Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica Eléctrica

DISEÑO DE DISPOSITIVO NO INVASIVO PARA RASTREO Y DETECCIÓN DE MOVIMIENTO GIRATORIO DE CABEZA MEDIANTE SISTEMAS MICROELECTROMECÁNICOS PARA CONTROL DE ORDENADOR PERSONAL

Vandria Eunise Alvarez Alvarez
Asesorado por el Ing. Byron Odilio Arrivillaga Méndez

Guatemala, septiembre de 2013

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

DISEÑO DE DISPOSITIVO NO INVASIVO PARA RASTREO Y DETECCIÓN DE MOVIMIENTO GIRATORIO DE CABEZA MEDIANTE SISTEMAS MICROELECTROMECÁNICOS PARA CONTROL DE ORDENADOR PERSONAL

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

VANDRIA EUNISE ALVAREZ ALVAREZ
ASESORADO POR EL ING. BYRON ODILIO ARRIVILLAGA MÉNDEZ

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO ELECTRÓNICO

GUATEMALA, SEPTIEMBRE DE 2013

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	Ing. Alfredo Enrique Beber Aceituno
VOCAL II	Ing. Pedro Antonio Aguilar Polanco
VOCAL III	Inga. Elvia Miriam Ruballos Samayoa
VOCAL IV	Br. Walter Rafael Véliz Muñoz
VOCAL V	Br. Sergio Alejandro Donis Soto
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

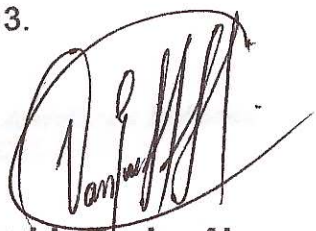
DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Ing. Carlos Eduardo Guzmán Salazar
EXAMINADORA	Inga. Ingrid Salomé Rodríguez de Loukota
EXAMINADOR	Ing. Otto Fernando Andrino González
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

DISEÑO DE DISPOSITIVO NO INVASIVO PARA RASTREO Y DETECCIÓN DE MOVIMIENTO GIRATORIO DE CABEZA MEDIANTE SISTEMAS MICROELECTROMECÁNICOS PARA CONTROL DE ORDENADOR PERSONAL

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería Mecánica Eléctrica, con fecha 27 de mayo de 2013.



Vandria Eunise Alvarez Alvarez

Guatemala, 30 de julio de 2013

Ingeniero
Carlos Eduardo Guzmán Salazar
Coordinador de Área de Electrónica
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería
Universidad de San Carlos de Guatemala

Señor Coordinador:

Por este medio tengo el gusto de informar a usted, que he concluido con el asesoramiento del trabajo de graduación con título: **Diseño de dispositivo no invasivo para rastreo y detección de movimiento giratorio de cabeza mediante sistemas microelectromecánicos para control de ordenador personal**, desarrollado por la estudiante Vandria Eunise Alvarez Alvarez, con carné 200819065. Después de revisar su contenido final doy mi entera aprobación al mismo.

Atentamente,

Ing. Byron Arrivillaga Méndez
Col. 5217

Ing. Byron Odilio Arrivillaga Méndez
Colegiado No. 5217

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

Ref. EIME 59. 2013
Guatemala, 1 de AGOSTO 2013.

Señor Director
Ing. Guillermo Antonio Puente Romero
Escuela de Ingenierìa Mecànica Elèctrica
Facultad de Ingenierìa, USAC.

Señor Director:

Me permito dar aprobación al trabajo de Graduación titulado: DISEÑO DE DISPOSITIVO NO INVASIVO PARA RASTREO Y DETECCIÓN DE MOVIMIENTO GIRATORIO DE CABEZA MEDIANTE SISTEMAS MICROELECTROMECÁNICOS PARA CONTROL DE ORDENADOR PERSONAL, de la estudiante Vandria Eunise Alvarez Alvarez que cumple con los requisitos establecidos para tal fin.

Sin otro particular, aprovecho la oportunidad para saludarle.

Atentamente,
ID Y ENSEÑAD A TODOS

Ing. Carlos Eduardo Guzmán Salazar
Coordinador Área Electrónica

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION ESCUELA DE INGENIERIA MECANICA ELECTRICA
FACULTAD DE INGENIERIA

sto

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

REF. EIME 59. 2013.

El Director de la Escuela de Ingeniería Mecánica Eléctrica, después de conocer el dictamen del Asesor, con el Visto Bueno del Coordinador de Área, al trabajo de Graduación de la estudiante; VANDRIA EUNISE ALVAREZ ALVAREZ titulado: DISEÑO DE DISPOSITIVO NO INVASIVO PARA RASTREO Y DETECCIÓN DE MOVIMIENTO GIRATORIO DE CABEZA MEDIANTE SISTEMAS MICROELECTROMECÁNICOS PARA CONTROL DE ORDENADOR PERSONAL, procede a la autorización del mismo.

Ing. Guillermo Antonio Puente Romero

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION ESCUELA DE INGENIERIA MECANICA ELECTRICA
FACULTAD DE INGENIERIA

GUATEMALA, 22 DE AGOSTO 2,013.

Universidad de San Carlos
De Guatemala

Facultad de Ingeniería
Decanato

Ref. DTG.640-2013

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería Mecánica Eléctrica, al trabajo de graduación titulado: **DISEÑO DE DISPOSITIVO NO INVASIVO PARA RASTREO Y DETECCIÓN DE MOVIMIENTO GIRATORIO DE CABEZA MEDIANTE SISTEMAS MICROELECTROMECÁNICOS PARA CONTROL DE ORDENADOR PERSONAL,** presentado por la estudiante universitaria: **Vandria Eunise Alvarez Alvarez,** autoriza la impresión del mismo.

IMPRÍMASE.

Ing. Murphy Olympo Paiz Recinos
Decano

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

Guatemala, septiembre de 2013

/cc

ACTO QUE DEDICO A:

A Dios	Por todo lo bueno que ha traído a mi vida
Mis padres	César Alvarez y Miriam Alvarez, por su apoyo, amor incondicional y por ser el pilar en mi vida que hizo posible este logro.
Mi familia	Por la confianza y fortaleza brindada, en especial a mis hermanas y mis abuelos, quienes siempre creyeron en mí.
Mis amigos	Gracias por su ayuda, por los días y noches de desvelo trabajando para alcanzar esta meta en común, en especial a Juan Luis Vega por siempre estar a mi lado.

AGRADECIMIENTOS A:

Ingenieros	Byron Arrivillaga, por su apoyo en la asesoría de este trabajo.
	Iván Morales, por su ayuda y consejo en el desarrollo de este trabajo.
Mis catedráticos	Por compartir sus conocimientos y experiencias conmigo.
Universidad de San Carlos de Guatemala	Por darme la oportunidad de culminar esta importante etapa estudiantil en mi vida.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
g	Gravedad
Hz	Hertz
KHz	Kilohertz
m	Metro
ms	Milisegundo
s	Segundo

GLOSARIO

Acelerómetro Dispositivo para medir la aceleración que experimenta un cuerpo al moverse, usualmente se mide respecto a la gravedad.

ADXL330 Acelerómetro de 3 ejes que proporciona como salida una señal de voltaje igual al vector de aceleración que se mide.

Algoritmo Conjunto de pasos o instrucciones ordenado y simple que permiten definir de manera más fácil un procedimiento a ejecutar.

Amplificador Dispositivo electrónico que mediante el uso de energía, aumenta la magnitud de una señal.

Ancho de banda Rango de frecuencias medido en Hz en el que se concentra la mayor cantidad de potencia de una señal.

Bit Dígito binario. Es la unidad mínima de información en comunicaciones informáticas.

