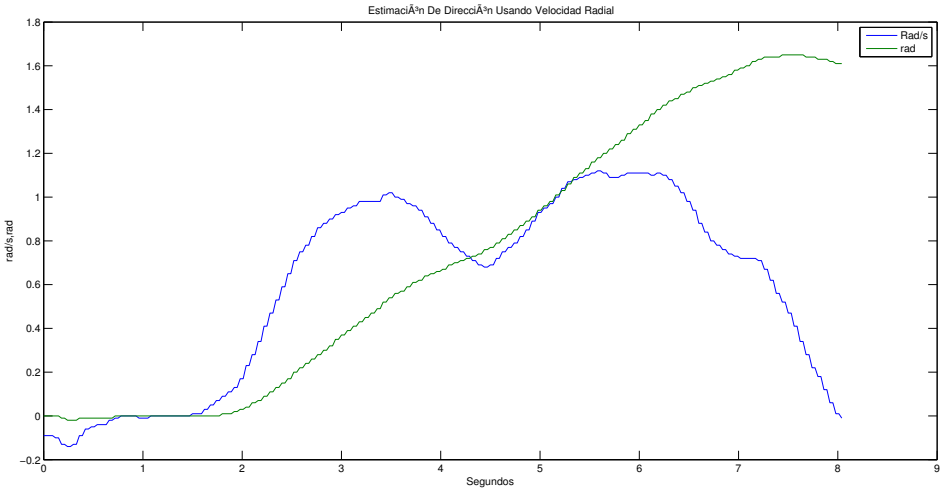

Figura 4.19: Velocidad angular y estimación de posición angular durante el giro en el eje

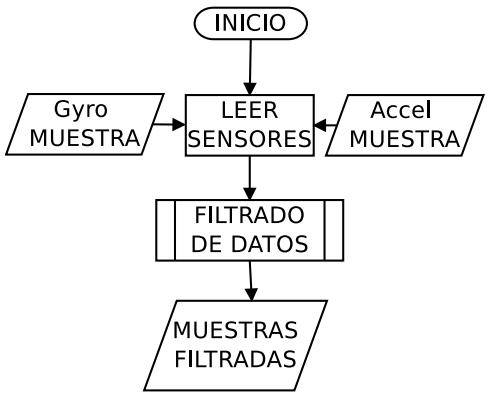

Figura 4.20: Proceso de muestreo y filtrado

adicional causa inestabilidad en los resultados esperados de los cálculos de estimación de actitud (deslizamiento), por lo que es muy importante eliminar este ruido haciendo uso de un filtro de Kalman (Figura 4.20.

Planteamiento teórico

El filtro de Kalman está diseñado específicamente para eliminar el ruido blanco gaussiano que está presente en una señal, y ha sido ampliamente estudiado y aplicado en los sistemas de navegación inercial. El filtro de Kalman presenta la ventaja de que el mismo algoritmo escoge una retroalimentación óptima durante su funcionamiento cuando se conocen las varianzas de los ruidos que afectan al sistema, mientras que un filtro convencional como puede ser un filtro pasa bajos, al ser de lazo abierto y de diseño totalmente diferente, solamente permiten el paso de frecuencias limitando la respuesta y la sensibilidad del sistema.

El filtro Kalman se plantea de la siguiente forma:

$$x_k = A_{k-1}x_{k-1} + B_{k-1}u_{k-1} + w_{k-1}$$
$$z_k = C_{k-1}x_{k-1} + v_{k-1}$$

donde u_{k-1} y w_{k-1} es ruido blanco de valor promedio igual a cero.

El filtro de Kalman es un algoritmo recursivo que está definido en dos pasos: predicción y corrección.

Durante la predicción se hace una estimación a priori

$$\hat{x}_{k|k-1} = \Phi_k x_{k-1|k-1}$$

Y se obtiene la covarianza del error asociada a la estimación a priori

$$P_{k|k-1} = \Phi_k P_{k-1|k-1}\Phi_k^T + Q_k$$

En la corrección se hace una actualización de la medición

$$\tilde{y}_k = z_k - H_k \hat{x}_{k|k-1}$$

Y se calcula la ganancia de Kalman

$$K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1}$$

Y ahora se hace la estimación a posteriori

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k$$

Y la covarianza del error asociada a la estimación a posteriori.

$$P_{k|k} = (I - K_k H_k) P_{k|k-1}$$

Efecto del filtro en la señal de los sensores

La aplicación del filtro de Kalman a las señales de los sensores eliminan el ruido blanco gaussiano de una forma muy satisfactoria, pero se observa que además del ruido del sensor existen las vibraciones causadas por el usuario en el smartphone. A continuación se mostrarán los resultados del filtro en las señales observadas anteriormente para su análisis.

En la Figura 4.21 se muestra el funcionamiento del filtro en la señal del acelerómetro en posición estática. Es notable la reducción del ruido en la señal del sensor pero aún queda un remanente de ruido que puede afectar a los resultados de los cálculos de estimación.

Sin embargo los resultados obtenidos con el filtro en la señal del giroscopio, observados en la Figura 4.22 son por demás satisfactorios, ya que la total del ruido en el sensor es eliminado.

Durante el movimiento controlado en la Figura 4.23 se aprecia mejor el resultado del filtro en la señal, pues el ruido fue eliminado casi en su totalidad. Se aprecia claramente con los datos el momento del inicio del movimiento y el final del mismo.

En los datos arrojados por el giroscopio en el movimiento controlado mostrado en la Figura 4.24 es claramente notable que no existe cambios de dirección del smartphone y el ruido a pesar de ser muy tenue es eliminado por completo.

En la práctica el tipo de datos que se obtienen del acelerómetro del smartphone es con el movimiento natural del usuario, donde como ya se ha descrito, existen también movimientos adicionales causados por un temblor involuntario de la mano. Es importante saber si el filtro es capaz de reducir este tipo de perturbaciones de la información de aceleración del movimiento. En la Figura 4.25 está el resultado de éste movimiento natural donde se observa una curva limpia sin una importante perturbación durante toda la señal.

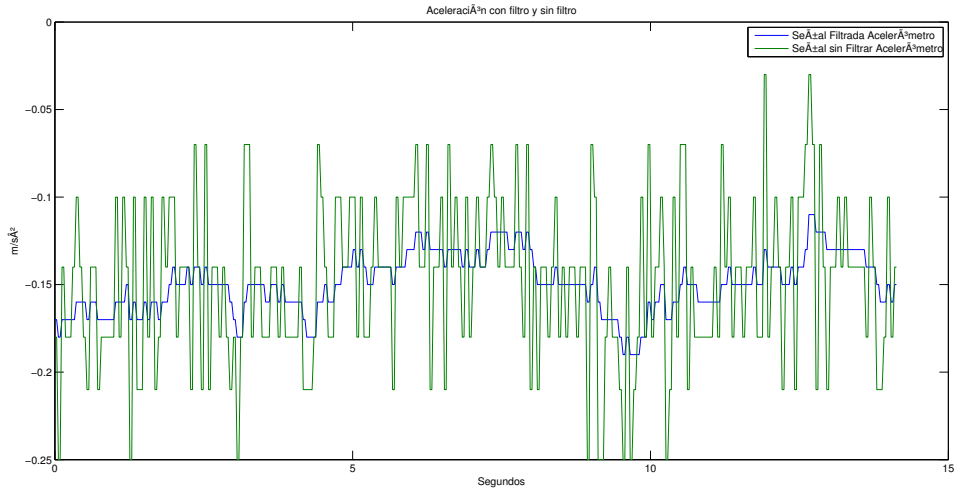

Figura 4.21: Señal del acelerómetro con filtro en posición estática

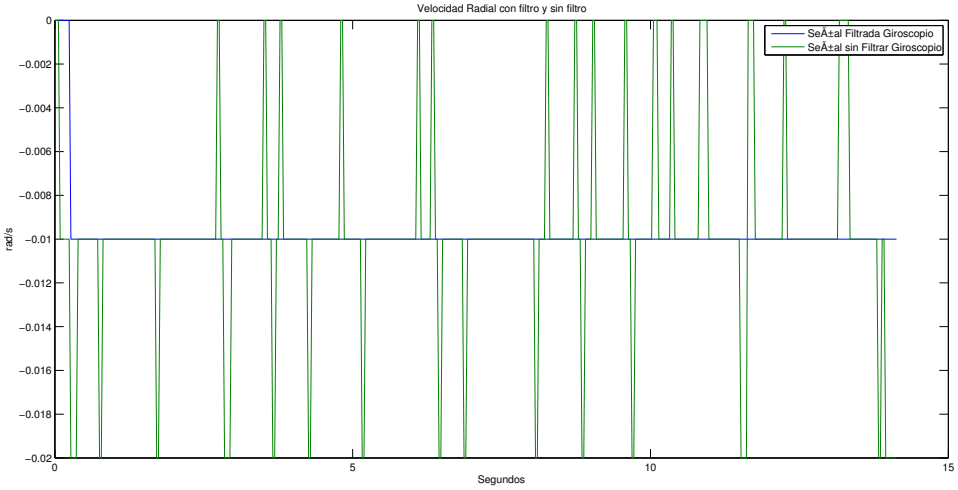

Figura 4.22: Señal del giroscopio con filtro en posición estática

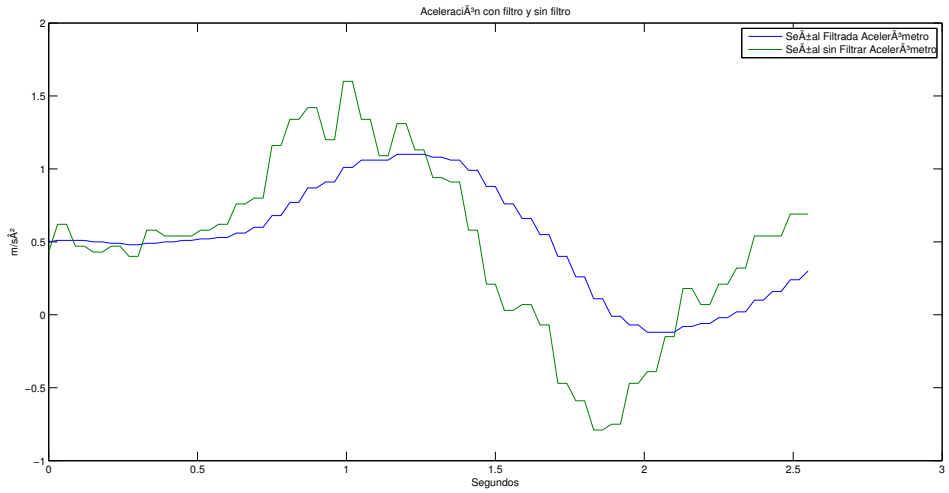

Figura 4.23: Señal del acelerómetro con filtro durante movimiento controlado

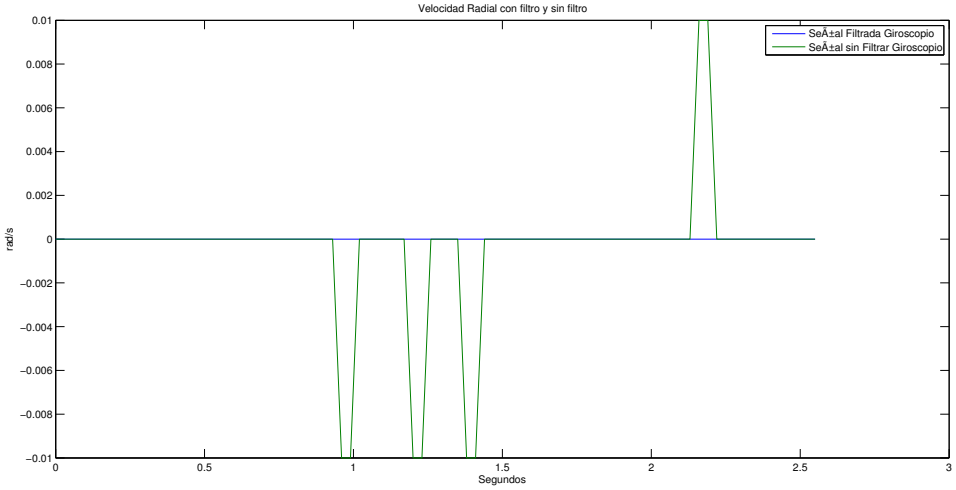

Figura 4.24: Señal del giroscopio con filtro durante movimiento controlado

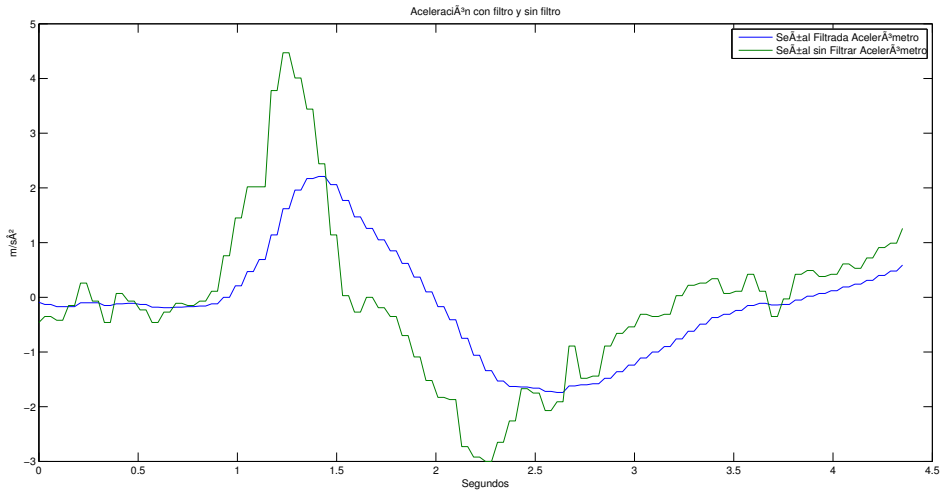

Figura 4.25: Señal del acelerómetro con filtro movimiento natural

Sin embargo en el giroscopio existen variaciones que en teoría no deberían estar (Figura 4.26) pues el movimiento es lineal en una dirección sin cambios de dirección. Esto es causado por el efecto humano descrito anteriormente debido a que en observaciones en posición estática y de movimiento controlado queda totalmente eliminado el ruido gaussiano. Se concluye que ésta información es debida a un cambio en la orientación del smartphone durante el desplazamiento y debe ser estimada, principalmente para poder eliminar el offset en los acelerómetros debida a la aceleración gravitacional.

En el giro del smartphone en uno de sus ejes, la señal del giroscopio muestra una velocidad angular con variaciones donde claramente son eliminadas por el filtro (Figura 4.27). Esto es una ventaja significativa pues el filtro es capaz de eliminar casi por completo las variaciones de velocidad involuntarias del usuario.

4.4. Offset por aceleración gravitacional

Como se explicó en el capítulo de navegación inercial, para estimar la actitud de un objeto se usan los datos proporcionados por el acelerómetro y giroscopio, previamente filtrados, para estimar el desplazamiento y la rotación en el espacio mediante procesos matemáticos. Adicionalmente es necesario resolver el problema del offset causado por la aceleración gravitacional haciendo uso de la información de ambos sensores (acelerómetro y giroscopio) para poder determinar matemáticamente el offset y poder eliminarlo de la medición real de los datos del acelerómetros.