Buffer	Espacio de memoria que permite el almacenaje temporal de datos que se envían desde un dispositivo hacia otro, asegurando la obtención y no pérdida de datos para el receptor.
Byte	Unidad de información que representa un múltiplo del *bit.* Un *byte* son ocho *bits* cuya agrupación define información.
C++	Lenguaje de programación orientado a objetos que resulta de la extensión del lenguaje de programación C. Permite la manipulación de objetos y también el uso de programación genérica.
Clase	Es una construcción que permite crear tipos personalizados propios mediante la agrupación de variables de otros tipos, métodos y eventos.
Cuadriplejía	Tipo de lesión en la médula espinal que provoca la pérdida total o parcial de las extremidades que se encuentran por debajo del cuello.
Cuaterniones	Representación matemática de las orientaciones y las rotaciones de un objeto en tres dimensiones.
Debian	Es un sistema operativo de uso libre que ya se encuentra precompilado y empaquetado para diferentes arquitecturas de ordenadores.

DMP	Procesador Digital de Movimiento (Digital Motion Processor). Dispositivo que realiza análisis matemático sobre datos de aceleración y velocidad angular proporcionando cuaterniones.
Driver	Controlador de dispositivo. Es un programa informático que permite al sistema operativo manejar un periférico creando una interfaz entre hardware y software.
eSpeak	Programa sintetizador de voz. Transforma texto en voz, que puede ser manejado desde una consola de texto.
FIFO	Primero en entrar, primero en salir (first in, first out). Memoria en la que los datos que primero entran son los primeros que salen.
Frecuencia	Magnitud que mide el número de repeticiones por unidad de tiempo de cualquier fenómeno que presenta periodicidad en la ocurrencia.
Giroscopio	Dispositivo mecánico que mide la orientación de un cuerpo respecto a un eje de simetría.
GPIO	Entrada/Salida de propósito general (General Purpose Input/Output). Pin de conexión físico en la tarjeta Raspberry Pi.

GUI	Interfaz Gráfica de Usuario (Graphic User Interface). Conjunto de formas y métodos que posibilitan la interacción de un sistema con los usuarios utilizando formas gráficas e imágenes.
Hardware	Conjunto de componentes físicos de un sistema.
HDMI	Interfaz multimedia de alta definición (High-Definition Multimedia Interface). Interfaz normada que provee audio y video digital cifrado sin compresión.
I2C	Inter-Circuitos Integrados (Inter-Circuits Integrated). Es un bus de comunicaciones en serie que utiliza únicamente dos líneas para transmitir la información, datos y reloj.
Kernel	Software principal del sistema operativo, se encarga del acceso seguro al hardware de los programas.
Librería	Conjunto de implementaciones de comportamiento, escritas para un lenguaje de programación en particular. Usualmente esta no funciona de manera autónoma ya que es llamada por otros programas.
Linux	Núcleo libre de sistema operativo basado en Unix y diseñado como alternativa frente a otros como Windows o Mac.

MEMS	Sistemas Microelectromecánicos (Microelectromechanical Systems). Dispositivos de escala micrométrica que integran funciones electromecánicas en circuitos electrónicos.
Módulo	Pertenecen o son parte de un programa que ejecuta tareas en específico al recibir datos propios del programa principal o de otros módulos.
MPU6050	Unidad de Procesamiento de Movimiento (Motion Processing Unit). Dispositivo que combina un giroscopio de 3 ejes y un acelerómetro de 3 ejes con un DMP para procesar complejos algoritmos de fusión de movimiento.
Pin	Clavija, en español. Es una terminal o patilla a cada uno de los contactos metálicos de un conector o material conductor de electricidad.
Puerto serial	Interfaz de comunicaciones digitales donde la información se transmite *bit* a *bit,* enviando un solo *bit* a la vez.
pyMouse	Módulo que encapsula las herramientas para el manejo del mouse desde Python.
pySerial	Módulo que encapsula el acceso al puerto serial desde Python.

Python	Lenguaje de programación interpretado que ofrece sintaxis más simple. Soporta programación orientada a objetos, imperativa y hasta funcional.
RAM	Memoria de acceso aleatorio (Random Access Memory). Almacena la información temporal a la que el procesador necesita acceder durante cada tarea que ejecuta.
SD	Digital Seguro (Secure Digital). Es un formato de tarjetas de memoria que usualmente se utiliza para almacenamiento en componentes electrónicos.
Sistema Operativo	Conjunto de programas que en un sistema informático gestiona los recursos de hardware con los con los programas de aplicación.
Software	Conjunto de componentes lógicos de un sistema.
UART	Transmisor-Receptor Asíncrono Universal (Universal Asynchronous Receiver-Transmitter). Realiza el control sobre los dispositivos que se manejan en el puerto serial.
USB	Bus universal en serie (Universal Serial Bus). Es el estándar que define el tipo de cable, conector y protocolos para comunicar y proveer de energía a los diferentes dispositivos electrónicos que se interconectan.

Vector	Magnitud física que se describe por la longitud, orientación y sentido.
Velocidad angular	Medida de la velocidad de rotación. Se define como el ángulo girado por unidad de tiempo.
Yaw, roll y pitch	Ángulos que describen la posición y orientación de un objeto en tres dimensiones de acuerdo a tres ejes de referencia.

RESUMEN

El propósito de este documento es realizar el diseño de un dispositivo capaz de manipular el cursor mediante el movimiento giratorio de la cabeza, que permitirá a una persona cuadripléjica, desempeñar actividades básicas en un computador.

Se inicia describiendo la relación usuario dispositivo y los sistemas microelectromecánicos (sensores), protocolos de comunicación, así como características relevantes de la microcomputadora (Raspberry Pi).

Seguido se indican las configuraciones iniciales que deben realizarse en la Raspberry Pi y la instalación de paquetes. Luego se detalla el proceso de adquisición de datos desde el MPU6050 hacia la Raspberry Pi, se hace uso del lenguaje de programación C++, donde se han desarrollado todos los programas que realizan tareas tales como: configuración del MPU6050, configuración del DMP, funciones de lectura de datos, entre otros.

Por último se explica la obtención y el análisis de datos por medio de los programas desarrollados en Python para mover el cursor. Empezando por el manejo del puerto serial, rutina de calibración, detección de clic y características adicionales que se han agregado, como la GUI y síntesis de voz. Finalizando con todas las funciones que cotejan la información y colocan al cursor en la posición deseada.

OBJETIVOS

General

Realizar el diseño de dispositivo no invasivo para rastreo y detección de movimiento giratorio de cabeza, mediante sistemas microelectromecánicos que permita a una persona manejar el ordenador personal.

Específicos

1. Exponer los fundamentos de asistencia tecnológica, para delimitar las herramientas de software y hardware a utilizar.

2. Detallar las configuraciones iniciales necesarias, para el buen desempeño de la computadora Raspberry Pi.

3. Describir el método de adquisición de datos desde los sistemas microelectromecánicos.

4. Presentar el análisis y procesamiento de datos en tiempo real realizado en Python para mover el cursor.

INTRODUCCIÓN

En la actualidad existen múltiples soluciones desde el punto de vista electrónico para controlar el cursor de una computadora, sin embargo, en la mayoría, dejan por fuera a aquellas personas con capacidades limitadas.

Con el afán de brindar otra solución adicional a este inconveniente, se diseña un dispositivo que permitirá manejar el cursor de la computadora por medio del movimiento giratorio de la cabeza.

Para el desarrollo, se divide a esta en varias etapas. La primera, engloba como en todo inicio de proyecto, la delimitación de las herramientas a utilizar en concordancia con las capacidades del usuario. Luego, en la segunda se identifican y ejecutan todos los procedimientos y configuraciones que son por defecto parte de los sistemas utilizados.

La tercera etapa involucra la adquisición de datos a través del control y comunicación entre computadora y los sistemas microelectromecánicos cuando el usuario realiza movimientos. Por último, la cuarta etapa es donde el análisis de los datos es llevado a cabo y cuyo resultado determina la posición en tiempo real del cursor.

1. DELIMITACIÓN DE HERRAMIENTAS DE ASISTENCIA TECNOLÓGICA

La asistencia tecnológica relaciona a un dispositivo electrónico y un operador humano; todo esto alrededor de una actividad que puede ejecutar este último. Basado en el fundamento teórico se delimitan las herramientas que mejor se adecúan para el diseño del dispositivo.

1.1. Sistemas tecnológicos

Un sistema tecnológico es un conjunto de equipos informáticos y software interconectados por medio de dispositivos físicos que envían y reciben impulsos eléctricos, ondas electromagnéticas, haces de luz, o cualquier otro tipo de medio para el transporte de datos, con la finalidad de compartir información, recursos y servicios.

El concepto que define un sistema tecnológico en rehabilitación es la relación entre un dispositivo tecnológico y un operador humano el cual tiene una discapacidad; todo esto alrededor de una actividad que puede ejecutar este último.

En un sistema tecnológico de asistencia se deben considerar los sistemas sensoriales visual, auditivo, táctil propioceptivo, kinestésico y vestibular. Los efectos se pueden describir como los elementos neural, muscular y esquelético del cuerpo humano que proveen movimiento.

Existen tres elementos relevantes respecto al criterio de selección y manipulación de la interfaz en dispositivos utilizados en tecnologías de asistencia: el control de interfaz, la selección de ajuste y método de selección. Estas tres partes se interrelacionan con el objetivo de optimizar el desempeño a nivel humano y tecnológico en lo referido a manejo del sistema.

1.2. Características principales de la cuadriplejía

La cuadriplejía es un tipo de lesión en la médula espinal (SCI sigla inglesa) que también recibe el nombre de tetraplejía. Esta puede darse en una persona si la médula espinal se lesiona o presenta ciertas enfermedades. La médula espinal hace parte del sistema nervioso central el cual permite que el cerebro se comunique con el cuerpo.

La palabra cuadriplejía significa que la parte de médula espinal que está dentro del cuello ha sido lesionada. Esta lesión causa la pérdida de sensibilidad y movimiento en brazos, piernas y torso o tronco del cuerpo. Las lesiones en la médula espinal son descritas de acuerdo al sitio de las lesiones en ella. Los médicos utilizan letras y números para describir el sitio donde la médula espinal ha sido lesionada. La letra C es usada para describir la lesión cervical. Esto es la lesión producida en la parte de médula espinal que está a la altura del cuello. La espina dorsal está compuesta por 8 vértebras cervicales, 12 torácicas, 5 lumbares y 4 sacras.

Los síntomas de la cuadriplejía son distintos de acuerdo al sitio y gravedad de la lesión en la médula espinal. Algunos de estos son:

- Músculos débiles o flácidos, especialmente en brazos y piernas

- Incapacidad para controlar el intestino (evacuaciones intestinales) o la vejiga o capacidad de orinar.
- Baja presión arterial
- Dificultad para respirar o incapacidad de respirar por sus propios medios
- Incapacidad para mover o sentir algo más abajo del área lesionada

1.3. Relación usuario dispositivo

Luego de conocer las capacidades limitadas que posee una persona cuadripléjica, se identifican aquellas tareas que si puede realizar y cuya acción deriva en eventos medibles.

Entre estas se encuentran:

- Movimiento vertical de la cabeza
- Movimiento horizontal de la cabeza (limitado)
- Movimiento vertical del hombro
- Capacidad de habla (limitada en algunos casos)
- Capacidad de escucha (limitada en algunos casos)
- Visibilidad

Al realizar el análisis y determinar cuál sería la mejor la solución para resolver el problema, se encontró que para el paciente sería más cómodo realizando el movimiento de la cabeza, ya que siempre es necesario estar observando a la pantalla cuando se mueve el cursor.

Seguido se identificaron los componentes electrónicos que podrían ayudar en la medición de los movimientos de la cabeza, tales como:

- Acelerómetro
- Giroscopio
- Sensores infrarrojos
- Sensores ultrasónicos
- Sensores capacitivos

Los últimos tres fueron descartados debido a la incerteza causada por factores externos, como; luminosidad, ruido eléctrico y distancia del paciente hacia la pantalla. Aunque estos también pueden afectar los resultados de las dos primeras opciones, lo hacen en menor grado y se tiene mayor facilidad para controlarles.

El acelerómetro solo proporciona la aceleración experimentada en referencia a la dirección de la gravedad de la tierra, por lo tanto solo es posible detectar un cambio originado por un movimiento vertical de la cabeza. Por otro lado el giroscopio permite medir el giro respecto a un eje predeterminado, en este caso, la línea recta y vertical que une el cuello a la cabeza, brindando así el cambio causado por un movimiento horizontal de la cabeza.

Por lo que el componente ideal para esto es el MPU6050; un circuito que incorpora el acelerómetro y giroscopio en un mismo encapsulado, y adicionalmente contiene un procesador de datos que entrega valores matemáticos utilizando los valores de aceleración y velocidad angular (giro).

Para realizar clic el paciente tendrá un segundo acelerómetro (ADXL330) montado sobre su hombro y únicamente deberá moverlo hacia arriba.

Demás análisis, obtención y manipulación de datos se realiza en la microcomputadora Raspberry Pi.

1.4. Sistemas microelectromecánicos

También conocidos como MEMS por sus siglas en inglés, los sistemas microelectromecánicos son dispositivos microscópicos utilizados para interactuar o producir cambios dentro de un ambiente controlado mediante sistemas mecánicos miniaturizados que son controlados electrónicamente. Este campo tiene una amplia gama de aplicaciones comerciales ya que abarca el diseño de microsensores y microactuadores.

1.4.1. MPU-6050

El MPU6050 es uno de los primeros dispositivos para detectar o dar seguimiento a un movimiento mediante los seis ejes que posee, diseñado para tener bajo consumo de potencia, costo reducido y requisitos de alto rendimiento que demandan dispositivos como teléfonos inteligentes, tabletas u otros.

Figura 1. **Diagrama de Sistema MPU-6000/6050**

Fuente: Diagrama MPU6050. http://www.invensense.com/mems/gyro/mpu6050. Consulta: 19 de mayo de 2013.

La combinación de la tecnología MotionFusion capaz de fusionar las salidas de datos de varios sensores proveyendo funcionalidades de detección

de movimiento de mayor exactitud que otros que utilizan un solo sensor, en conjunto con el *firmware* de calibración en ejecución, permite a cualquiera integrarle con cualquier dispositivo discreto sin tener problemas complejos y sin necesidad de ser un experto en el tema.

El MPU6050 combina un acelerómetro de 3 ejes y un giroscopio de 3 ejes sobre la misma tarjeta en conjunto con el Procesador Digital de Movimiento (DMP) capaz de procesar algoritmos de MotionFusion de hasta 9 ejes.

Para obtener detección de movimientos con buena precisión es posible programar la escala de medición, modificando ciertos registros dentro del MPU, para el caso del giroscopio se tienen las siguientes escalas ±250, ±500, ±1000, y ±2000°/sec (gps) y en el acelerómetro las escalas son ±2*g*, ±4*g*, ±8*g*, y ±16*g*.

1.4.1.1. DMP

El MPU-6050 posee dentro del mismo circuito integrado un procesador digital de movimiento (DMP, por sus siglas en inglés), con el cuál es posible (extrayendo los datos provenientes del acelerómetro y giroscopio) realizar complejos algoritmos, y así, obtener nuevos datos (cuaterniones, ángulos de Euler, entre otros) para posterior procesamiento en el ordenador.

1.4.1.2. Cuaterniones

Los cuaterniones son una extensión de los números reales, parecidos a los números complejos, la diferencia principal radica en adicionar a un valor escalar tres valores imaginarios (no sólo uno) usualmente denotados por la letras *i,j*y *k,* donde $i^2=j^2=k^2=ijk=-1$. Estos brindan una notación matemática que sirve para representar las orientaciones y las rotaciones de objetos en tres

dimensiones, la simplicidad en cuanto al análisis comparado con otros métodos de obtención de orientación o rotación de un objeto los hace útiles en aplicaciones de gráficos dentro de dispositivos electrónicos ligados a la computación, robótica, entre otros.