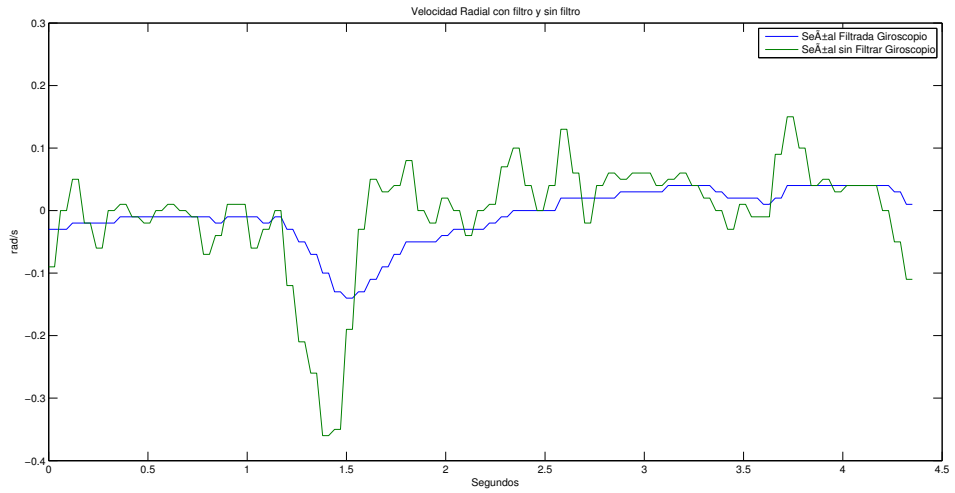

Figura 4.26: Señal del giroscopio con filtro durante movimiento natural

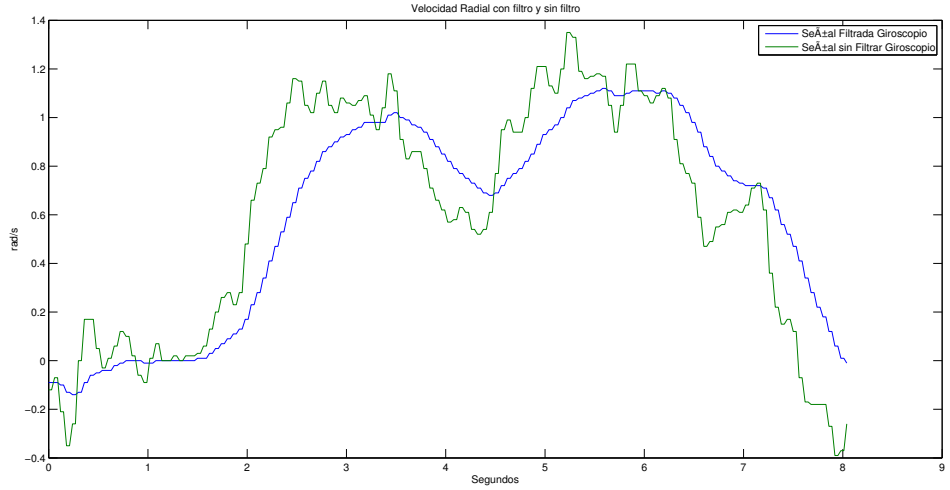

Figura 4.27: Señal del giroscopio con filtro durante el giro en el eje de medición

Problema de la aceleración gravitacional

Ya se ha mencionado que los acelerómetros MEMS son capaces de medir la aceleración gravitacional y en algunas aplicaciones puede ésta magnitud determinar la dirección del smartphone con respecto al centro de la tierra. Pero en el caso de la navegación inercial ésta magnitud resulta indeseada pues se requiere medir aceleración lineal sin que se refleje el *offset* causado por la aceleración gravitacional.

Planteamiento de la aceleración gravitacional

La aceleración gravitacional, como el nombre lo indica, es causada por la fuerza de atracción que ejerce el planeta sobre los cuerpos, llamada gravedad terrestre, y por lo tanto un sensor acelerómetro lo detecta y mide su magnitud que es aproximadamente igual a $9.81 m/s^2$ y es denominado con la letra g.

Las magnitudes medidas de g por los acelerómetros del smartphone corresponden a los componentes del vector tridimensional del mismo dependiendo de la posición en la que se encuentren tal que:

$$a_x(t) = k_a\ddot{x}_x(t) + w_x(t) + d_x + g\cos\gamma$$
$$a_y(t) = k_a\ddot{x}_y(t) + w_y(t) + d_y + g\cos\beta$$
$$a_z(t) = k_a\ddot{x}_z(t) + w_z(t) + d_z + g\cos\alpha$$

Donde: γ es la dirección del eje x con respecto la dirección del vector gravitacional β es la dirección del eje y con respecto la dirección del vector gravitacional α es la dirección del eje z con respecto la dirección del vector gravitacional

Este *offset* sumado a la aceleración que se desea medir es necesario eliminarlo para poder estimar los desplazamientos. El uso de un filtro pasa altos sería la opción más obvia pero debido a la naturaleza del mismo crea sobrepasos que alteran el resultado. La forma correcta es calcular los componentes del vector de aceleración gravitacional de forma dinámica por medio de matrices de rotación usando la información de la dirección estimada por el giroscopio y después restarlos al vector de aceleración medido por el acelerómetro.

Calculo del *offset* usando matrices de rotación

Para calcular el vector de aceleración gravitacional en cada momento de la estimación de actitud es necesario hacer uso de matrices de rotación usando los ángulos de dirección del smartphone calculados con el giroscopio. Las matrices de rotación funcionan en cada uno de los ángulos de Euler, es decir los giros de *Yaw, Pitch y Roll*, conocido como rotación RPY.

- Giro en Yaw. Es un giro en el eje z del sistema. Se define con la matriz de rotacion siguiente.

$$R_\alpha = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Giro en Pitch. Es un giro en el eje y del sistema.

$$R_\beta = \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix}$$

- Giro en Roll. Es un giro en el eje x del sistema.

$$R_\gamma = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & \sin\gamma \\ 0 & -\sin\gamma & \cos\gamma \end{bmatrix}$$

De tal forma que una matriz de rotación RPY queda constituida de la siguiente forma:

$$R_{\alpha\beta\gamma} = R_\alpha R_\beta R_\gamma = \begin{bmatrix} \cos\beta\cos\alpha & \sin\beta\sin\gamma - \cos\beta\sin\alpha\cos\gamma & \cos\beta\sin\alpha\sin\gamma + \sin\beta\cos\gamma \\ \sin\alpha & \cos\alpha\cos\gamma & -\cos\alpha\sin\gamma \\ -\sin\beta\cos\alpha & \sin\beta\sin\alpha\cos\gamma + \cos\beta\sin\gamma & \cos\beta\cos\gamma - \sin\beta\sin\alpha\sin\gamma \end{bmatrix}$$

De ésta forma se puede calcular el *offset* en todo momento usando los ángulos de dirección calculadas por el giroscopio de la siguiente manera:

$$\vec{G}_1 = \vec{G}\chi R_{\alpha\beta\gamma}$$

$$\begin{bmatrix} G_y \\ G_z \\ G_x \end{bmatrix} = \begin{bmatrix} G_y & G_z & G_x \end{bmatrix} \begin{bmatrix} \cos\beta\cos\alpha & \sin\beta\sin\gamma - \cos\beta\sin\alpha\cos\gamma & \cos\beta\sin\alpha\sin\gamma + \sin\beta\cos\gamma \\ \sin\alpha & \cos\alpha\cos\gamma & -\cos\alpha\sin\gamma \\ -\sin\beta\cos\alpha & \sin\beta\sin\alpha\cos\gamma + \cos\beta\sin\gamma & \cos\beta\cos\gamma - \sin\beta\sin\alpha\sin\gamma \end{bmatrix}$$

Eliminación del *offset*

Cuando se ha calculado la magnitud del *offset* por aceleración gravitacional en los tres ejes, la eliminación del *offset* se realiza por una resta de vectores de la siguiente forma:

$$\vec{A}_1 = \vec{A} - \vec{G}$$

$$\begin{bmatrix} A_y \\ A_z \\ A_x \end{bmatrix} = \begin{bmatrix} A_y \\ A_z \\ A_x \end{bmatrix} - \begin{bmatrix} G_y \\ G_z \\ G_x \end{bmatrix}$$

Donde $\vec{A}_1$ es el vector tridimensional de aceleración sin *offset*, $\vec{A}$ es el vector de aceleración tridimensional captado por el acelerómetro y $\vec{G}$ es el vector calculado de aceleración gravitatoria.

4.5. Cinemática inversa

La cinemática inversa es una técnica que emplea cálculos matemáticos para encontrar los ángulos de las coyunturas o articulaciones de un robot para que el extremo del mismo llegue a una posición y orientación determinada en el espacio. Eso significa que la posición calculada por la técnica de navegación inercial en el espacio (x,y,z) no puede ser transmitida al robot directamente y es necesario calcular los ángulos de los motores en las articulaciones para que el brazo se posicione en el punto tridimensional en el espacio.

La cinemática inversa se expresa como una función que transforma un vector tridimensional de coordenadas en un vector de ángulos:

$$q = K^{-1}\left(\Delta x, \Delta y, \Delta z\right), \quad q \in R^n$$

Donde N son los grados de libertad DOF (degress of freedom). K-1 es la cinemática inversa la cual transforma las coordenadas (x,y,z) a los ángulos de cada articulación ($q_1 \dots q_n$)

La complejidad de la cinemática inversa varía dependiendo del número de grados de libertad que tenga el robot en específico pero para esta implementación se escogió un brazo con 5-DOF (Figura 4.28) que permitirá demostrar el funcionamiento del HMI y realizar las pruebas de precisión de los algoritmos de navegación inercial.

Existen varios métodos para encontrar los valores de q_n siendo los más comunes la matriz de transformación homogénea, que usa métodos iterativos y el método geométrico. Usar un brazo con 5-DOF permite la ventaja de poder reducir el costo de proceso de la aplicación por que es posible implementar el método geométrico, que su ventaja principal contra la transformación homogénea es que el cálculo de los ángulos se obtienen directamente a través de las funciones trigonométricas convencionales ya incluidas en las clases estándar de Java en vez de hacer complicadas operaciones de matrices.

El proceso de obtener los ángulos se basa en el planteamiento geométrico del brazo en el que la Figura 4.28 representa la cinemática del brazo real sobre un espacio tridimensional permitiendo señalar la ubicación de los ejes x,y,z donde se va a referenciar

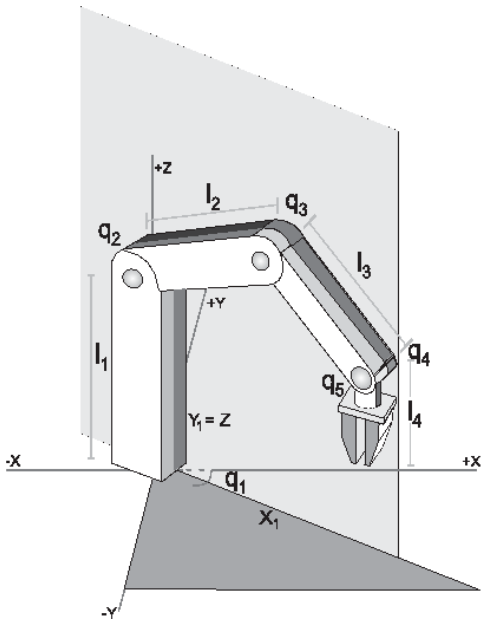

Figura 4.28: Brazo robot con 5-DOF

el sistema de coordenadas calculadas por la navegación inercial. También se muestra la hubicación y nomenclatura de las articulaciones $(q_1 \dots q_n)$ y los eslabones $(l_1 \dots l_n)$.

El primer ángulo puede ser obtenido directamente desde éste planteamiento. El ángulo de la base que está representado por q_1 es el que permite mover el ángulo del resto del brazo en los cuadrantes de x y y. El resto de las articulaciones solo se mueven en un plano ficticio donde se hará una transformación para simplificar el resto del proceso de los cálculos.

Con los valores que corresponden a x y y de las coordenadas se hace uso de la tangente inversa obtener este ángulo de la base q_1

$$q_1 = \arctan\left(\frac{x}{y}\right)$$

Con el valor de q_1 ya no es necesario seguir trabajando en un espacio tridimensional por lo que la cinemática del brazo se replantea en un plano x_1 y y_1 mostrado en la Figura 4.29.

$$x_1 = \sqrt{x^2 + y^2}$$
$$y_1 = z$$

Este nuevo planteamiento permite conocer datos adicionales sobre la geometría del brazo para ir planteando triángulos que permitan conocer los ángulos de las articulaciones.

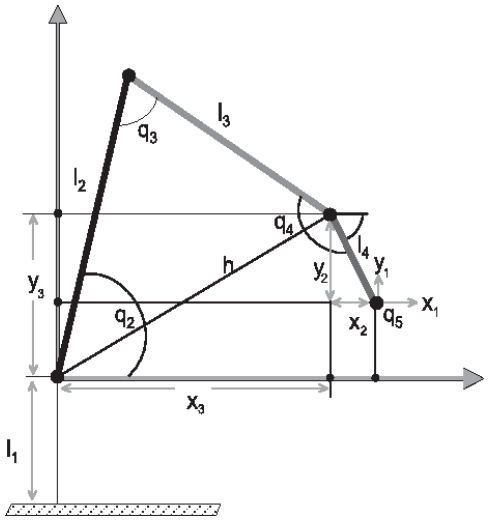

Figura 4.29: Cinemática inversa trigonométrica en los ejes Y Z

$$
\begin{aligned}
x_2 &= l_4 \cos(p_x) \\
x_3 &= x_1 - x_2 \\
y_2 &= l_4 \sin(p_x) \\
y_3 &= y_1 - y_2 - l_1 \\
h &= \sqrt{x_3^2 + y_3^2}
\end{aligned}
$$

Una vez que se han calculado los datos necesarios para los ángulos, es posible calcularos directamente.

$$
\begin{aligned}
q_1 &= \arctan\left(\frac{x}{y}\right) \\
q_2 &= \arctan(\tfrac{y_3}{x_3}) + \cos^{-1}(\tfrac{l_2^2 - l_3^2 + h^2}{2hl_2}) \\
q_3 &= \pi - \cos^{-1}(\tfrac{l_2^2 + l_3^2 - h^2}{2l_2l_3}) \\
q_4 &= p_x - q_2 - q_3 \\
q_5 &= p_z
\end{aligned}
$$

Los ángulos obtenidos tienen magnitudes en $\frac{\pi}{rad}$ que no son las unidades que especifica el software del robot y por lo tanto se realiza una simple conversión con regla de 3 para obtener los valores esperados por el software.

4.6. Comunicación inalámbrica

Los smartphone incluyen una interfaz inalámbrica que opera con el Standard IEEE802.11 para poderse comunicar a las redes WLAN comerciales bajo el protocolo TCP. Al igual que cualquier terminal de red, el smartphone tiene una dirección IP ya sea estática o dinámica solicitada al DHCP de la red. Esto es una gran ventaja pues no es necesario

Figura 4.30: Interfaz gráfica de usuario (GUI)

desarrollar intérpretes ni adquirir conversores de señal, haciendo que la comunicación con otros dispositivos en la red TCP sea transparente. El software de control de Robix incluye una biblioteca en java (rbxUsbor.jar) para poder incluirse en desarrollos. La biblioteca contiene métodos para hacer la conexión TCP al servidor a donde está conectado el hardware y para enviar comandos y configuraciones de manera remota usando como parámetros *strings* con el *script* de robix.

4.7. GUI

La interfase gráfica de usuario (*GUI Graphics User Interfase*) del smartphone está diseñada para ejecutar dos tareas principalmente: Enviar comandos de configuración y monitorear los datos que se ejecutan en la aplicación.

En la Figura 4.30 se observa la pantalla principal de la aplicación donde se encuentran 5 botones y diversos parámetros.

Los botones tienen diferentes funciones que se explican a continuación:

- *Connect*. Cuando la aplicación es ejecutada no se conecta automáticamente para permitir abrir la aplicación aunque no se encuentre el servidor del robot en línea. La aplicación funciona normalmente haciendo estimaciones y realizando todos los cálculos pero sin enviar los datos al servidor.

- *Reset*. Reinicia todos los parámetros de la aplicación sin desconectarse del servidor. También mueve el brazo a una posición inicial.

- *Right Arm/Left Arm*. La aplicación está diseñada para usar dos brazos idénticos ya sea simultáneamente o por separado eligiéndolos con éstos dos botones.

- *Servos On*. Permite apagar los servos sin modificar los parámetros ya calculados. Al encenderse los servos nuevamente el brazo retomará la posición que tenía anteriormente.

Como monitoreo la aplicación despliega una tabla con datos donde se observan en las columnas los datos pertenecientes a cada eje (x,y,z) y en las filas al sensor al que corresponde.

- *Accelerometer*. Son los datos del sensor acelerómetro sin ningún procesado.
- *Gyroscope*. Son los datos del sensor giroscopio sin ningún procesado.
- *Offset*. Es el *offset* de aceleración gravitacional estimado con los datos del giroscopio que será restado a los datos del acelerómetro.
- *Accel Corr*. Es la aceleración menos el *offset* calculado para obtener una aceleración lineal y poder estimar los desplazamientos del smartphone.
- *Delta direction*. Es el desplazamiento de dirección o rotación estimado del smartphone.
- *Delta position*. Es el desplazamiento de posición estimado del smartphone.
- *Direction*. Es la dirección que se envía al brazo.
- *Position*. Es la posición que se envía al smartphone.