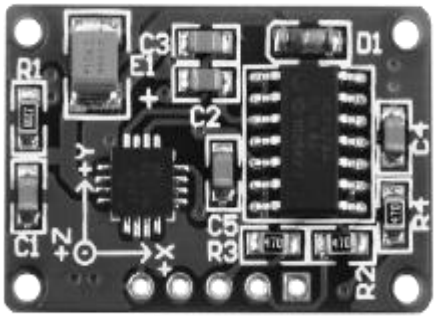
1.4.1.3. *Yaw, Pitch y Roll*

Los ángulos de *yaw, pitch y roll* conforman un conjunto de tres coordenadas angulares que representan la orientación de un objeto en un sistema de referencia de ejes ortogonales respecto a otro sistema de referencia de ejes ortogonales, *yaw* representa el ángulo que se gira sobre el eje *z, pitch* representa el ángulo que se gira sobre el eje *y* y roll representa el angulo que se gira sobre el eje *x* para un sistema de coordenadas ortogonal *xyz.*

1.4.2. ADXL330

El ADXL330 es una tarjeta pequeña, de bajo consumo de energía que contiene un acelerómetro de 3 ejes que proporciona en la salida una señal de voltaje (salida analógica) igual al vector de aceleración que se mide, todo en un mismo circuito integrado. Este mide la aceleración con un valor mínimo de escala completa de ± 3g, con un nivel de tensión de 5V. Es posible medir la aceleración estática y la aceleración dinámica y debido al bajo costo y facilidad de uso es usado en múltiples aplicaciones desde dispositivos móviles a dispositivos médicos.

Figura 2. **ADXL330**



Fuente: ADXL330 Mikroelektronika. http://www.mikroe.com/add-on-boards/measurement/three-axis-accelerometer/. Consulta: 27 de mayo de 2013.

1.4.2.1. Adquisición analógica

La adquisición analógica de datos consiste en convertir una señal física (proveniente del medio exterior) a una señal eléctrica (generada por un sensor o transductor), que a la vez se acondiciona (filtra y compara) para la conversión análogo/digital o bien para la detección de umbral cuando solo se desea conocer un cambio en el estado de la señal física, para lo cual es necesario configurar una adecuada etapa de acondicionamiento de la señal eléctrica que permita la parametrización de los datos de salida de acuerdo con los datos de entrada.

1.5. Protocolos de comunicación

Los protocolos de comunicación son las reglas, procedimientos, estándares que deben conocer y tener configurados los dos puntos (dispositivos, nodos u otros) entre los que se necesita establecer algún tipo de

comunicación para el envío de datos ya sea de forma unidireccional o bidireccional y así asegurar el envío y recepción de ambos lados.

1.5.1. UART

UART se traduce en español a Transmisor-Receptor Asíncrono Universal, es el encargado de los puertos que manejan el protocolo de comunicación serial asíncrona para el intercambio de información con otros dispositivos; la función general es la conversión de los datos de serial a paralelo (para el manejo interno) o viceversa (para la transmisión externa), así como el control de interrupciones durante la transmisión o recepción.

1.5.1.1. Características principales

- Utiliza un registro de desplazamiento para la conversión de datos que provienen de forma serial a paralelo.
- Tiene estándares de señalización por voltaje, entre los más populares se encuentranRS-232, RS-422 y RS-485.
- Usualmente permite la configuración de ciertos parámetros, como; velocidad de transmisión, *bits* de paridad, *bits* de parada y longitud de la trama.

1.5.2. I2C

El protocolo I2C o Intercircuitos Integrados es un bus de comunicaciones en serie que utiliza una velocidad de transmisión de 100 *kbit/s* en el modo estándar, el principal uso es en sistemas integrados ya que solo utiliza dos líneas para la transmisión de datos y una tercera para la referencia a tierra, estas se conocen como; SDA o línea de datos, SCL o señal de reloj y GND o

señal de referencia. Cabe resaltar que en un circuito cada dispositivo que se encuentre conectado al bus I2C tendrá una única dirección y podrá ser configurado como maestro o esclavo.

1.5.3. HDMI

La HDMI (por sus siglas en inglés) o Interfaz Multimedia de Alta Definición es un estándar utilizado en dispositivos de audio y video para la transmisión de datos o contenido digital sin necesidad de compresión, esto permite el uso de vídeo computarizado, mejorado o de alta definición, así como audio digital multicanal en un único cable. Hasta la fecha existen ya varias versiones de esta, de acuerdo a características como la tasa de transferencia, cantidad de megapíxeles que soporta en él envío y número de canales.

1.6. Raspberry Pi rev2 512 MB

La Raspberry Pi es una computadora de pequeñas dimensiones, un poco mayor a una tarjeta de crédito, que puede conectarse a un teclado y una pantalla externas y realizar tareas como las que normalmente hacemos en una computadora común como el manejo de hojas de cálculo o de datos, juegos e inclusive ver videos de alta definición. Sin embargo aunque se pueden realizar este tipo de tareas, la Raspberry Pi no es una computadora comercial como tal, puede verse a esta como un dispositivo que brinda la oportunidad de programarle y de utilizar abiertamente los puertos y funciones que posee.

1.6.1. Componentes principales

La Raspberry Pi dispone de diferentes componentes tales como;

- Procesador. Basado en una arquitectura ARM11, este es un chip de 32 *bits* y frecuencia de 700 MHz. El modelo B utilizado en este proyecto tiene 512 MB de RAM.

- Tarjeta SD (Secure Digital). La tarjeta SD hace la función de disco duro en la Raspberry Pi, ya que este como tal no trae uno, es recomendado utilizar una de 4 GB o más.

- Puertos USB. En el modelo utilizado se encuentran dos puertos USB 2.0

- Puerto Ethernet. En el modelo utilizado se encuentra un puerto RJ45 Ethernet estándar, que básicamente es un adaptador USB a Ethernet.

- Conector HDMI.El puerto HDMI provee salidas video y audio digital. Soporta hasta 14 diferentes tipos de resolución de video, la señal puede ser convertida a DVI, señal análoga compuesta y SCART utilizando adaptadores externos.

- LEDs indicadores. Estos LEDs proveen retroalimentación visual de ciertos procesos.

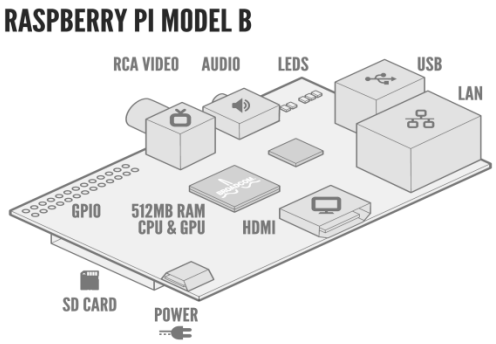
- Salida de audio análogo. Contiene un conector de audio de 3.5 mm para poder manejar cargas de alta impedancia.

- Salida de video compuesto. Dispone de un conector RCA capaz de proveer señales de video compuesto NTSC o PAL, está es de más baja resolución que HDMI.

- Entrada de energía. Para proveer de energía a la Raspberry Pi se dispone de un conector microUSB.

- Pines de propósito general. Estos pines permiten la conexión de dispositivos externos, para comunicación, energía, mediciones u otros.

Figura 3. **Raspberry Pi Model B**



Fuente: Raspberry Pi. http://www.raspberrypi.org/wp-content/uploads/2011/07/RaspiModelB.png. Consulta: 27 de mayo de 2013.