Capítulo 5

Resultados y análisis

Para demostrar la validez del sistema desarrollado se realizaron pruebas específicamente para cuantificar y analizar factores de precisión, error y sensibilidad entre los dos sistemas de navegación inercial implementados en la interfaz hombre máquina usando los sensores *MEMS* del smartphone.

Los resultados se dividen en dos tipos de pruebas donde se pueda apreciar el comportamiento de ambas técnicas. En un tipo es posible cuantificar los datos que se obtuvieron en pruebas con dispositivos diseñados para medir los datos. Este tipo de pruebas se realizaron en circunstancias de movimientos controlados para obtener los datos de magnitudes reales de desplazamiento, pero no abarcan todas las circunstancias de un uso real por el usuario. El otro tipo de prueba trata de simular la operación que se puede presentar en la práctica real y se diseñaron pruebas que permiten apreciar de forma cualitativa la precisión del sistema.

La validación de los datos que se obtienen con la técnica propuesta de navegación inercial, se comparan los resultados obtenidos con la técnica convencional.

5.1. Descripción de las pruebas experimentales

Primero se hara una descripción de las características físicas de cada uno de los sistemas con los cuales se realizaron las implementaciones y las pruebas.

En la Figura 5.1 se muestra una fotografía de todos los componentes físicamente y conectados.

- smartphone. El modelo escogido es el SONY XPERIA S (también conocido como Sony Ericsson Nozomi, Sony Ericsson Xperia Arc HD, Sony LT26i y Sony Ericsson Xperia NX en Japón) el cual cuenta en sus características más notables el sistema operativo Android 2.3.7, pantalla táctil de 4.3", cámara trasera de 12 MepaPíxel, procesador Qualcomm dual core, 1GB de memoria RAM, 32 GB de almacenamiento interno, Wi-Fi con estándar IEEE802.11 b/g/n, sensores acelerómetro, giroscopio, sensor de proximidad, GPS y compás digital. Las

Figura 5.1: Sistema completo

características más importantes del dispositivo es que se requieren los sensores giroscopio y acelerómetro y la comunicación Wi-Fi para la comunicación con el servidor. Aunque las características de memoria y procesador superan por mucho el mínimo requerido no son requisitos para la aplicación.

- Router. Es la parte que se encarga de la comunicación inalámbrica y no requiere especificaciones adicionales exceptuando la comunicación Wi-Fi IEEE802.11 para establecer la infraestructura inalámbrica. El *router* escogido es uno de uso comercial y de bajo costo, el Next Nebula 150 el cual tiene 4 puertos de comunicación LAN 10/100 Mbps y comunicación Wi-Fi IEEE802.11n. Se decidió escoger éste dispositivo pues el objetivo es comprobar que no se requieren especificaciones adicionales sobre la comunicación inalámbrica que las que puede ofrecer cualquier *router* del mercado.

- Servidor. El servidor tiene como función la ejecución del software de comunicación con el hardware del robot y sus especificaciones mínimas es el uso de Windows XP y conexión USB. Para las pruebas se utilizó una PC con procesador INTEL CORE i7 con 2GB de memoria RAM, disco duro de 500GB y conexiones USB. El sistema operativo se instaló el Windows 7 32 bits con el programa de desarrollo Eclipse Indigo SR2 for Java Developers de The Eclipse Foundation con el plug-in Android Development Tools para eclipse de Google para el desarrollo de la aplicación del sistema.

- Robot. El brazo con 5 grados de libertad (5-DOF) está construido con partes del kit de desarrollo Robix el cual fue escogido por ser un robot comercial y de bajo costo y su función es el de demostrar el funcionamiento del sistema experimental por lo cual no fue necesario el uso de un robot más especializado.

Las pruebas realizadas tienen como objetivo cuantificar la precisión de las dos técnicas de navegación y se describirá en que consisten:

- Movimiento lineal controlado. El objetivo de esta prueba es obtener datos del movimiento lineal midiendo la distancia real recorrida del smartphone en un solo eje y el movimiento estimado por la aplicación y al comparar los resultados obtener el error medio y la precisión. Para ésta prueba el smartphone es montado sobre un riel con graduación de desplazamiento en centímetros de manera que es posible medir la distancia desde un punto A hasta un punto B. Estos datos son comparados con los datos estimados por la aplicación y la distancia real que el robot se desplaza para conocer el error del cálculo en base a los sensores y el error de la cinemática inversa.

- Movimiento natural. Es una prueba similar al movimiento lineal controlado, pero en este caso no se tiene una distancia de referencia. Es una prueba en condiciones reales de funcionamiento, es decir que el usuario realiza movimientos del smartphone y se capturan los datos para analizar por medio de graficas el comportamiento del sistema y las estimaciones. El objetivo es examinar como trabajan los 6 sensores simultáneamente durante un movimiento en una dirección, en tres direcciones y en un giro.

- Movimiento circular. El objetivo de ésta prueba es para observar el movimiento real del robot contra el movimiento real del smartphone. Los movimientos son realizados en el plano (x,y) de dos formas. Primero se realiza un circulo calculando matemáticamente por medio de senos y cosenos las coordenadas y enviadas directamente a la cinemática inversa para observar el error en el movimiento real del robot. Posteriormente se realiza el circulo con el smartphone desplazándolo sobre un círculo dibujado sobre una superficie para observar la respuesta del sistema completo.

5.2. Resultados y análisis

5.2.1. Movimiento lineal controlado

En ésta prueba se hace desplazar al smartphone en el riel de pruebas una distancia de 65 cm, que es aproximadamente la distancia que puede recorrer en línea recta un brazo humano. Se realizan diez recorridos para obtener la estimación por la técnica de navegación de la distancia y analizar su precisión.

En la tabla 5.1 se muestra cada uno de los datos obtenidos con la técnica de navegación inercial convencional y el error. La distancia media medida con ésta técnica es de 72.7 con un error medio de 7.7. En la tabla 5.2 se muestra cada uno de los datos obtenidos con la técnica de navegación inercial convencional y el error. La distancia media medida con ésta técnica es de 60.3 con un error medio de 4.7.

Measure	Estimated	Error
65	79.5	14.5
65	74.7	9.7
65	75.9	10.9
65	73.5	8.5
65	66	1
65	71.4	6.4
65	77.4	12.4
65	66.6	1.6
65	73.5	8.5
65	69.3	4.3

Tabla 5.1: Tabla de error de estimación de desplazamiento con la técnica de navegación inercial convencional

Measure	Estimated	Error
65	68.7	3.7
65	59.1	5.9
65	69	4
65	56.7	8.3
65	75	10
65	51.3	13.7
65	51.9	13.1
65	52.2	12.8
65	56.7	8.3
65	62.5	12.5

Tabla 5.2: Tabla de error de estimación de desplazamiento con la técnica de navegación inercial propuesta

En la Figura 5.2 está la gráfica de los datos de aceleración aplicado al smartphone y la aceleración calculada por el sistema con el menor error mientras que en la Figura 5.3 los datos de el mayor error. Resulta importante al comparar ambas gráficas que la forma de la curva de la aceleración tiene una diferencia notable.

La comparación de los datos de ambas técnicas dejan como resultado que la navegación inercial propuesta tiene una mejor precisión al acercarse más a los valores reales de desplazamiento de la distancia máxima teórica de un brazo humano. Esto se comprueba haciendo una simulación de la IMU convencional y la IMU modificada en MATLAB haciendo uso de valores reales de los datos obtenidos por los sensores. Los resultados se muestran y se comparan en la tabla 5.3. De esta tabla se compara el funcionamiento de ambos IMU en la Figura 5.5 para el caso con mayor error y en la Figura 5.4 el caso con menor error.

Measure	IMU Conv	IMU Mod
65	44.1	55.0
65	51.1	54.5
65	34.5	55.2
65	39.3	60.9
65	37.6	65.0
65	37.7	68.4
65	34.1	57.9
65	43.1	55.0

Tabla 5.3: Tabla de comparacion de resultados con simulación de IMU en MATLAB usando datos de ensayos reales

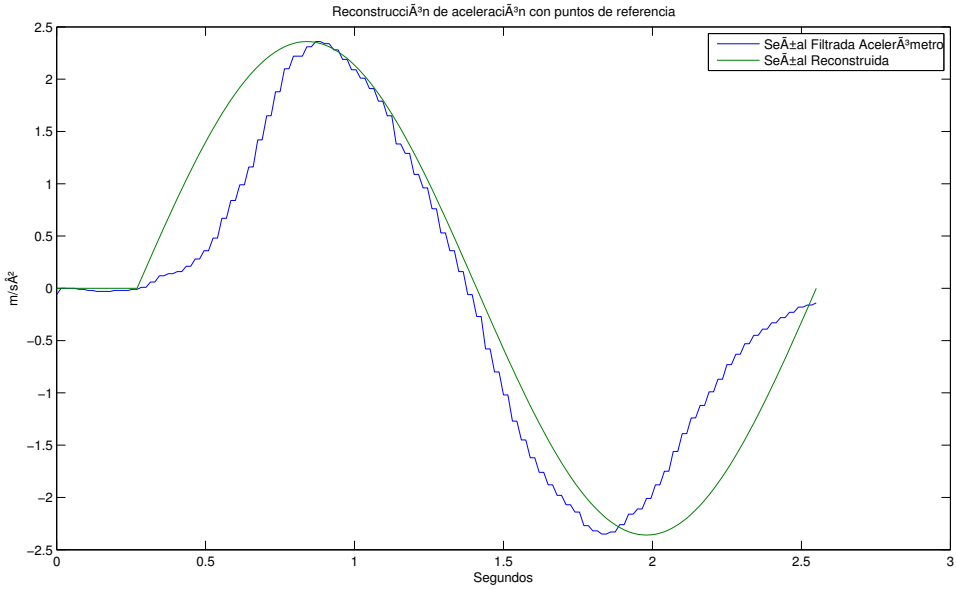

Figura 5.2: Comparación del patrón de aceleración estimada y aceleración real del movimiento normal con error de 0

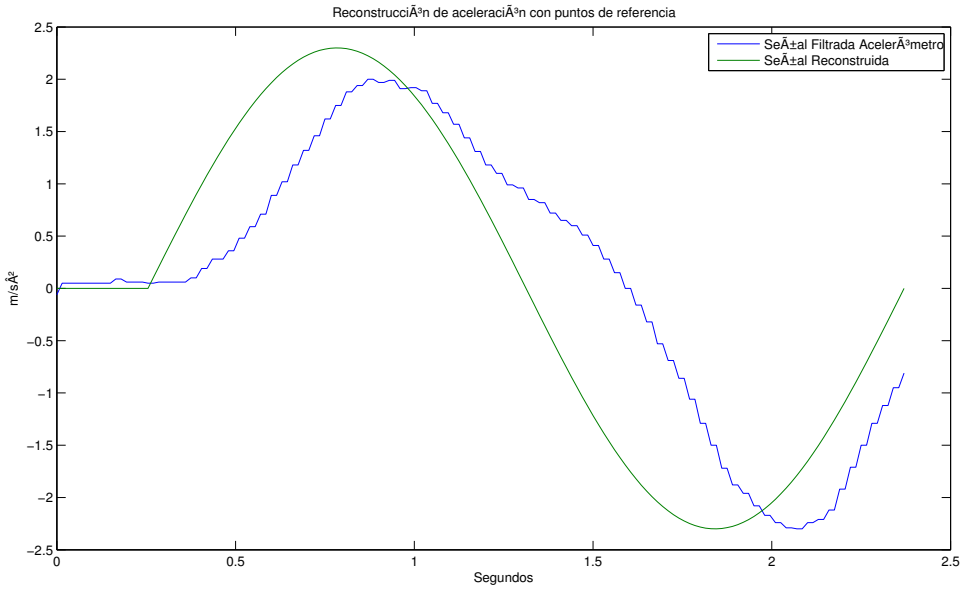

Figura 5.3: Comparación del patrón de aceleración estimada y aceleración real del movimiento normal con error de -10.5

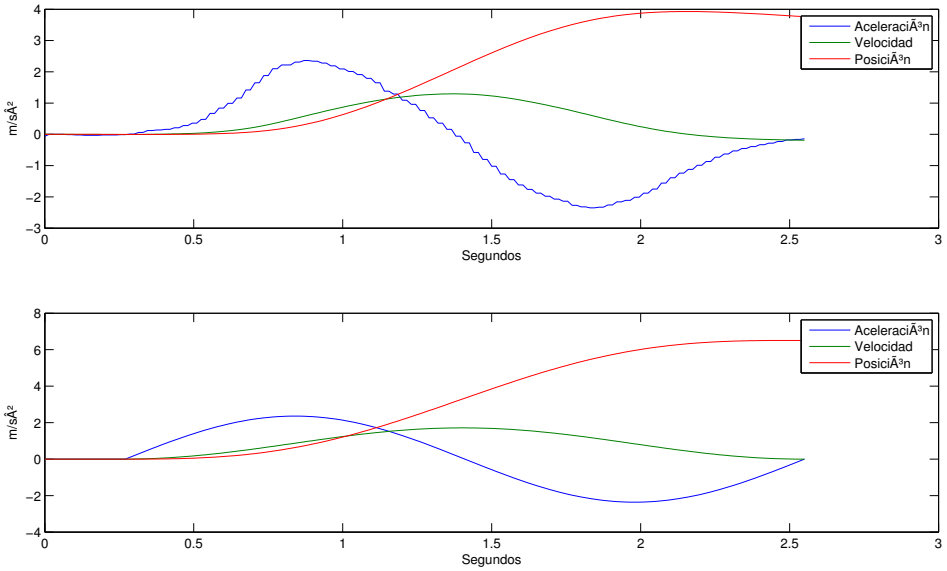

Figura 5.4: Comparativa de funcionamiento de la IMU convencional con la IMU modificada durante el movimiento normal con error de 0

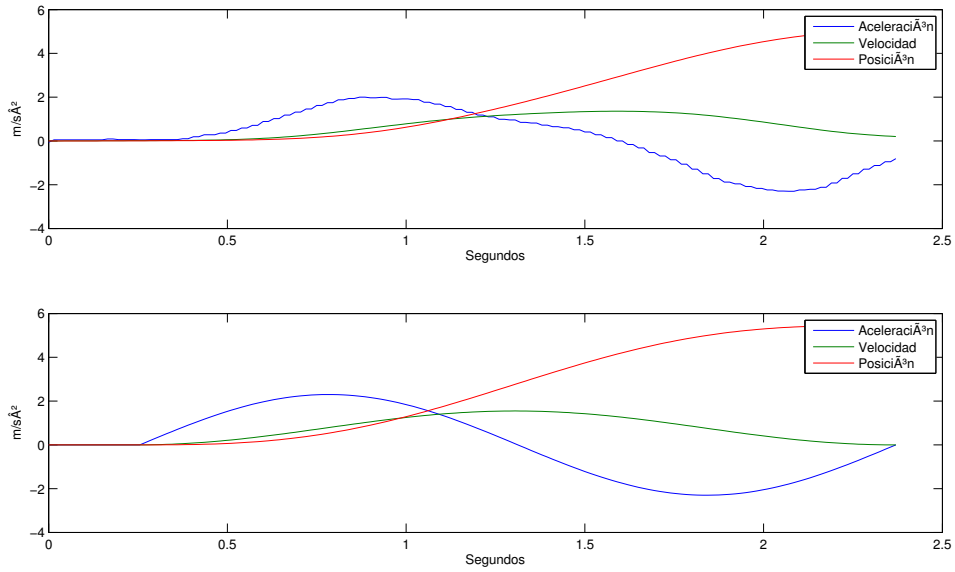

Figura 5.5: Comparativa de funcionamiento de la IMU convencional con la IMU modificada durante el movimiento normal con error de -10.5

En la tabla 5.4 se realizó la misma prueba pero también se consideró el movimiento del robot físicamente. El error mostrado es ahora comparado entre la distancia estimada y el movimiento real del robot.

La diferencia importante entre los dos ensayos es la distancia con la que se está trabajando. La distancia mínima que pueden medir ambas técnicas es la distancia que se tomó como referencia a medir, esto es para conocer la precisión que se tiene en la menor distancia que puede estimarse. Es importante este dato pues es este valor el que define la precisión que se puede trabajar con el smartphone en movimientos delicados.