Tabla I. **Especificaciones técnicas**

	Modelo A	Modelo B
Precio:	$25	$35
SoC:	Broadcom BCM2835 (CPU + GPU + DSP + SDRAM + puerto USB)	
CPU:	ARM1176JZF-S a 700 MHz (familia ARM11)	
GPU:	Broadcom VideoCore IV, OpenGL ES 2.0, -2 y VC-1 (con licencia),1080p30 H.264/MPEG-4 AVC	
Memoria (SDRAM):	256 MB (compartidos con la GPU)	512 MB (compartidos con la GPU)
Puertos USB 2.0:	1	2 (vía hub USB integrado)
Entradas de vídeo:60	Conector [[MIPI] CSI que permite instalar un módulo de camara desarrollado por la RPF	
Salidas de vídeo:	Conector RCA (PAL y NTSC), HDMI (rev1.3 y 1.4), Interfaz DSI para panel LCD	
Salidas de audio:	Conector de 3.5 mm, HDMI	
Almacenamiento integrado:	SD / MMC / ranura para SDIO	
Conectividad de red:	Ninguna	10/100 Ethernet (RJ-45) via hub USB
Periféricos de bajo nivel:	8 x GPIO, SPI, I²C, UART	
Reloj en tiempo real:	Ninguno	
Consumo energético:	500 mA, (2.5 W)	700 mA, (3.5 W)
Fuente de alimentación:	5 V vía Micro USB o GPIO header	
Dimensiones:	85.60mm × 53.98mm64 (3.370 × 2.125 inch)	
Sistemas operativos soportados:	Debian, Fedora, Arch Linux,Slackware Linux, RISC OS	

Fuente: Raspberry Pi. http://es.wikipedia.org/wiki/Raspberry_Pi.
Consulta: 27 de mayo de 2013.

1.6.2. Linux embedido

Linux es un sistema operativo creado por Linus Torvalds como una alternativa a sistemas como Windows, Mac OS, MS-DOS, etc. En términos generales este es una interfaz entre el hardware de un computador/servidor y los programas que funcionan en este.

Un importante detalle de Linux es la historia en el desarrollo, aunque el inicio de este se debe a Linus Torvalds, este abrió las puertas para que programadores en todo el mundo se unieran y dieran sus comentarios, opiniones, e inclusive lo modificarán para ser utilizado de manera más eficiente

de acuerdo con la utilización que quería dársele, a esto se debe que Linux sea considerado como un sistema libre.

Hoy en día Linux es un sistema operativo muy utilizado en servidores de grandes redes debido a su bajo nivel de fallas y que no requiere reiniciarlo constantemente como es el caso con otros similares. También se ha vuelto muy amigable debido a las diferentes distribuciones como Ubuntu, Linspire, Xandros, Debian, entre otras.

1.6.2.1. Debian

El proyecto Debian fue iniciado en 1993 por Ian Murdock y su esposa Debra, este tiene como entidad que lo respalda, a la organización sin ánimo de lucro Software en el Interés Público. Debian es una versión bastante versátil y es la base de conocidas distribuciones como Ubuntu o Xandros.

Debian es caracterizado por:

- Una distribución sólida y de calidad con una larga historia dentro de Linux.
- Ha sido la base de fundación de muchas otras distribuciones de calidad de Linux.
- Es no comercial y no lucrativo, esto beneficia a los usuarios que no disponen de recurso monetario.
- Múltiples aplicaciones en cuanto al tipo de uso o usuarios, desde un aprendiz hasta profesionales en el tema.

- Fácil de obtener y actualizar. Existe una enorme cantidad de voluntarios creando nuevas versiones actualizadas y mejoradas.

- Tiene más de 29 000 paquetes y corre sobre 9 arquitecturas

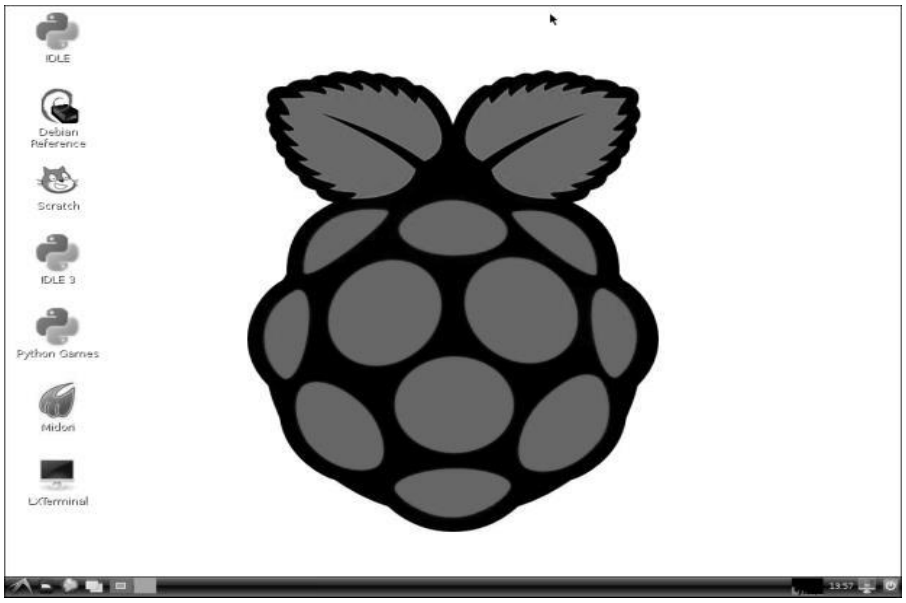
1.6.2.1.1. Raspbian

Es un sistema operativo libre basado en una optimización de Debian para ser utilizado en la Raspberry Pi. Un sistema operativo es el *set* básico de programas y utilidades que hacen funcionar a la Raspberry Pi, sin embargo Raspbian es más que un SO, contiene más de 35 000 paquetes y software precompilado de fácil instalación.

La versión de Raspbian que se instala inicialmente sobre la Raspberry Pi trae un pequeño grupo de programas, por lo que es necesario descargar e instalar los programas que se requieren de acuerdo a la utilización.

Esta trae el LXDE (Lightweight X11 Desktop Environment) como el entorno gráfico de escritorio predeterminado, este es un entorno recortado-desplegable de los sistemas X Window que se ha utilizado en las interfaces gráficas de usuario en Unix y Linux, en general este permite manejar las vistas y da la sensación del manejo de ventanas y menús para controlarle.

Figura 4. **Escritorio**



Fuente: How to get started with the Raspberry Pi. http://reviews.cnet.co.uk/desktops/how-to-get-started-with-the-raspberry-pi-50009845/. Consulta: 27 de mayo de 2013.

Entre las herramientas de manejo dentro de este entorno se pueden encontrar:

- Buscador Web. Midori es el buscador web que por defecto se encuentra en Raspbian, la ventaja de este sobre otros más populares se debe al bajo uso de recursos provocando una disminución en procesamiento para ser aprovechado por otras tareas que se ejecutan en paralelo.

- Audio y video. Omxplayer es el reproductor de audio y video para Raspbian y está diseñado para trabajar con la Unidad de Procesamiento de Gráficos.

- Editores de texto. Leafpad es el editor de texto por defecto de Raspbian y se puede encontrar en el menú principal.

- Consola. Se utiliza para ejecutar comandos de línea de modo que se pueda realizar alguna tarea que no puede ejecutarse desde el modo gráfico.

1.6.3. Lenguajes de programación

Los lenguajes de programación son el idioma utilizado para controlar la ejecución de tareas o procesos de un equipo (computadora, teléfono, etc.) que posea la tecnología adecuada para la interpretación del mismo. Usualmente el ser humano escribe programas o algoritmos para indicar al equipo qué funciones realizar; existe una diversidad de tipo de lenguajes y diferentes clasificaciones de acuerdo a la funcionalidad o fin.

1.6.3.1. C++

C++ es un lenguaje de programación creado en 1983 por Bjarne Stroustrup, como una versión mejorada del lenguaje de programación C. Este es orientado a objetos y es considerado un lenguaje de alto nivel, haciéndolo muy popular.

Los inicios datan desde 1979 cuando recibía el nombre de C con Clases y fue utilizado internamente por compañías como AT & T, pero no fue lanzado comercialmente sino hasta 1985.