En la tabla 5.4 se muestra cada uno de los datos obtenidos con la técnica de navegación inercial convencional y el error. La distancia media medida con ésta técnica es de 8.5 con un error medio de 0.8 para una distancia de 9. En la tabla 5.5 se muestra cada uno de los datos obtenidos con la técnica de navegación inercial convencional y el error. La distancia media medida con ésta técnica es de 4.5 con un error medio de 0.6 para una distancia de 5.

En la Figura 5.6 está la gráfica de los datos de aceleración aplicado al smartphone y la aceleración calculada por el sistema con el menor error mientras que en la Figura 5.7 los datos de el mayor error.

Comparando las gráficas de las distancias máxima y mínima de la técnica de navegación inercial propuesta, se observa que las mejores estimaciones se realizan cuando la curva de aceleración inicial está en fase con la curva de aceleración calculada por el sistema, a pesar que la curva de aceleración negativa no esté en fase o tenga una

forma similar a la calculada.

Este hecho permite concluir que la aceleración inicial y el atraso o adelanto de la misma es la que determina el error en el cálculo de la distancia en esta técnica.

Measure	Estimated	Displacement	Error
9	8.3	9	0
9	9.7	10.7	1.7
9	7.5	7.5	-1.5
9	8.7	8.7	-0.3

Tabla 5.4: Tabla de error de estimación y cinemática inversa

Measure	Estimated	Displacement	Error
5	4.5	4.2	-0.8
5	5.4	5.3	0.3
5	4.5	4.2	-0.8
5	3.9	4	-1

Tabla 5.5: Tabla de error de estimación y cinemática inversa

5.2.2. Movimiento Natural

El objetivo de ésta prueba es el de analizar como se comporta el sistema de estimación de desplazamiento y dirección en conjunto. Se omite el funcionamiento de los filtros pues fueron explicados en el capítulo de software. Sin embargo se hará notorio como los 6 sensores trabajan en conjunto para poder estimar un desplazamiento, especialmente si el movimiento está en tres direcciones simultáneamente. Las gráficas

Measure	IMU Conv	IMU Mod
5	6.6	6.1
5	6.6	5.9
5	2.1	4.9
5	4.0	4.5
5	3.6	6.0
5	4.9	3.9
5	4.6	6.1
5	2.8	3.9

Tabla 5.6: Tabla de comparacion de resultados con simulación de IMU en MATLAB usando datos de ensayos reales

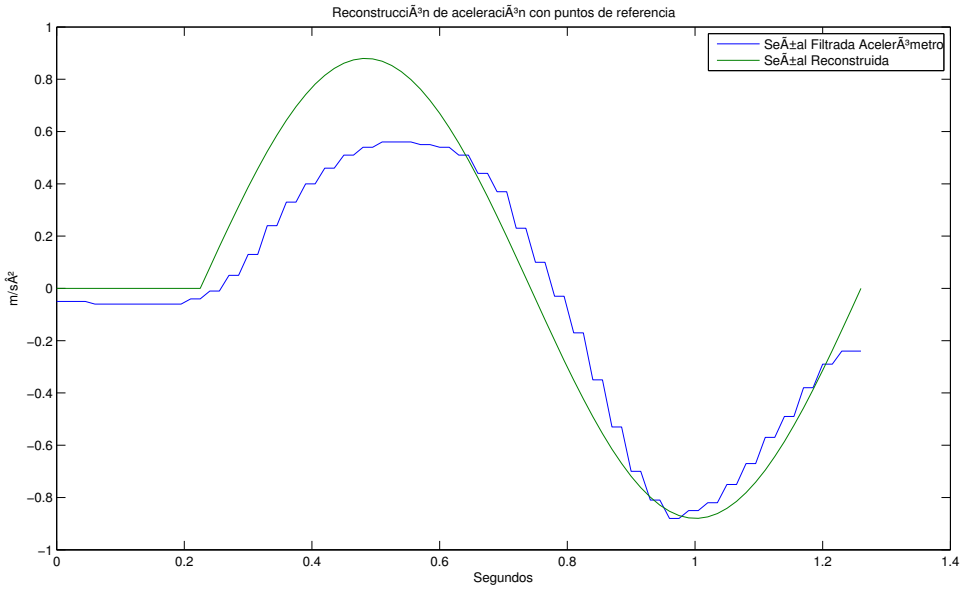

Figura 5.6: Comparación del patrón de aceleración estimada y aceleración real del movimiento mínimo con error de 0.3

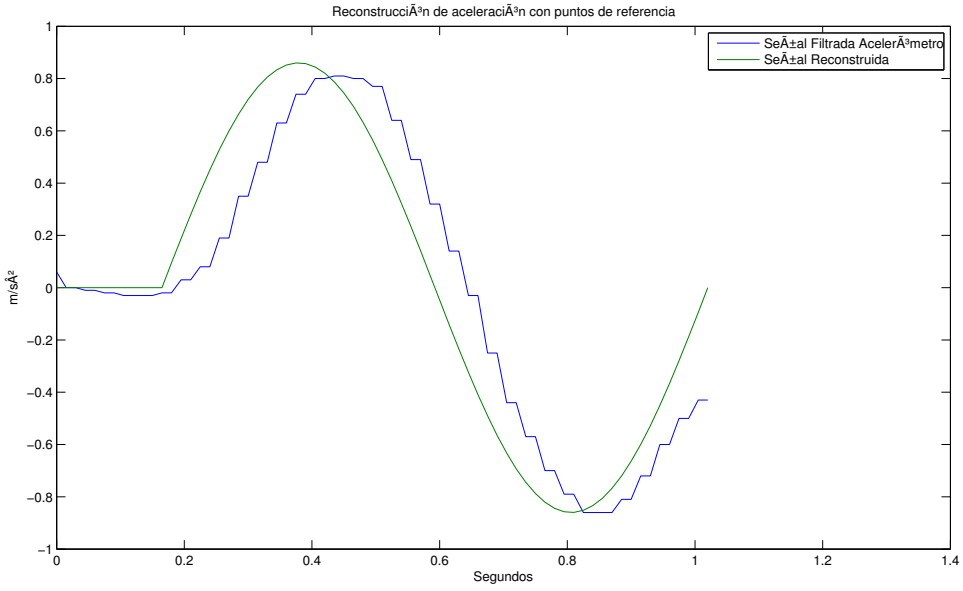

Figura 5.7: Comparación del patrón de aceleración estimada y aceleración real del movimiento mínimo con error de -1

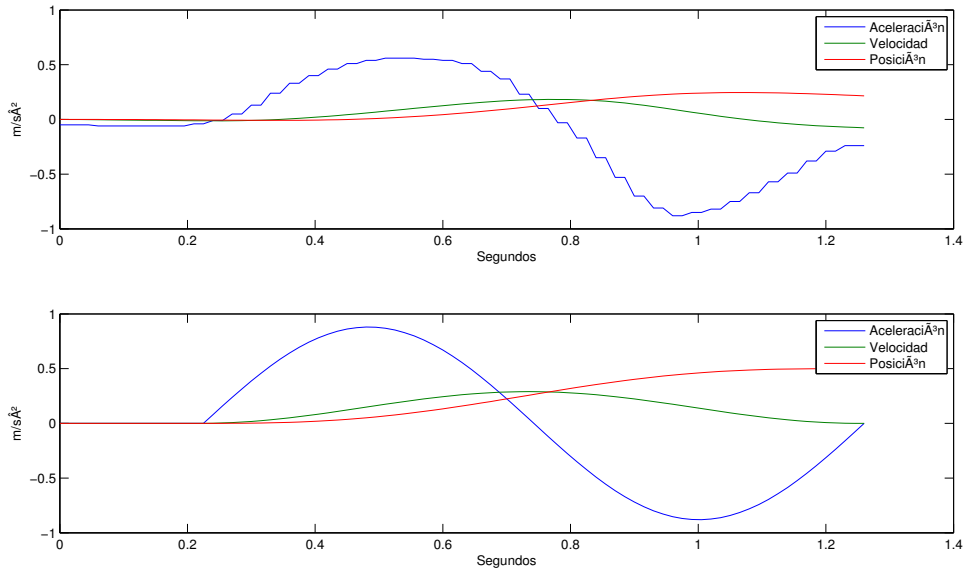

Figura 5.8: Comparativa de funcionamiento de la IMU convencional con la IMU modificada durante el movimiento mínimo con error de 0.3

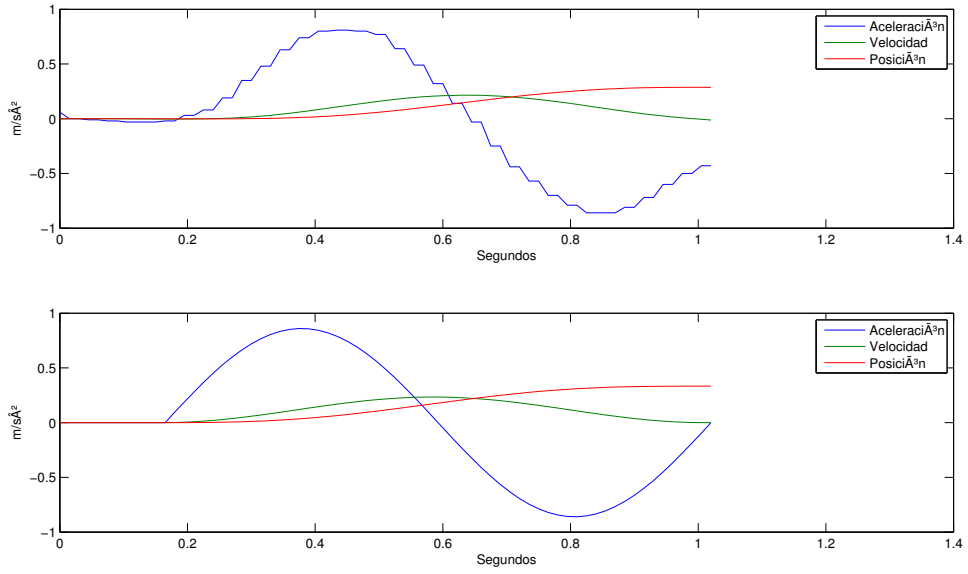

Figura 5.9: Comparativa de funcionamiento de la IMU convencional con la IMU modificada durante el movimiento mínimo con error de -1

mostradas incluye la información de los sensores en los 3 ejes operando simultáneamente, y es como funciona el sistema en condiciones reales. En cada movimiento se muestras 3 juegos de graficas. Primero se analizan las señales del acelerómetro y la corrección del *offset* de forma dinámica. En la segunda gráfica se muestra la señal del acelerómetro comparada con la señal de aceleración estimada por el sistema. En la tercera se muestra la señal del giroscopio y la estimación de la dirección.

Movimiento en una dirección

Este primer movimiento, aunque ya se estudiaron dos casos en el movimiento controlado, se enfocará a mostrar el funcionamiento completo del sistema y no al error.

En las gráficas de la figura 5.10 se muestran 3 señales de aceleración por cada eje. En rojo se tiene la señal de la aceleración corregida y la que se utiliza para estimar los movimientos del smartphone. Se resalta que existe una aceleración negativa pronunciada en el eje X, lo que viene siendo un movimiento hacia la izquierda del smartphone. En el eje Y a penas y existe perturbación en la aceleración y en el eje Z existe una variación muy pequeña, lo que indica que el smartphone no siguió una trayectoria completamente lineal. Es un problema ya explicado en el desarrollo del sistema debido a la imprecisión de los movimientos humanos.

En azul se representa la señal filtrada del acelerómetro y es notable que existe un *offset* causado por la aceleración gravitacional, por lo que la aceleración no es 0 inicialmente como se debería de esperar. En verde se representa la señal calculada por el sistema de la aceleración gravitacional haciendo uso de la dirección calculada con la señal del giroscopio, donde la señal del acelerómetro (azul) se resta a la de la aceleración gravitacional (verde) dando como resultado la aceleración del movimiento solamente (rojo).

El *offset* no es una constante, se debe ir calculando de forma dinámica durante todo el tiempo de la estimación, y se aprecia que existen variaciones de su magnitud, que son causadas por el giro del smartphone por el usuario que causa que el vector de aceleración gravitacional cambie de dirección, afectando a las mediciones del acelerómetro.

En las gráficas mostradas en la figura 5.11 se muestra el funcionamiento del algoritmo que estima el patrón de aceleración en cada uno de los ejes del smartphone en base a los datos de máxima aceleración y duración. El algoritmo tiene una regla que no inicia la estimación si la aceleración no excede a $0.5m/s^2$ de manera que permite discriminar las variaciones de movimiento involuntario del usuario, mostrado en los ejes Y y Z donde no existe ninguna aceleración en el algoritmo de estimación mientras que el eje X si se realizó una estimación. Esta regla en el algoritmo permite agregar precisión en la experiencia del usuario al asumir que solo existe movimiento en una sola dirección.

La figura 5.12 muestra la gráfica de la señal del giroscopio y la dirección calculada en los tres ejes. Esta gráfica complementa la primer gráfica al mostrar como existe un

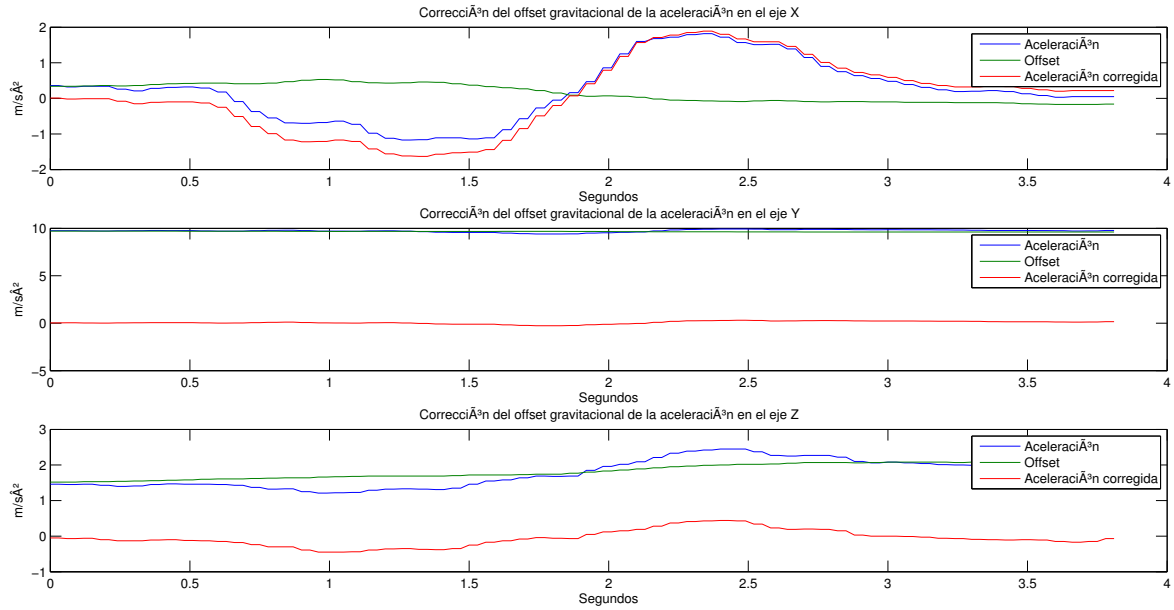

Figura 5.10: Datos de XYZ de Acelerómetro filtrado, *offset* calculado y Aceleración corregida en movimiento en un solo eje

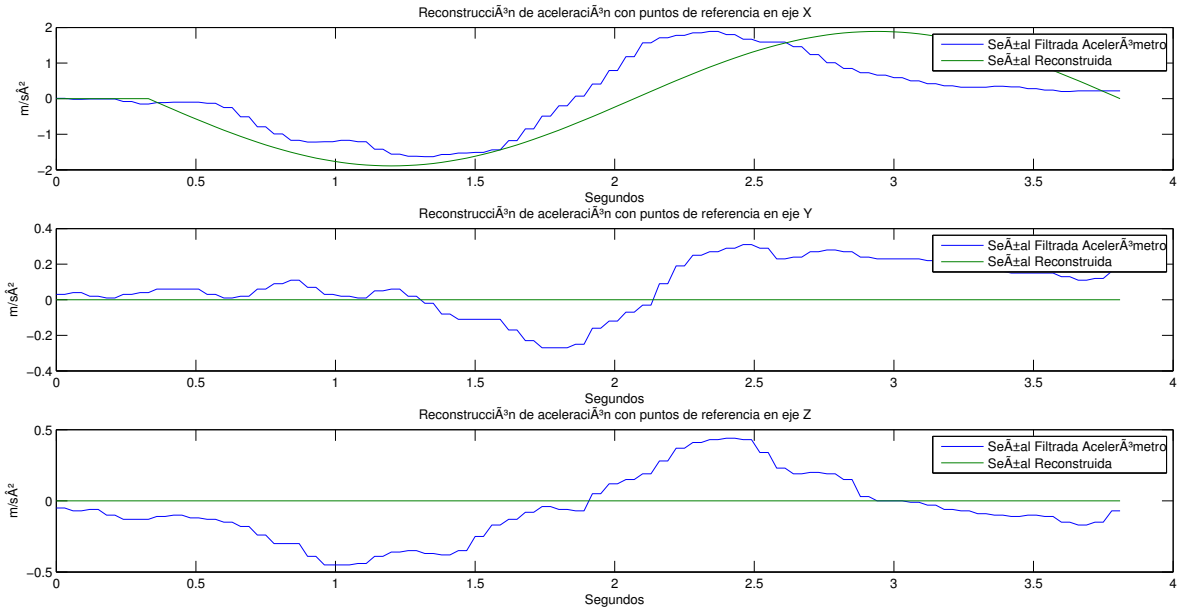

Figura 5.11: Datos de XYZ de acelerómetro filtrado y aceleración estimada en movimiento en un solo eje

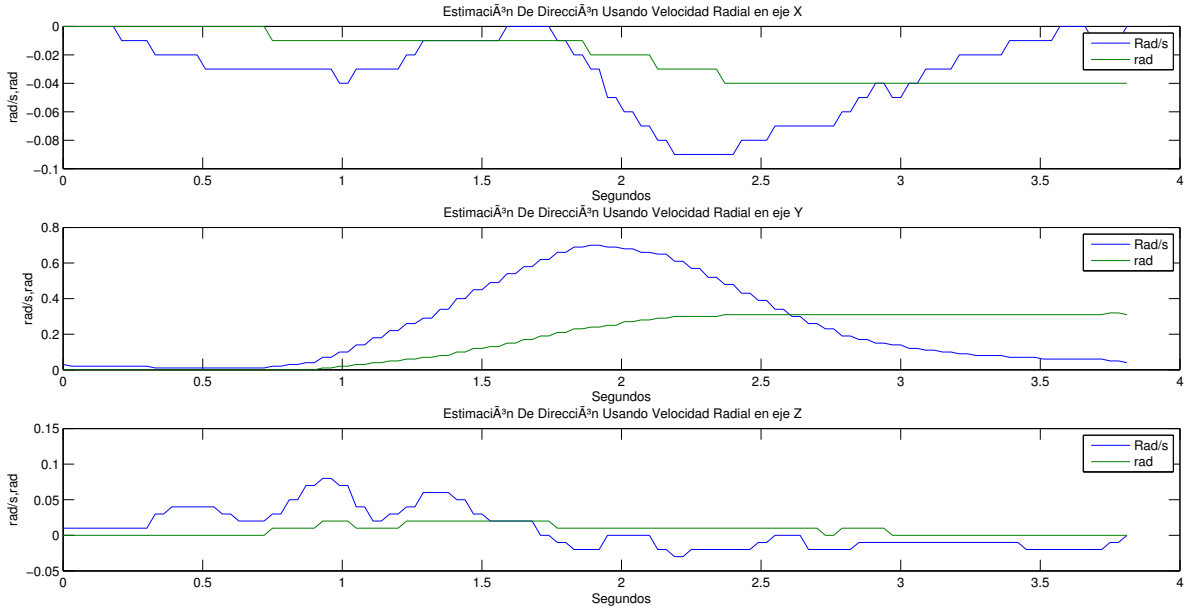

Figura 5.12: Datos de XYZ de giroscopio filtrado y dirección estimada en movimiento en un solo eje

cambio de dirección durante el desplazamiento del smartphone que hace que varíe la magnitud del *offset* en las componentes del vector. Los ángulos no solo sirven para calcular el *offset* si no que también es la orientación de la herramienta de la muñeca del brazo manipulador. Así el sistema completo estima la dirección y desplazamiento simultáneamente para enviar estos datos a la cinemática inversa del robot.

Movimiento en tres direcciones

Este movimiento a diferencia del ejemplo anterior, se realizó desplazando el smartphone en diagonal para obtener una estimación en los tres ejes. Este tipo de movimiento permite estimar los movimientos con mayor naturalidad pues en la práctica no se realizan movimientos en una sola dirección.

Al analizar las gráficas de la figura 5.13 se observa que se registran movimientos en los tres sensores acelerómetro. La relevancia de éstas muestras se observan en la figura 5.14 pues se realizan estimaciones de desplazamiento en los tres ejes. Es posible saber la dirección de los movimientos por la dirección en la que se realiza la aceleración. En el eje X la aceleración positiva indica que se tiene un movimiento hacia la derecha, en el eje Y es un movimiento hacia arriba por la aceleración positiva y por último el eje Z indica por la aceleración negativa que es un movimiento hacia atrás. Todos éstos movimientos son en relación al smartphone como referencia si se observa de frente en posición vertical.

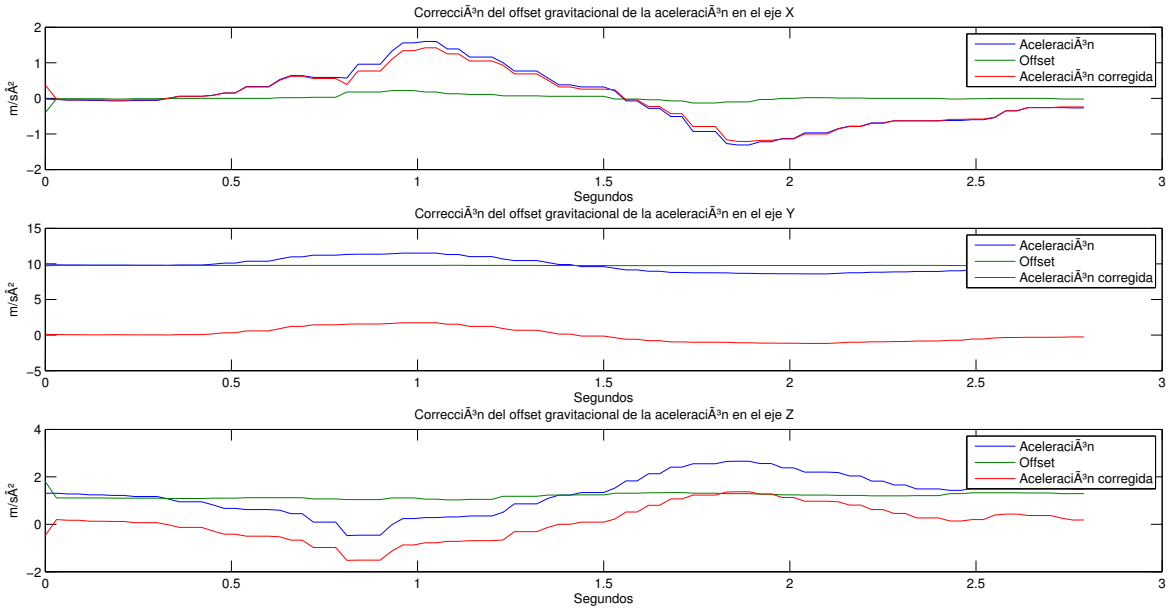

Figura 5.13: Datos de XYZ de Acelerómetro filtrado, *offset* calculado y Aceleración corregida en movimiento en tres ejes

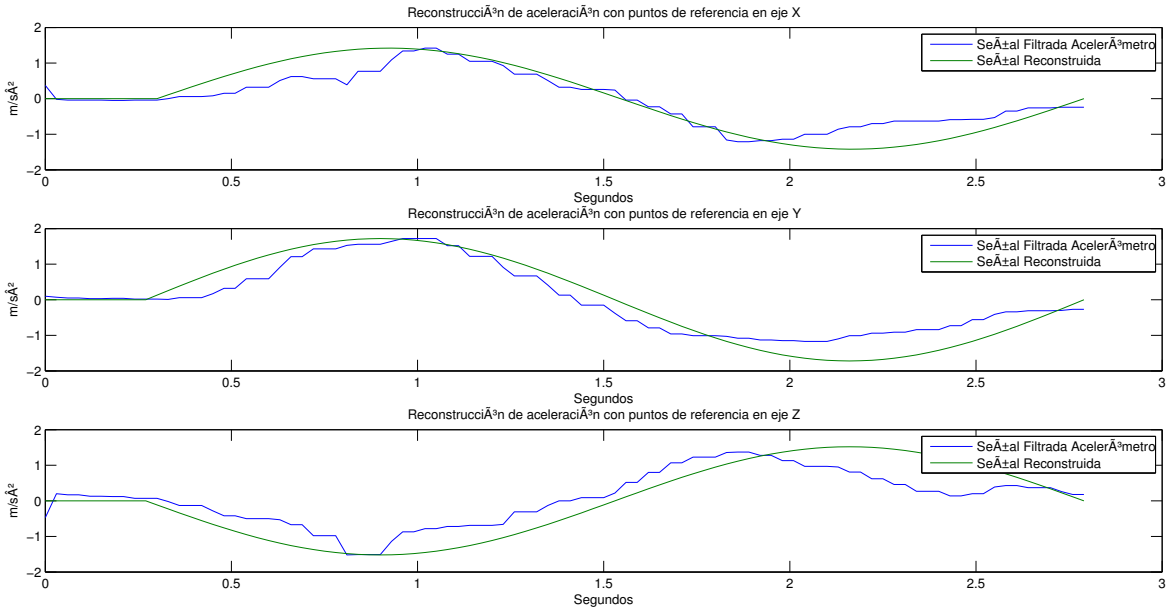

Figura 5.14: Datos de XYZ de acelerómetro filtrado y aceleración estimada en movimiento en tres ejes

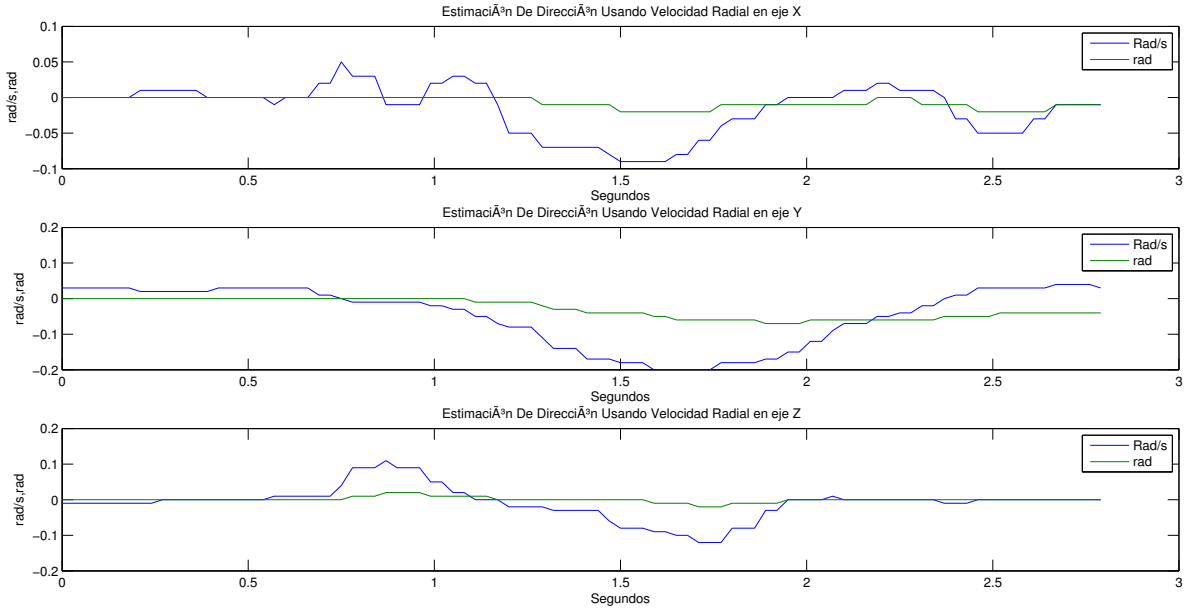

Figura 5.15: Datos de XYZ de giroscopio filtrado y dirección estimada en movimiento en un tres ejes

Por último las gráficas de la figura 5.15 indican que existe un cambio de dirección muy pequeño del smartphone durante el trayecto. El giroscopio se aprecia, que a comparación del acelerómetro, tiene una mayor precisión lo que permite hacer una estimación de la posición angular por el método de sumatoria de la velocidad angular para obtener la dirección. Esta característica del giroscopio permitió que fuera posible calcular el *offset* de forma dinámica, pues de tener el mismo problema de ruido que el acelerómetro, no sería posible aplicar ésta técnica.

Movimiento de giro

Los anteriores datos analizados son movimientos de desplazamiento en línea recta, sobre un eje o en varios ejes, pero en éste caso solo se observará el comportamiento del sistema en un giro del smartphone sin desplazamiento.

La magnitud captada por los acelerómetros tienen un comportamiento diferente a un movimiento lineal. Al cambiar de dirección el acelerómetro, el vector de aceleración gravitacional cambia de dirección también causando que el *offset* se desplace dramáticamente como se observa en la figura 5.16. La estimación del *offset* permite ir cancelando este desplazamiento dinámicamente permitiendo que solo quede la aceleración residual que es por el mismo movimiento del smartphone durante el giro.

En la figura 5.17 se observan con mas detalle las aceleraciones residuales. En condiciones ideales, un giro no deberían presentarse éstas aceleraciones pues no existe un

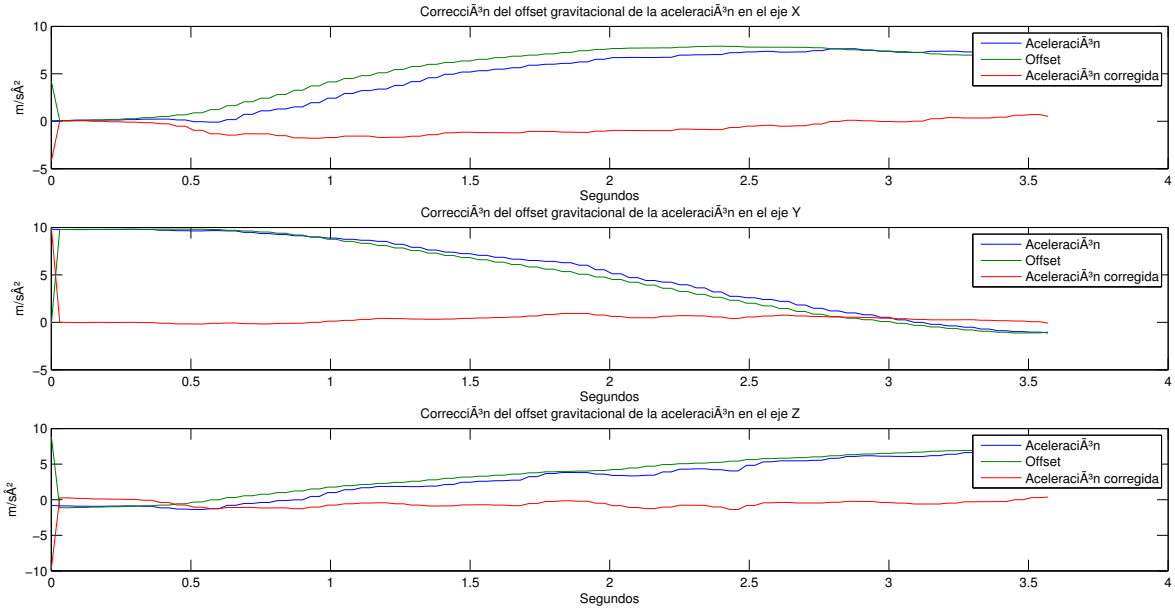

Figura 5.16: Datos de XYZ de Acelerómetro filtrado, *offset* calculado y Aceleración corregida en giro de tres ejes

desplazamiento, pero el giro realizado por el usuario no es perfecto, de manera que existen desplazamientos junto con el giro.

Las gráficas del giroscopio en la figura 5.18 muestran una mayor actividad que en los anteriores experimentos. Si se complementa la gráfica del giroscopio con la del acelerómetro se puede apreciar claramente como el cambio de dirección afecta de forma significativa al *offset* calculado.