Entre las características más notables se pueden encontrar; la utilización de clases, funciones virtuales, formatos predefinidos, manejo de operadores cuando se da una sobrecarga y chequeo de escritura.

Este lenguaje fue creado para Unix por lo que permite a los programadores modificarlo sin cambiarlo, el código es reutilizable y es posible la creación de librerías de forma más ordenada. Es considerado como portable ya que no requiere de un tipo especial de equipo o sistema operativo.

El compilador utilizado para C++ es g++ este pertenece a la colección de compiladores de GNU C++ cuyo desarrollo proviene del proyecto GNU pensado para desarrollar software libre.

1.6.3.2. Python 2.7

Python es un lenguaje de programación interpretado, orientado a objetos y de alto nivel, basado en una sintaxis muy limpia que permita crear código ordenado y fácil de entender. Se trata de un lenguaje multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional.

Este fue diseñado para ser leído con facilidad, una de las características que posee es el uso de palabras donde otros lenguajes utilizarían símbolos, el contenido de los bloques de código (bucles, funciones, clases, etc.) es delimitado mediante espacios o tabuladores, conocidos como indentación, antes de cada línea de órdenes pertenecientes al bloque.

Python se diferencia así de otros lenguajes de programación que mantienen como costumbre declarar los bloques mediante un conjunto de caracteres, normalmente entre llaves.

Tiene una gran biblioteca estándar, usada para una diversidad de tareas. Esto viene de la filosofía pilas incluidas en referencia a los módulos de Python. Los módulos de la biblioteca estándar pueden mejorarse por módulos personalizados escritos tanto en C como en Python.

Debido a la gran variedad de herramientas incluidas en la biblioteca estándar, combinada con la habilidad de usar lenguajes de bajo nivel como C y C++, los cuales son capaces de interactuar con otras bibliotecas, Python es un lenguaje que combina la clara sintaxis que posee con el inmenso poder de lenguajes menos elegantes.

2. CONFIGURACIONES INICIALES EN RASPBERRY PI

La Raspberry Pi es una microcomputadora con la capacidad de realizar tareas como las de una computadora comercial, sin embargo, existe una serie de configuraciones que deben realizarse previo al uso.

2.1. Configuración de Raspbian en memoria SD

La memoria SD hace la función de disco duro en la Raspberry Pi, en esta se almacenarán todos los archivos de datos y se creará la imagen del sistema operativo Raspbian. Este software puede obtenerse desde la página web de la Raspberry Pi, en la sección de descargas. Existen varias versiones que pueden utilizarse, de acuerdo a las funcionalidades ofrecidas; en este caso se utiliza Raspbian wheezy.

Para dejar un procedimiento más claro acerca de la configuración de Raspbian wheezy en la memoria SD, se detalla a continuación una serie de pasos:

- Descarga de archivo zip con versión de Raspbian wheezy desde la página web de Raspberry Pi.

- Descomprimir el archivo .img, o imagen del sistema operativo

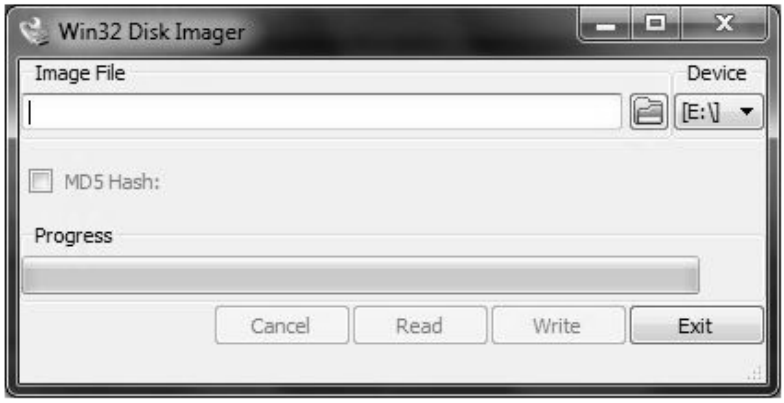
- Aunque no es necesario, se puede comprobar la integridad del archivo .img por medio de un programa que verifique funciones hash

criptográficas MD5, utilizando el archivo digerido que se encuentra en la misma página.

- Descargar Win32DiskImager, este programa servirá para transferir los datos de la imagen comprimida (archivo .img) hacia las particiones que se crearán para albergar el sistema operativo en la unidad de inicio de estado sólido.

- Serán generadas tres particiones al realizar el paso anterior:
  - FAT32: esta contiene la configuración de arranque del sistema operativo, accesible desde Windows.
  - ext3: es la partición principal en donde se almacena y ejecuta el sistema operativo propio de la Raspberry Pi (Raspbian).
  - SWAP: en esta se encuentra la memoria dinámica auxiliar de intercambio, la cual es utilizada como RAM cuando esta última llega al límite de capacidad.

Figura 5. **Captura de Win32DiskImager**



Fuente: Win32DiskImager. http://sobrebits.com/montar-un-servidor-casero-con-raspberry-pi-parte-1-instalar-raspbian-en-una-tarjeta-sd/. Consulta: 03 de junio de 2013.

- Al finalizar con los pasos anteriores se debe insertar la memoria SD en la Raspberry Pi para proceder al arranque inicial de la misma.

2.2. Configuración de arranque inicial

La configuración de arranque inicial se subdivide entre el archivo de configuración inicial config.txt y los parámetros que se establecen con la herramienta *raspi-config* al encender la Raspberry Pi.

2.2.1. Archivo de configuración inicial

En este fichero (config.txt) se encuentra la configuración que la Raspberry Pi tomará por defecto al iniciar el sistema operativo, en el cual se definen parámetros tales como: resolución de la pantalla (*pixeles*), frecuencia del procesador (por defecto 700 MHz), frecuencia del núcleo, frecuencia de la RAM, entre otros.

Los parámetros que se modifican en el archivo original son los siguientes:

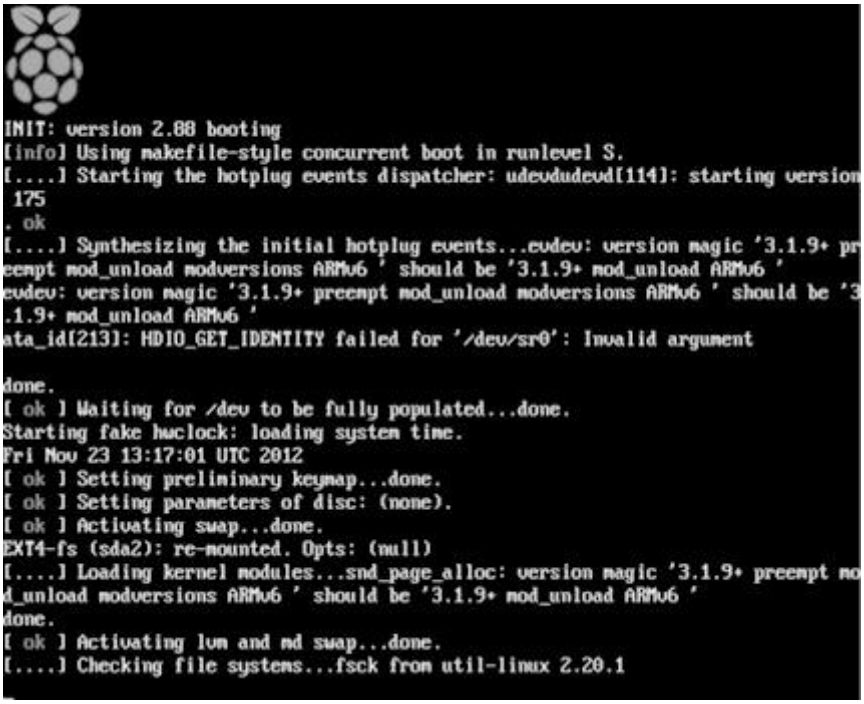
- framebuffer_width=1360: se establece en función de la resolución horizontal de la pantalla.
- framebuffer_height=720: se establece en función de la resolución vertical de la pantalla.
- hdmi_force_hotplug=1: forza la salida de video al puerto HDMI
- hdmi_drive=2: forza la salida de audio por el puerto HDMI
- config_hdmi_boost=2: aumenta la amplitud de la señal de salida del HDMI.