5.2.3. Movimiento circular

La prueba del movimiento circular permite conocer la precisión del sistema cuando se hacen movimientos en dos ejes y poder evaluar la precisión de estimación de la técnica.

Para tener una referencia sobre el circulo que se debe dibujar, se hizo un circulo calculado desde software donde las coordenadas son el resultado de senos y cosenos para determinar la trayectoria del brazo. El resultado observado de el comportamiento del brazo manipulador se aprecia en la Figura 5.19. El circulo presenta una notable deformación causada por la imprecisión mecánica del brazo y los servos. Se pretende que el movimiento capturado por el usuario realizando un cículo debe de asemejarse al patron observado por el círculo calculado matemáticamente.

En la Figura 5.20 están tres ensayos realizados con la estimación de la técnica de navegación inercial convencional. El patrón observado es notablemente diferente

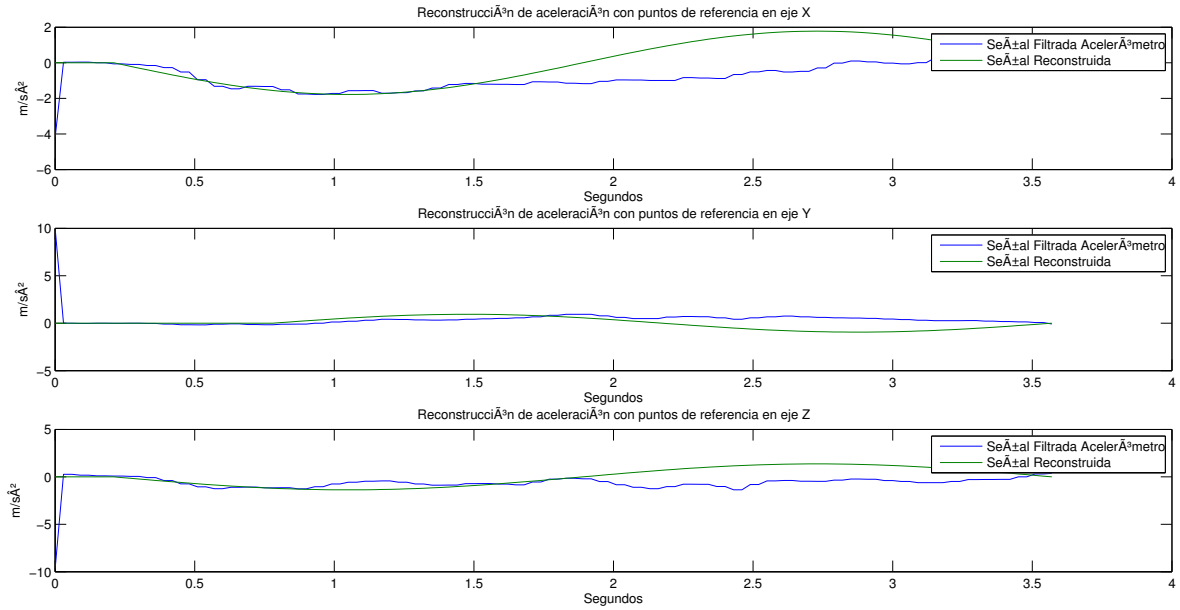

Figura 5.17: Datos de XYZ de acelerómetro filtrado y aceleración estimada en movimiento en giro de tres ejes

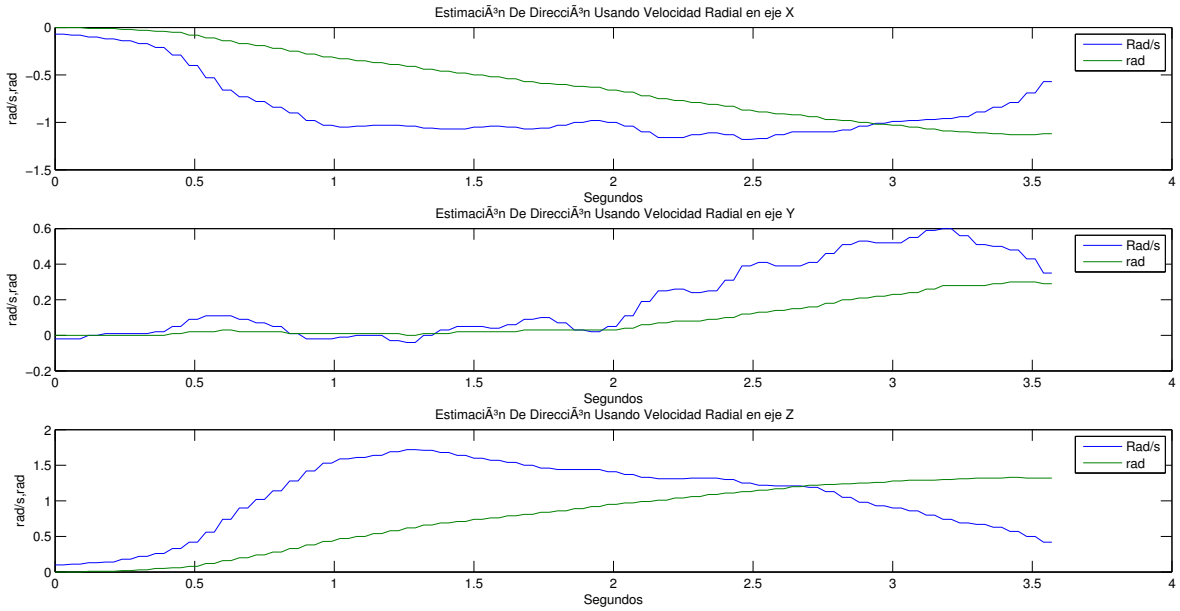

Figura 5.18: Datos de XYZ de giroscopio filtrado y dirección estimada en movimiento en giro de tres ejes

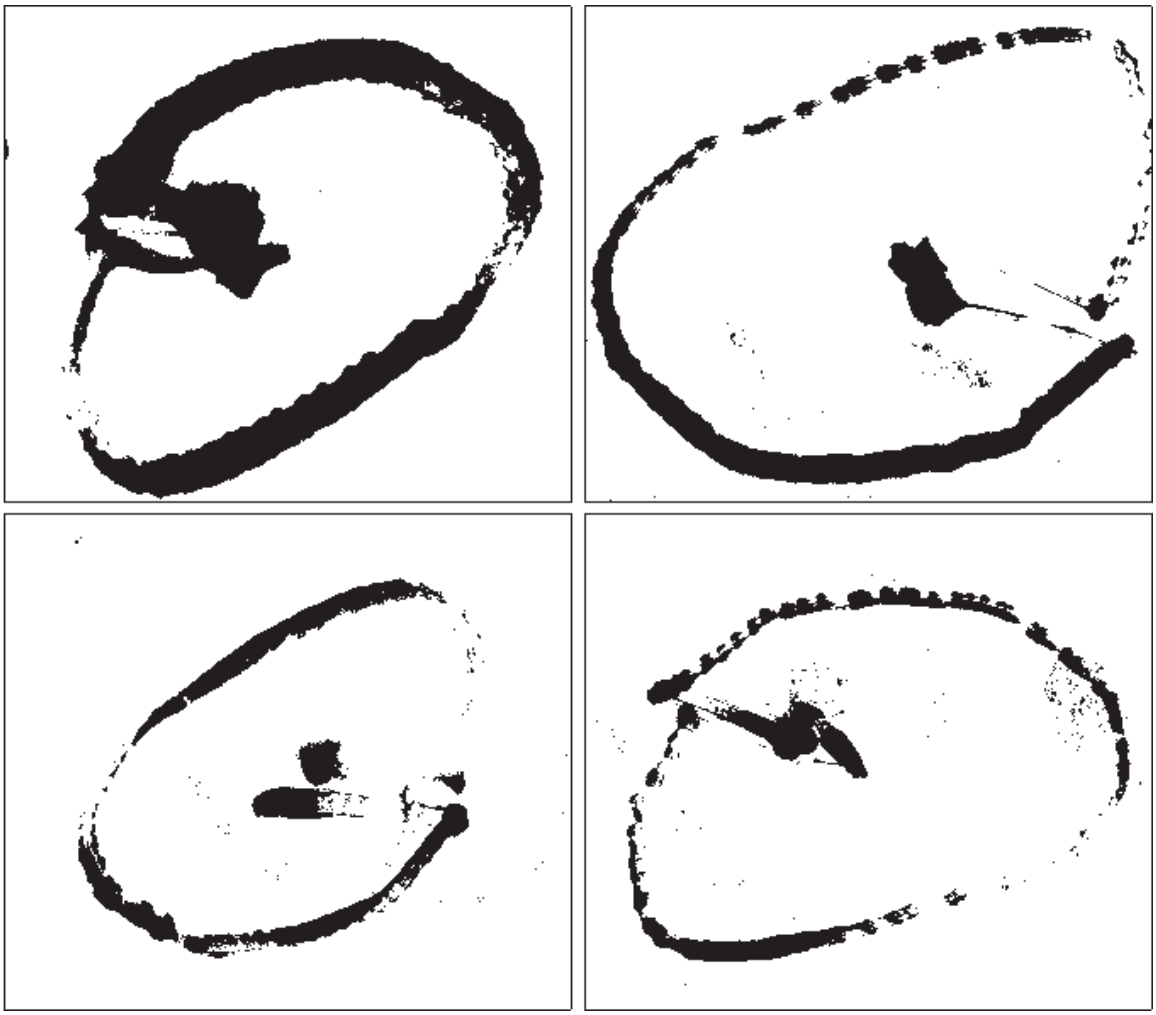

Figura 5.19: Movimiento circular del brazo con coordenadas de un círculo calculado por software

al esperado, causado en gran medida por la incapacidad de la técnica de estimar distancias pequeñas.

En la Figura 5.21 y 5.22 el patrón tiene una semejanza notablemente mayor. El resultado de esta mayor sensibilidad de la técnica propuesta para la navegación inercial permite hacer movimientos de mayor precisión para manipular objetos o herramientas, como en éste caso el uso de una brocha sobre el papel.

5.2.4. Manipulación de objetos

La manipulación de objetos es una prueba cualitativa que permite conocer la experiencia del usuario con el HMI. El objetivo fue recoger un plumón e insertarlo dentro de una caja. La dificultad de esta prueba permite conocer de forma cualitativa si la

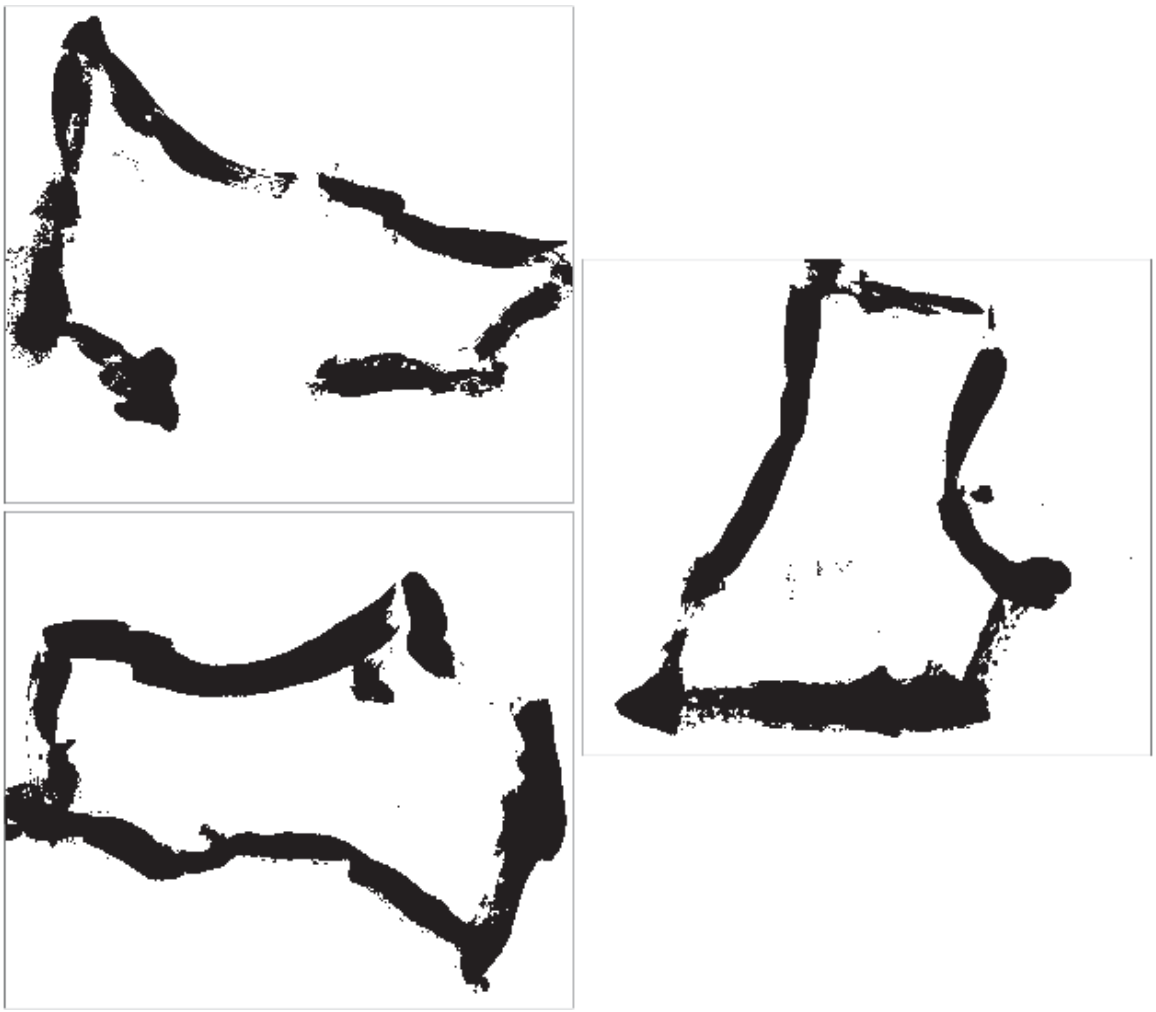

Figura 5.20: Movimiento circular capturado con navegación inercial convencional

Figura 5.21: Movimiento circular capturado con navegación inercial propuesto

Figura 5.22: Movimiento circular capturado con navegación inercial propuesto

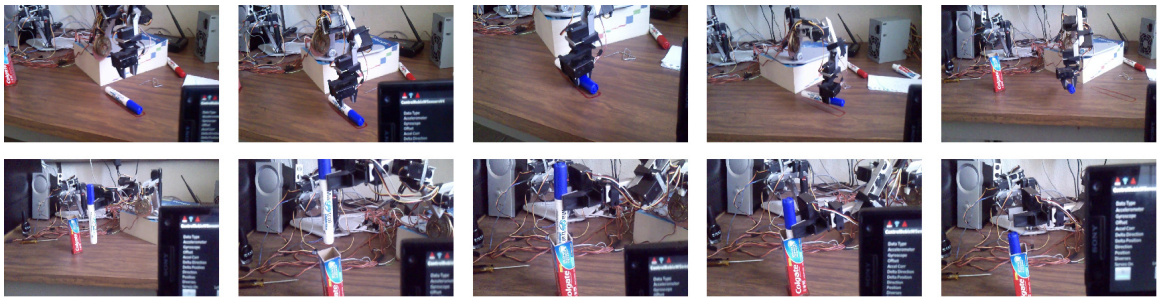

Figura 5.23: Plumon manipulado por el brazo robot controlado con el Smartphone HMI

precisión de las mediciones hechas por el IMU de los gestos del usuario es suficiente para realizar tareas concretas.

En la Figura 5.23 se muestra una secuencia de imágenes donde el brazo controlado por el HMI y haciendo uso de la detección de gestos por medio de la IMU, es llevado desde una posición inicial hasta donde está un plumón para recogerlo y posteriormente llevarlo a una caja para insertarlo dentro del mismo.

El resultado demostró que la interfaz cumple con la precisión suficiente para la manipulación de objetos en tareas medianamente complejas.

Capítulo 6

Conclusiones

En el planteamiento del problema se propuso el desarrollo de una IMU modificada de la técnica de navegación inercial para el uso específico con los smartphone para controlar un brazo manipulador. Esta IMU modificada esta adaptada específicamente para el uso en pequeñas distancias con la precisión de captar los movimientos de un brazo humano sin necesidad de usar referencias externas para corregir los errores de deriva ya que el uso del GPS como sistema auxiliar es imposible por la resolución de los datos que entrega.

Se hizo la implementación de una interfaz hombre maquina que permite capturar los movimientos del brazo del usuario, para manipular un brazo robot de forma inalámbrica usando la comunicación Wi-Fi del smartphone. Para validar la mejora que ofrece la técnica propuesta contra la técnica convencional, se implementaron dos interfases para comparar los resultados de precisión y tener una referencia de la ventaja que se ofrece.

Esta investigación tiene como objetivo ampliar el conocimiento sobre la aplicación del smartphone en una variedad de áreas, por mencionar la médica para la rehabilitación de pacientes, la militar para la tele operación de robots en el desarmado de bombas, la de rescate para localizar y auxiliar víctimas de derrumbes o incendios. También se espera que permita abrir la posibilidad de nuevos nichos de trabajo para personas con discapacidades al poder manipular vehículos para diversas tareas de forma remota.

Los resultados de las pruebas experimentales validan que la técnica propuesta de estimación de patrones ofrece buenos resultados contrarrestando los problemas de deslizamiento lo que permite continuar una mejora de la misma para aumentar su precisión. Se utilizó el patrón generado por una onda senoidal, pero la implementación de patrones más complejos como pueden ser trapezoidales o senoides prometen tener precisiones satisfactorias.

En los análisis de los resultados se observó que la fase entre el patrón estimado y el patrón del movimiento real influye directamente sobre la precisión, lo que sugiere que es importante continuar con la posibilidad de corregir el desfase.

En ésta tesis se usó un brazo manipulador para hacer la interfaz hombre-máquina y probar su aplicación en condiciones reales de funcionamiento. Sobre el hardware del robot se recomienda el uso de un robot que permita el flujo continuo de datos entre el smartphone y robot sin necesidad de esperar la respuesta del término de ejecución de los movimientos para darle más naturalidad al uso de la interfaz. En el diseño del robot es recomendable para trabajo futuro que se implemente en una plataforma de desarrollo con Android para evitar el uso de una PC para realizar la comunicación y control físico del hardware.

También se recomienda la implementación de una técnica de calibración de los sensores para solventar problemas de linealidad en las mediciones, *offset* y rango. Existen varias técnicas desarrolladas pero la aplicación de las mismas deben ser estudiadas para permitir que sea aplicable a los sensores de la mayoría de los *smartphones*.

La comunicación inalámbrica del smartphone haciendo uso del protocolo TCP le da la posibilidad de ser una interfaz remota, para manipular cámaras por ejemplo, sin necesidad de hacer cambios en el software, ya que el uso del *tunneling* permite la comunicación a través de Internet. El smartphone tiene la capacidad de reproducir video y sonido en tiempo real con el protocolo UDP, de tal forma que solo es necesario incluir ésta capacidad a la interfaz. Como ejemplo de aplicación de ésta interfaz es la de tener la posibilidad de manipular cámaras de circuito cerrado desde un smartphone de un agente que se encuentre en movimiento.

Apéndice A

Clases en JAVA más relevantes del HMI

Listing A.1: Clase para la conexión con el brazo manipulador

```
package controlrobixwsensors.app.pkg;
import com.robix.RbxGhostException;
import com.robix.nexway.NexusConnection;
import com.robix.nexway.Nexway;
import com.robix.nexway.Pod;
import com.robix.nexway.Servo;
import com.robix.nexway.ServoProperty;
import com.robix.nexway.Usbor;


public class RobixAPIConector {
       // Program Parameters

       private String NEXUS_HOST = "192.168.123.80";
       private int USBOR_INDEX = 0;
       private int POD_ID      = 1;
       private int SERVO_ID    = 1;

       private int ABSI_THRESH = 250;

       private String MAIN_SCRIPT_NAME = "ScriptAutogenerated";
       private String MACRO_NAME  = "react";
       private String QUICK_SCRIPT;
       private NexusConnection nexusConn;
       private Usbor usbor;
       private Pod pod;
       private Servo servo;
       /** Our one and only Nexway. */
       private Nexway nexway = new Nexway( "ControlRobixWSensors" );

       /**
        * Creates a connection to the Nexus which is running on the
        * specified host.
        *
        * @param host  target host; may be an IP address, a host name, or
        *              'localhost'
        *
        * @return the connected NexusConnection
        */
```

```
public NexusConnection connectToNexus( String host )
{


   // Create nexus connection
   NexusConnection nexusConn = nexway.createNexusConnection( host );

   try
   {
      // Connect to nexus
      nexusConn.connect();
   }
   catch ( Exception ex )
   {
   }
   return nexusConn;
}

/**
 * Returns the specified Usbor.
 *
 * @param nexus  the Nexus which contains the Usbor
 * @param index  index of the Usbor in the specified Nexus
 *               (zero-based)
 *
 * @return the specified Usbor
 */
public Usbor getUsbor( NexusConnection nexus, int index )
{
   System.out.println( "Getting Usbor: " + index );

   // Get Usbor
   Usbor usbor = nexus.getUsborByIndex( index );

   if ( usbor == null )
   {
      System.out.println( "Could not find Usbor." );
      System.exit( 1 );
   }

   return usbor;
}

/**
 * Returns the specified Pod.
 *
 * @param usbor  the Usbor which contains the Pod
 * @param podId  ID of the Pod in the specified Usbor
 *               (one-based)
 *
 * @return the specified Pod
 */
public Pod getPod( Usbor usbor, int podId )
{
   System.out.println( "Getting Pod: " + podId );

   int podCount = usbor.getPodCount();
   if ( podId <= 0 || podId > podCount )
   {
      System.out.println( "Could not find Pod." );
      System.exit( 1 );
   }

   // Get Pod
   return usbor.getPod( podId );
}
```

```
/**
 * Waits for the specified Pod command to finish.
 *
 * @param pod     the Pod
 * @param seqNum  sequence number (aka tracking number) of the
 *                desired Pod command
 *
 * @throws RbxGhostException  if the Pod has become disconnected
 */
public void waitForPodCmdFinished( Pod pod, int seqNum )
   throws RbxGhostException
{
   // Wait for pod command to finish
   while ( !pod.isPodCmdFinished( seqNum ) )
   {
      if ( pod.isGhost() )
         throw new RbxGhostException( "Disconnected from Pod." );

      try
      {
         Thread.sleep( 100 );
      }
      catch ( InterruptedException ex )
      {
         // Ignored in this single-threaded example application.
      }
   }
}


/**
 * Main function.
 */
public void conectar()
{

      ///// Connect and get objects /////
    nexusConn = connectToNexus( NEXUS_HOST );
    usbor = getUsbor( nexusConn, USBOR_INDEX );
    pod = getPod( usbor, POD_ID );
    servo = pod.getServo( SERVO_ID );

      servosPoweron();
      moveTostartpos();
      servosInverted();
      //moveBootharm(300, 45, -234, 643, 1000, -13, 342, -452, 1245, -600, ↩
          1262, -290);
      servosPoweroff();


}

public void servosPoweron()
{
    String script = "power all on ;\n";
    try
    {
        int seqNum = pod.runQuickScript( script );
        waitForPodCmdFinished( pod, seqNum );
    }
    catch ( Exception ex )
       {
          ex.printStackTrace();
       }
}
```

```
    public void servosPoweroff()
    {
        String script = "power all off ;\n";

        try {
            int seqNum = pod.runQuickScript( script );
            waitForPodCmdFinished( pod, seqNum );
        } catch (RbxGhostException e) {
         // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }

    public void servosInverted()
    {
        String script = "invert 2,3,4,5,6,7 on ;\n";
        try
        {
            int seqNum = pod.runQuickScript( script );
            waitForPodCmdFinished( pod, seqNum );
        }
        catch ( Exception ex )
           {
              ex.printStackTrace();
           }
    }

    public void moveTostartpos()
    {
        String script = "move 2,3,4,5,8 to -1600, 6 to -1500, 1,7 to 000, 9 to ↩
            0, 10 to -1690, 11 to 1300, 12 to 1520, 13 to 1430, 14 to -630 ;\n";
        int seqNum;
     try {
         seqNum = pod.runQuickScript( script );
         waitForPodCmdFinished( pod, seqNum );
     } catch (RbxGhostException e) {
         // TODO Auto-generated catch block
         e.printStackTrace();
     }


    }


    public void moveLeftArm(ArmData adata)
    {
        String script =  "move 1 to " + adata.PosA + ", " +
                         "2 to " + adata.PosB + ", " +
                         "3 to " + adata.PosC + ", " +
                         "4 to " + adata.PosD + ", " +
                         "5 to " + adata.PosE + ", " +
                         "6 to " + adata.PosF + ", " +
                         "7 to " + adata.PosG + ", " +
                         "8 to " + adata.PosH + ";\n";
        int seqNum;
        try {
            seqNum = pod.runQuickScript( script );
            waitForPodCmdFinished( pod, seqNum );
        } catch (RbxGhostException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
```

```
public void moveRightArm(ArmData adata)
{
    String script =  "move 9 to " + adata.PosA + ", " +
                     "10 to " + adata.PosB + ", " +
                     "11 to " + adata.PosC + ", " +
                     "12 to " + adata.PosD + ", " +
                     "13 to " + adata.PosE + ", " +
                     "14 to " + adata.PosF + ", " +
                     "15 to " + adata.PosG + ", " +
                     "16 to " + adata.PosH + ";\n";
    int seqNum;
    try {
        seqNum = pod.runQuickScript( script );
        waitForPodCmdFinished( pod, seqNum );
    } catch (RbxGhostException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

public void moveBothArm(ArmData Radata, ArmData Ladata)
{
    String script =  "move 1 to " + Ladata.PosA + ", " +
                     "2 to " + Ladata.PosB + ", " +
                     "3 to " + Ladata.PosC + ", " +
                     "4 to " + Ladata.PosD + ", " +
                     "5 to " + Ladata.PosE + ", " +
                     "6 to " + Ladata.PosF + ", " +
                     "7 to " + Ladata.PosG + ", " +
                     "8 to " + Ladata.PosH + ", " +
                     "9 to " + Radata.PosA + ", " +
                     "10 to " + Radata.PosB + ", " +
                     "11 to " + Radata.PosC + ", " +
                     "12 to " + Radata.PosD + ", " +
                     "13 to " + Radata.PosE + ", " +
                     "14 to " + Radata.PosF + ", " +
                     "15 to " + Radata.PosG + ", " +
                     "16 to " + Radata.PosH + ";\n";
    int seqNum;
    try {
        seqNum = pod.runQuickScript( script );
        waitForPodCmdFinished( pod, seqNum );
    } catch (RbxGhostException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

public void setAccelLeftArm(ArmData adata)
{
    String script =  "accdec 1 " + adata.AcelA + ";\n" +
                     "accdec 2 " + adata.AcelB + ";\n" +
                     "accdec 3 " + adata.AcelC + ";\n" +
                     "accdec 4 " + adata.AcelD + ";\n" +
                     "accdec 5 " + adata.AcelE + ";\n" +
                     "accdec 6 " + adata.AcelF + ";\n" +
                     "accdec 7 " + adata.AcelG + ";\n" +
                     "accdec 8 " + adata.AcelH + ";\n";
    int seqNum;
    try {
        seqNum = pod.runQuickScript( script );
        waitForPodCmdFinished( pod, seqNum );
    } catch (RbxGhostException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
```

```
        }

        public void setAccelRightArm(ArmData adata)
        {
            String script =  "accdec 9 " + adata.AcelA + ";\n" +
                             "accdec 10 " + adata.AcelB + ";\n" +
                             "accdec 11 " + adata.AcelC + ";\n" +
                             "accdec 12 " + adata.AcelD + ";\n" +
                             "accdec 13 " + adata.AcelE + ";\n" +
                             "accdec 14 " + adata.AcelF + ";\n" +
                             "accdec 15 " + adata.AcelG + ";\n" +
                             "accdec 16 " + adata.AcelH + ";\n";
            int seqNum;
            try {
                seqNum = pod.runQuickScript( script );
                waitForPodCmdFinished( pod, seqNum );
            } catch (RbxGhostException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }

        public void setAccelBothArm(ArmData Radata, ArmData Ladata)
        {
            String script =  "accdec 1 " + Ladata.AcelA + ";\n" +
                             "accdec 2 " + Ladata.AcelB + ";\n" +
                             "accdec 3 " + Ladata.AcelC + ";\n" +
                             "accdec 4 " + Ladata.AcelD + ";\n" +
                             "accdec 5 " + Ladata.AcelE + ";\n" +
                             "accdec 6 " + Ladata.AcelF + ";\n" +
                             "accdec 7 " + Ladata.AcelG + ";\n" +
                             "accdec 8 " + Ladata.AcelH + ";\n" +
                             "accdec 9 " + Radata.AcelA + ";\n" +
                             "accdec 10 " + Radata.AcelB + ";\n" +
                             "accdec 11 " + Radata.AcelC + ";\n" +
                             "accdec 12 " + Radata.AcelD + ";\n" +
                             "accdec 13 " + Radata.AcelE + ";\n" +
                             "accdec 14 " + Radata.AcelF + ";\n" +
                             "accdec 15 " + Radata.AcelG + ";\n" +
                             "accdec 16 " + Radata.AcelH + ";\n";
            int seqNum;
            try {
                seqNum = pod.runQuickScript( script );
                waitForPodCmdFinished( pod, seqNum );
            } catch (RbxGhostException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }


}
```

Listing A.2: Clase de rotación de vector de aceleración gravitacional

```
public class Rotacion {

    private double t11, t12, t13;
    private double t21, t22, t23;
    private double t31, t32, t33;
    private double rx, ry, rz;
    private double arx, ary, arz;
    public double vpx, vpy, vpz;
```

```
public void rotavect(double vx, double vy, double vz, double rotx, double roty,↩
     double rotz)
{
    rx = rotx;
    ry = roty;
    rz = rotz;
    t11 = Math.cos(ry)*Math.cos(rz);
    t12 = Math.sin(ry)*Math.sin(rx)-Math.cos(ry)*Math.sin(rz)*Math.cos(rx);
    t13 = Math.cos(ry)*Math.sin(rz)*Math.sin(rx)+Math.sin(ry)*Math.cos(rx);
    t21 = Math.sin(rz);
    t22 = Math.cos(rz)*Math.cos(rx);
    t23 = -Math.cos(rz)*Math.sin(rx);
    t31 = -Math.sin(ry)*Math.cos(rz);
    t32 = Math.sin(ry)*Math.sin(rz)*Math.cos(rx)+Math.cos(ry)*Math.sin(rx);
    t33 = Math.cos(ry)*Math.cos(rx)-Math.sin(ry)*Math.sin(rz)*Math.sin(rx);

    vpx = vx*t11+vy*t21+vz*t31;
    vpy = vx*t12+vy*t22+vz*t32;
    vpz = vx*t13+vy*t23+vz*t33;

}

public void rotavectabs(double vx, double vy, double vz, double rotx, double ↩
    roty, double rotz)
{
    rx = rotx-arx;
    ry = roty-ary;
    rz = rotz-arz;
    arx = rotx;
    ary = roty;
    arz = rotz;

    t11 = Math.cos(ry)*Math.cos(rz);
    t12 = Math.sin(ry)*Math.sin(rx)-Math.cos(ry)*Math.sin(rz)*Math.cos(rx);
    t13 = Math.cos(ry)*Math.sin(rz)*Math.sin(rx)+Math.sin(ry)*Math.cos(rx);
    t21 = Math.sin(rz);
    t22 = Math.cos(rz)*Math.cos(rx);
    t23 = -Math.cos(rz)*Math.sin(rx);
    t31 = -Math.sin(ry)*Math.cos(rz);
    t32 = Math.sin(ry)*Math.sin(rz)*Math.cos(rx)+Math.cos(ry)*Math.sin(rx);
    t33 = Math.cos(ry)*Math.cos(rx)-Math.sin(ry)*Math.sin(rz)*Math.sin(rx);

    vpx = vx*t11+vy*t21+vz*t31;
    vpy = vx*t12+vy*t22+vz*t32;
    vpz = vx*t13+vy*t23+vz*t33;

}

public void calculavectabs(double rotx, double roty, double rotz)
{
    double vx, vy, vz;
    vx = vpx;
    vy = vpy;
    vz = vpz;
    rx = rotx-arx;
    ry = roty-ary;
    rz = rotz-arz;
    arx = rotx;
    ary = roty;
    arz = rotz;

    t11 = Math.cos(ry)*Math.cos(rz);
    t12 = Math.sin(ry)*Math.sin(rx)-Math.cos(ry)*Math.sin(rz)*Math.cos(rx);
    t13 = Math.cos(ry)*Math.sin(rz)*Math.sin(rx)+Math.sin(ry)*Math.cos(rx);
    t21 = Math.sin(rz);
    t22 = Math.cos(rz)*Math.cos(rx);
```

```
        t23 = -Math.cos(rz)*Math.sin(rx);
        t31 = -Math.sin(ry)*Math.cos(rz);
        t32 = Math.sin(ry)*Math.sin(rz)*Math.cos(rx)+Math.cos(ry)*Math.sin(rx);
        t33 = Math.cos(ry)*Math.cos(rx)-Math.sin(ry)*Math.sin(rz)*Math.sin(rx);

        vpx = vx*t11+vy*t21+vz*t31;
        vpy = vx*t12+vy*t22+vz*t32;
        vpz = vx*t13+vy*t23+vz*t33;

    }

    public void setvectaccel(double vx, double vy, double vz)
    {
        vpx = vx;
        vpy = vy;
        vpz = vz;
    }

    public void setvectrot(double rotx, double roty, double rotz)
    {
        arx = rotx;
        ary = roty;
        arz = rotz;
    }

}
```