Este archivo debe modificarse desde otro ordenador antes de encender la Raspberry Pi para asegurarse de que la salida de video se hará hacia la pantalla con HDMI. Si se utiliza una pantalla con salida de video RCA no será necesario modificar los parámetros de HDMI.

2.2.2. Configuración de parámetros principales

Al encender por primera vez la Raspberry Pi se observará en la pantalla el arranque de varios procesos como la inicialización de la interfaz de red, reconocimiento de los periféricos USB, entre otros.

Figura 6. **Procesos de arranque en Raspberry Pi**



Fuente: Arranque Raspberry Pi. http://reviews.cnet.co.uk/desktops/how-to-get-started-with-the-raspberry-pi-50009845/. Consulta: 06 de junio de 2013.

Al finalizar estos procesos, se desplegará en pantalla la herramienta *raspi-config tool* para el establecimiento de la configuración de la Raspberry Pi.

Figura 7. **Herramienta Raspi-config**

Fuente: primera ejecución de Raspbian. http://sobrebits.com/montar-un-servidor-casero-con-raspberry-pi-parte-2-primera-ejecucion-de-raspbian/. Consulta: 06 de junio de 2013.

A continuación se describe cada uno de estos parámetros:

- Info: brinda información acerca de la herramienta Raspi-config

- Expand_rootfs: por defecto si no se cambia, expande la partición en la memoria SD hasta abarcar toda la capacidad de esta última,

considerando que la memoria es mayor a 2 GB y se utiliza únicamente para la Raspberry Pi.

- Overscan: esta opción sirve para habilitar o deshabilitar el sobreescaneo, en caso de que la salida de video no despliegue la imagen entera en la pantalla que se está utilizando.

- Configure_keyboard: reconfigura el teclado (# dpkg-reconfigure keyboard-configuration) y recarga la configuración (# invoke-rc.d keyboard-setup start).

- Change_pass: cambia la contraseña para el usuario pi ($ passwd pi)

- Change_locale: reconfigura la codificación de carácteres (# dpkg-reconfigure locales).

- Change_timezone: reconfigura la zona horaria del sistema (# dpkg-reconfigure tzdata).

- Memory_split: configuración de la cantidad de RAM que se asigna a la unidad de procesamiento gráfico o GPU por sus siglas en inglés y el resto se asigna al procesador.

- Overclock: si se desea permite subir la frecuencia de operación del procesador para hacerlo más rápido, pero puede causar inestabilidad en el sistema y sobrecalentar la tarjeta.

- SSH: habilita o deshabilita el servicio SSH en la Raspberry Pi

- Boot_behaviour: sirve para habilitar o deshabilitar la ejecución por defecto del entorno gráfico al encender la raspberry pi.

- Update: esta opción actualizará los repositorios (# apt-get update) e intentará actualizar la herramienta raspi-config (# apt-get install raspi-config).

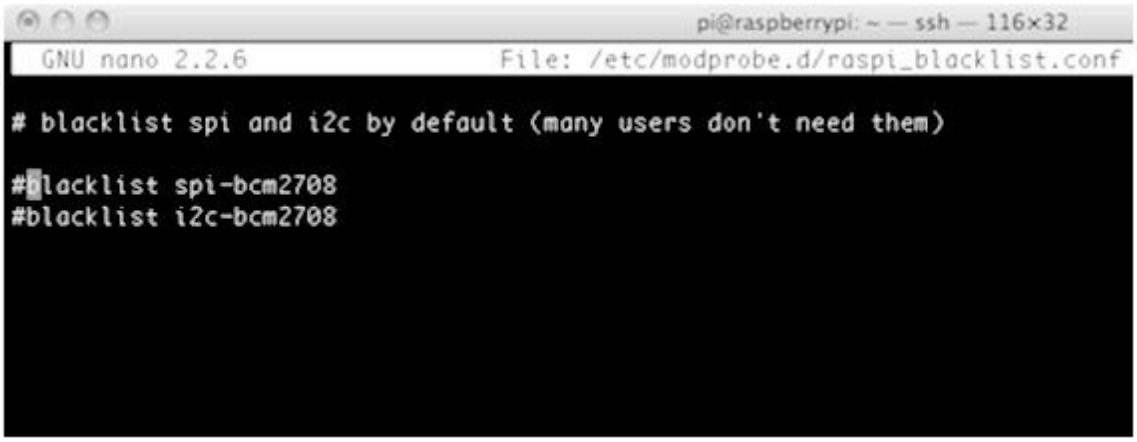
Al finalizar los cambios se debe seleccionar finalizar y esto mostrará la línea de comando, entonces debe escribirse en la terminal *sudo reboot* para reiniciar la Raspberry Pi para completar la configuración.

2.3. Configuración *driver* I2C

El *driver* I2C es utilizado en la Raspberry Pi para manejar la comunicación con el sensor externo. Este ya viene instalado por defecto dentro de Raspbian, pero es necesario habilitarlo realizando una serie de pasos que a continuación se detallan.

- El fichero /etc/modprobe.d/raspi-blacklist.conf contiene el listado de dispositivos que serán descartados durante el arranque del sistema operativo, por lo que es necesario comentar (utilizando el signo # previo al dispositivo) las líneas correspondientes al bus I2C.

Figura 8. **Configuración de dispositivos descartados al inicio**



Fuente: Configuración I2C. http://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-setup/configuring-i2c. Consulta: 03 de junio de 2013.

- Seguido, debe editarse el archivo que contiene los módulos que se cargan al arranque: */etc/modules*. Agregar el dispositivo I2C, escribiendo lo siguiente: *i2c-dev* e *i2c-bcm2708*.

Figura 9. **Configuración de fichero de módulos**

Fuente: Configuración I2C. http://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-setup/configuring-i2c. Consulta: 03 de junio de 2013.

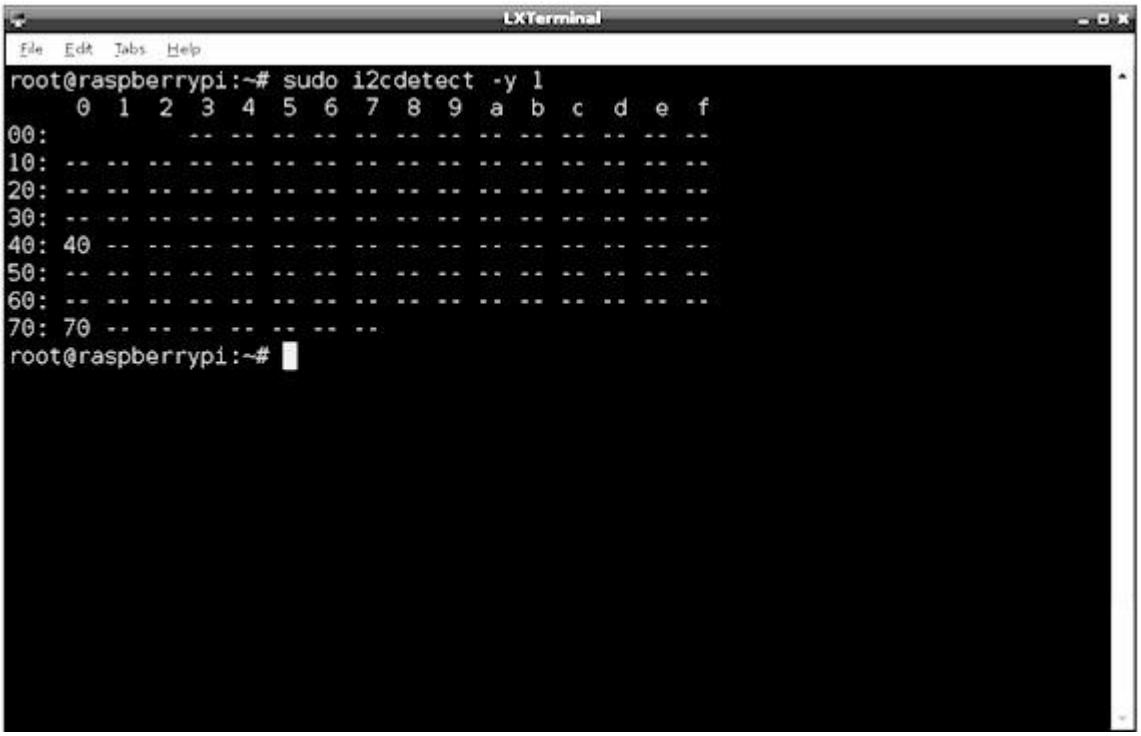
- Instalar el paquete de herramientas de I2C por medio del comando: *sudo apt-get install i2c-tools*.