Listing A.3: Clase de la cinemática inversa del brazo manipulador

```
public class InvertCinem {

    private double anggiro, modulo, x, y, z, xprima, yprima, afx, ladob, ladoa, afy↩
        , hipotenusa, alfa,beta, angbrazo, gamma, angantbr, angmunec, cabeceo;
    private double longpivote = 10;
    private double longbrazo = 6.0;
    private double longantbr = 6.0;
    private double longmuneca = 2;
    private double alturah = 9.5;
    private double maxarmpos = 1480;
    private double cteconv = 2*maxarmpos/Math.PI;

    void InverCinem()
    {
        anggiro=modulo=x=y=z=xprima=yprima=afx=ladob=ladoa=afy=hipotenusa=alfa=beta↩
            =angbrazo=gamma=angantbr=angmunec=cabeceo=0;
        cteconv = maxarmpos/Math.PI;
    }



    public ArmData CalculaCinemB(double Xpos, double Ypos, double Zpos, double dirX↩
        , double dirZ, double herramienta)
    {
        ArmData ArtData = new ArmData();

        x = Xpos;
        y = Ypos;
        z = Zpos;
        cabeceo = dirX;

        anggiro = Math.atan2(y, x);
        modulo = Math.sqrt(Math.pow(x, 2) + Math.pow(y, 2));
        xprima = modulo;
        yprima = z;
```

```
        afx = Math.cos(cabeceo)*longmuneca;
        ladob=xprima-afx-longpivote;
        afy=Math.sin(cabeceo)*longmuneca;
        ladoa=yprima-afy-alturah;
        hipotenusa=Math.sqrt(Math.pow(ladoa, 2) + Math.pow(ladob, 2));
        alfa = Math.atan2(ladoa, ladob);
        beta = Math.acos((Math.pow(longbrazo,2)-Math.pow(longantbr,2)+Math.pow(↩
            hipotenusa,2))/(2*longbrazo*hipotenusa));
        angbrazo = alfa + beta;
        gamma = Math.acos((Math.pow(longbrazo,2)+Math.pow(longantbr,2)-Math.pow(↩
            hipotenusa,2))/(2*longbrazo*longantbr));
        angantbr = Math.PI - gamma;
        angmunec = cabeceo - angbrazo + angantbr;

        ArtData.PosA = (int) (anggiro * cteconv);
        ArtData.PosB = -(int) ((angbrazo) * cteconv);
        ArtData.PosC = -(int) ((angantbr) * cteconv);
        ArtData.PosD = -(int) ((angmunec) * cteconv);
        ArtData.PosE = (int) (dirZ * cteconv);
        ArtData.PosF = (int) (herramienta * cteconv);

        return ArtData;
    }


}
```

Listing A.4: Clase para la estimación para la posición angular

```
public class GyroVelToPos {

    private KalmanFilter filtroK0, filtroK1;
    private double vel, posfiltered, pos, sampleno, maxvel, fsampling, fcpa, fcpb;
    int estimate;


    GyroVelToPos()
    {
        fsampling = 30;
        fcpa = 3;
        fcpb = 10;
        filtroK0 = new KalmanFilter(fsampling);
        filtroK1 = new KalmanFilter(fsampling);
        posfiltered = vel = pos = sampleno = 0;
        estimate=0;

    }

    public void SetNewSample(double velsamp)
    {
        vel = filtroK0.sampleK(velsamp);

        if (estimate == 1)
        {
            pos += vel/fsampling;

            posfiltered = pos;

            if (vel<0)
            {
                if(maxvel<-vel) maxvel = -vel;
            }
            else
            {
```

```
                if(maxvel<vel) maxvel = vel;
            }

        }


    }

    public void startEstimation()
    {
        estimate = 1;
    }

    public void stopEstimation()
    {
        estimate = 0;
    }

    public void resetEstimation()
    {
        vel = pos = maxvel = 0;
    }


}
```

Listing A.5: Clase para la de la posición por patrones

```
public class AccelAcelToPos {

    private KalmanFilter filtroK0, filtroK1, filtroK2, filtroK3 ;
    private double acelcoffilter, acelcoff, offset, acel, vel, pos, ceroadj, ↩
        maxacelpos, maxacelneg, maxacel, fsampling, fcpb, fcpa, wangle, direction;
    int estimate, sampcapt, faseno, maxacelpossamp, maxacelnegsamp;
    private double trigger = 0.5;


    AccelAcelToPos()
    {
        fsampling = 30;
        fcpb = 2;
        fcpa = 1;
        filtroK0 = new KalmanFilter(fsampling);
        filtroK1 = new KalmanFilter(fsampling);
        filtroK2 = new KalmanFilter(fsampling);
        filtroK3 = new KalmanFilter(fsampling);
        acelcoff= offset = acel = vel = pos = 0;
        estimate=0;

    }

    public void SetNewSample(double acelsamp, double aceloffset)
    {

        acelcoff = acelsamp;
        acelcoffilter = filtroK0.sampleK(acelsamp);

        offset = aceloffset;

        acel = acelcoffilter - offset;


        if (estimate == 1)
        {
```

```
            if (faseno==0)
            {
                if (acel<=-trigger)
                {
                    faseno=1;
                    sampcapt = 0;
                }
                if (acel>=trigger)
                {
                    faseno=1;
                    sampcapt = 0;
                }
            }

            else
            {
                if (acel<0)
                {
                    if(maxacel<-acel)
                    {
                        maxacelneg = -acel;
                        maxacelnegsamp = sampcapt;
                    }
                }
                else
                {
                    if(maxacel<acel)
                    {
                        maxacelpos = acel;
                        maxacelpossamp = sampcapt;
                    }
                }
                sampcapt += 1;
            }

        }

    }

    public void startEstimation()
    {
        estimate = 1;

    }

    public void stopEstimation()
    {
        estimate = 0;

        if (maxacelpossamp<maxacelnegsamp) direction = 1;
        else direction = -1;

        if (maxacelpos>maxacelneg) maxacel = maxacelpos;
        else maxacel = maxacelneg;

        if (faseno!=0)
        {
            wangle=2*Math.PI/(double)sampcapt;
            for (int lap=0; lap<=sampcapt; lap+=1)
            {
                vel += direction*maxacel*Math.sin(wangle*(double)lap)/fsampling;
                pos += vel/fsampling;
            }
        }
```

```
    }

    public void resetEstimation()
    {
        vel = pos = maxacel = 0;
        sampcapt = 0;
        faseno=0;
    }

}
```

Apéndice B

Biblioteca para Java Rascal

Listing B.1: Bibliotecas Robix para lenguaje Java

```
import com.robix.RbxGhostException;
import com.robix.nexway.NexusConnection;
import com.robix.nexway.Nexway;
import com.robix.nexway.Pod;
import com.robix.nexway.Servo;
import com.robix.nexway.ServoProperty;
import com.robix.nexway.Usbor;
```

Listing B.2: Método de codificación de datos en el script de Robix

```
public void moveLeftArm(ArmData adata)
	{
	    String script =  "move 1 to " + adata.PosA + ", " +
	                     "2 to " + adata.PosB + ", " +
	                     "3 to " + adata.PosC + ", " +
	                     "4 to " + adata.PosD + ", " +
	                     "5 to " + adata.PosE + ", " +
	                     "6 to " + adata.PosF + ", " +
	                     "7 to " + adata.PosG + ", " +
	                     "8 to " + adata.PosH + ";\n";
	    int seqNum;
	    try {
	        seqNum = pod.runQuickScript( script );
	        waitForPodCmdFinished( pod, seqNum );
	    } catch (RbxGhostException e) {
	        // TODO Auto-generated catch block
	        e.printStackTrace();
	    }
	}
```

Bibliografía

[1] Xiaoli Yang, Dorina C. Petriu, Thom E. Whalen, Emil M. Petriou. A Web.Based 3D Virtual Robot Remote Control System *CCECE 2004 - CCGEI 2004*, .

[2] Cristiano Spelta, Vincenzo Manzoni, Andrea Corti, Andrea Goggi, Sergio Matteo Savaresi. Smartphone-Based Vehicle-to-Driver/Environment Interaction System for Motorcycles. *IEEE EMBEDDED SYSTEMS LETTERS*, VOL. 2, NO. 2, JUNE 2010.

[3] Daryush D. Mehta, Matías Zanartu, Shengran W. Feng, Harold A. CheyneII, Robert E. Hillman. Mobile Voice Health Monitoring Using a Wearable Accelerometer Sensor and Smartphone Platform. *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING*, VOL. 59, NO. 11, NOVEMBER 2012.

[4] Hyojeong Shin, Yohan Chon, Hojung Cha. Unsupervised Construction of an Indoor Floor Plan Using a Smartphone. *IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS - PART C: APPLICATIONS AND REVIEWS*, VOL. 42, NO. 6, NOVEMBER 2012.

[5] Joseph J. Oresko, Zhanpeng Jin, Jung Cheng, Shimeng Huang, Yuwen Sun, Heather Duschl, Allen C. Cheng. A Wearable Smartphone-Based Platform for Real-Time Cardiovascular Disease Detecnion Via Electrocardiogram Processing. *IEEE TRANSACTIONS ON INFORMATION TECHNOLOGY IN BIOMEDICINE*, VOL. 14, NO. 3, 2010.

[6] . Emmanouil Koukoumidis, Margaret Martonosi, Li-Shiuan Peh. Leveraging Smartphone Cameras for Collaborative Road Advisories. *IEEE TRANSACTIONS ON MOBILE COMPUTING*, VOL. 11, NO. 5, MAY 2012.

[7] Jeonghee Kim, Xueliang Huo, Julia Minocha, Jaimee Holbrook, Anne Laumann, Maysam Ghovanloo. Evaluation of a Smartphone Platform as a Wireless Interface Between Tongue Drive System and Electric-Powered Wheelchairs. *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING*, VOL. 59, NO. 6, JUNE 2012.

[8] Aksel A. Transeth, Oystein Skotheim, Henrik Shumann-Olsen, Gorm Johansen, Jens Thielemann and Erik Kyrkjebo. A Robotic Concept for Remote Maintenance Operations: A Robust 3D Objet Detection and Pose Estimation Method and a

Novel Robot Tool. *The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 18-22, 2010, Taipei, Taiwan.

[9] Byung-Hyug Lee, Sheng-Hai An, Dong-Ryeol Shin. A Remote Control Service for OSGi-based Unmanned Vehicle using Smartphone in Ubiquitous Environment. *2011 Third International Conference on Computational Intelligence, Communication Systems and Networks* ,.

[10] Qiang Li, Weijun Quin, Liqun Li, Limin Sun. Poster Abstract: Smartphone Heterogeneous Network Handoff Based on the closed Control Loop *2010 IEEE/ACM Third International Conference on Cyber-Physical Systems.*

[11] Sung Wook Moon, Young Jin Kim, Ho Jun Myeong, Chang Soo Kim, Nam Ju Cha, Dong Hwan Kim. Implementation of Smartphone Evironment Remote Control and Monitoring System for android Operating System-based Robot Platform. *International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, Nov 23-26, 2011 in Songdo ConventiA, Incheon, Korea.

[12] Hou-Tsan Lee, Hsiang-lin Tsai, Zhong-Quan Chen, Yu-Ting Jiang, Jia-Xing He, Yu-Chi Lin. Mobile Detecting Robot with IPCam Feedback *SICE Annual Conference 2012* , August 20-23, 2012, Akita University, Akita, Japan.

[13] Huy-Kyung Oh, In-Cheol Kim. Hybrid Control Architecture of the Robotic Surveillance System Using Smartphones *2011 8th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)* , Nov 23-26, 2011, in Songdo ConventiA, Incheon Korea.

[14] Cinemática Inversa III - Robótica *https://sites.google.com/site/proyectosroboticos/cinematica-inversa-iii*,

[15] Enrique Mandado Perez. Instrumentacion Electronica *MARCOMBO S.A.*, 1995, España.

[16] Robotica *http://proton.ucting.udg.mx/materias/robotica/r166/r49.htm*

[17] Joel C. Perry, Jacob Rosen. Design of a 7 Degree-of-Freedom Upper-Limb Powered Exoskeleton *The First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics*, 2006.

[18] Terrence Fong, Chris Provencher, Mark Micire, Myron Diftler, Geginald Berka, Bill Bluethmann, David Mittman The Human Exploration Telerobotics project: Objectives, approach, and testing *Aerospace Conference*, 2012 IEEE .

[19] Amber M. Walker, David P. Miller Tele-operated robot control using attitude aware smartphones *International Conference on Human-Robot Interaction (HRI) 7th ACM/IEEE*, March 5-8, 2012, Boston, Massachusetts, USA.

[20] Fausto Ferreira, Marco Bibuli, Massimo Caccia, Giorgio Bruzzone, Gabriele Bruzzone Enhancing autonomous capabilities and human-robot interaction for unmanned sufrace vehicles *Mediterranean Conference on Control and Automation (MED)*, July 3-6, 2012, Barcelona, Spain.

[21] Jong Hyun Park, Tae Houn Song, Ji Hwan Park, Jae Wook Jeon Usability Analysis of a PDA-based user interface for mobile robot teleoperation *The IEEE International Conference on Industrial Information (INDIN 2008)*, july 13-16, 2008, DCC, Daejeon, Korea.

[22] Saso Koceski, Natasa Koceska, Ivica Kocev Design and evaluation of cell phone pointing interface for robot control *International Journal of Advanced Robotic systems*, 2012, Vol9.

[23] Martínez A. Gloria M., Jáquez O. Sonia A., Rivera M. José y Sandoval R. Rafael Diseño propio y Construcción de un Brazo Robótico de 5 GDL *RIEEC, REVISTA DE INGENIERIA ELECTRICA, ELECTRÓNICA Y COMPUTACIÓN* Vol. 4 No. 1, JULIO, 2008.

[24] Chin-Woo Tan, Sungsu Park Design of Accelerometer-Based Inertial Navigation Systems IEEE Transactions on instrumentation and measurement,Vol. 54, No. 6 , Dec 2005.

[25] Jan Wendel, Oliver Meister, Christian Schlaile, Gert Trommer An integrated GPS/MEMS-IMU navigation system for an autonomous helicopter Aerospace Science and Technology, 527-533, 2006.

[26] Isaac Skog, Peter Handel Calibration of a MEMS inertial measurement unit XVII IMEKO WORLD CONGRESS Metrology for a sustainable Development, Sep 17-22, 2006 in Rio de Janeiro, Brazil.

[27] Carlos Parga, Xiaoou Li, Wen Yu Tele-Manipulation of Robot Arm with Smartphone IEEE 6th International Symposium on Resilient Control Systems, Ago 13-15, 2013 in San Francisco, United States.

[28] Carlos Parga, Xiaoou Li, Wen Yu Smartphone-based Human Machine Interface with Application to Remote Control of Robot Arm IEEE 10th International Conference on Electrical Engineering, Computing Science and Automatic Control, Sep 30 - Oct-4, 2013 in Mexico City, Mexico.

[29] Carlos Parga, Xiaoou Li, Wen Yu Smartphone-based Human Machine with Application to Remote Control IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS Ago 13-16, 2013 in Manchester, UK.