- Agregar al usuario pi los permisos para administrar el grupo I2C mediante el comando: *sudo adduser pi i2c*.

- Para completar la instalación debe reiniciarse el sistema

- Si se desea detectar los dispositivos I2C conectados a la Raspberry Pi puede hacerse uso del comando: *sudo i2cdetect –y 1*.

Figura 10. **Detección de dispositivos I2C conectados**



Fuente: Configuración I2C. http://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-setup/configuring-i2c. Consulta: 03 de junio de 2013.

2.4. Instalación de paquetes

Los paquetes son componentes adicionales a la instalación básica de Python que viene instalada por defecto en la Raspberry Pi. Están diseñados para realizar tareas específicas y deben ser seleccionados cuidadosamente en función de la aplicación en la que se deseen utilizar.

2.4.1. PySerial

Este paquete contiene los métodos y propiedades que permiten el acceso directo al hardware de comunicación serial a través del UART de la Raspberry Pi. Asimismo, proporciona el soporte para Python corriendo sobre Windows, Linux, BSD y otros. Posee las siguientes características:

- Utiliza una misma clase global en todos los sistemas operativos
- Acceso a la configuración del puerto a través de propiedades de Python
- Soporta distintos tamaños de *bytes* de datos, *bits* de parada y paridad, y control de flujo a través de software y hardware.
- Es capaz de funcionar con o sin límite de tiempo de espera
- Acceso sencillo a escritura o lectura del puerto
- Los archivos en este paquete son 100 % código Python

Para la instalación del paquete se debe ejecutar el siguiente código en la línea de comandos:

- *sudo apt-get install python-serial*

2.4.2. PyMouse

El paquete PyMouse proporciona el medio para controlar el mouse del ordenador a través del manejo de métodos ya definidos.

Para la instalación de este, debido a que se está utilizando Raspbian como sistema operativo y este ya trae por defecto Python, no es necesario

descargarlo, entonces es posible descargar e instalar la librería utilizando las siguientes líneas de comando:

- *sudo apt-get install python-pip*
- *sudo pip install pymouse*

Con la primera línea de comando se instala la herramienta pip: esta es una herramienta utilizada para instalar y manejar paquetes de Python. Asimismo, con la segunda línea de comando se instala el paquete PyMouse.

En Pymouse es necesario instanciar el mouse como objeto, utilizando (por ejemplo) el siguiente código:

- *mouse = PyMouse()*

Es importante que siempre se importe o llame al paquete cuando se inicia el código de programa en python, escribiendo antes de instanciar el objeto, la línea:

- *from pymouse import* Pymouse

Pymouse dispone de una serie de métodos para manejar funciones del mouse entre las que cabe resaltar las siguientes:

- *move(x,y)*: coloca el cursor en las posiciones absolutas *x* y *y*. En línea de código se escribiría: mouse.move(x,y).

- *click(x,y,int button)*: realiza el clic en las posiciones absolutas *x* y *y*, *int button* define qué tipo de clic se dio; 1 es clic izquierdo, 2 clic derecho y 3

clic en el centro. En línea de código se escribiría: *mouse.click(x,y,int button).*

- *screen_size()*: devuelve el valor de resolución horizontal y resolución vertical de la pantalla. En línea de código se escribiría: *mouse.screen_size().*

- *position()*: devuelve la posición *x* y *y* del *mouse*. En línea de código se escribiría: *mouse.position().*

2.4.3. eSpeak

Es un sintetizador de voz de código abierto para distintos idiomas, incluyendo inglés y español. Además, puede ser utilizado en Windows y Linux. Este utiliza un método de síntesis de formación, lo que permite que varios lenguajes se provean en tamaño pequeño. El habla es clara y puede utilizarse a velocidades altas, sin embargo no es tan clara como en otros sintetizadores y algunos fonemas podrán sonar mejor para ciertos idiomas y un poco distorsionado o fuera de tono para otros. Sin embargo eSpeak representa una ventaja en cuanto a personalizaciones sobre otros sintetizadores más pesados que utilizan grabaciones predeterminadas de voz humana.

Para la instalación y configuración de eSpeak en Raspbian se deben ejecutar los siguientes pasos:

- Instalar eSpeak utilizando el comando: *sudo apt-get install espeak*

- eSpeak cuenta con archivos de voz de acuerdo al idioma dentro de uno de sus directorios para hacer referencia a ellos debe agregarse –v

seguido del nombre del idioma, usualmente escrito en dos letras de acuerdo con la Norma ISO que se esté utilizando, para el caso del español sería –ves.

- Existe una serie de variantes de voz ya predefinidos en eSpeak que pueden utilizarse; tales como: +m1, +m2, +m3, +m4, +m5, +m6 para voces masculinas y +f1, +f2, +f3, +f4, +f5, +f6 para voces femeninas, -k cuando se quiere acentuar las letras mayúsculas, -s para que el habla sea lenta, entre otros más.

- De modo que, si se quiere reproducir un texto en español con voz femenina que acentúe las letras mayúsculas de forma lenta puede utilizarse la línea de comando: *espeak –ves+f3 –k5 –s150 texto*.

A continuación se describen los parámetros u opciones modificables:

- -a: define la amplitud como un entero en el rango de 0 a 200, el valor por defecto es 100.
- -b: define al texto de entrada con un tamaño de 8 *bits* codificado
- -f: utiliza el texto de entrada desde un archivo de texto
- -k: indica como pronunciar las letras mayúsculas de acuerdo al número entero que se le adiciona.
- -s: define la velocidad del habla de acuerdo con la cantidad de palabras por minuto, este puede estar en el rango de 80 a 370.

3. ADQUISICIÓN DE DATOS DESDE DISPOSITIVO DE RASTREO DE POSICIÓN

La adquisición de los datos que proporciona el MPU6050 se hizo por medio de programación orientada a objetos en el lenguaje C++. Los programas que se utilizan resultan de algunas modificaciones realizadas al código que fue desarrollado en principio por Jeff Rowberg, con la contribución de Richard Steele. Es importante mencionar que este es de uso libre, por lo que fue posible la modificación y utilización sin quebranto de ley a conveniencia de este proyecto.

3.1. Tipos de archivos en programación modular

Al hablar de programación modular es necesario hacer referencia a los tipos de archivos que se utilizan. En C++ se permite el uso de dos tipos de archivos: de descripción de interfaces y de implementación.

3.1.1. Archivos de descripción de interface

También llamados archivos de encabezado o archivos include (archivos.h). Contienen las declaraciones de constantes, variables y funciones del módulo al que pertenecen, también pueden tener llamados hacia otros archivos de encabezado si fuese necesario.

3.1.2. Archivos de implementación

Aquí se escribe el código para las funciones que se han declarado en el archivo de descripción de interface del mismo módulo (archivos.cpp), de modo que, en el proceso de compilación de los archivos de implementación se les añade el archivo de descripción de interface, creando un archivo objeto, que se adicionará a otros del mismo tipo, para formar por último un único archivo ejecutable.

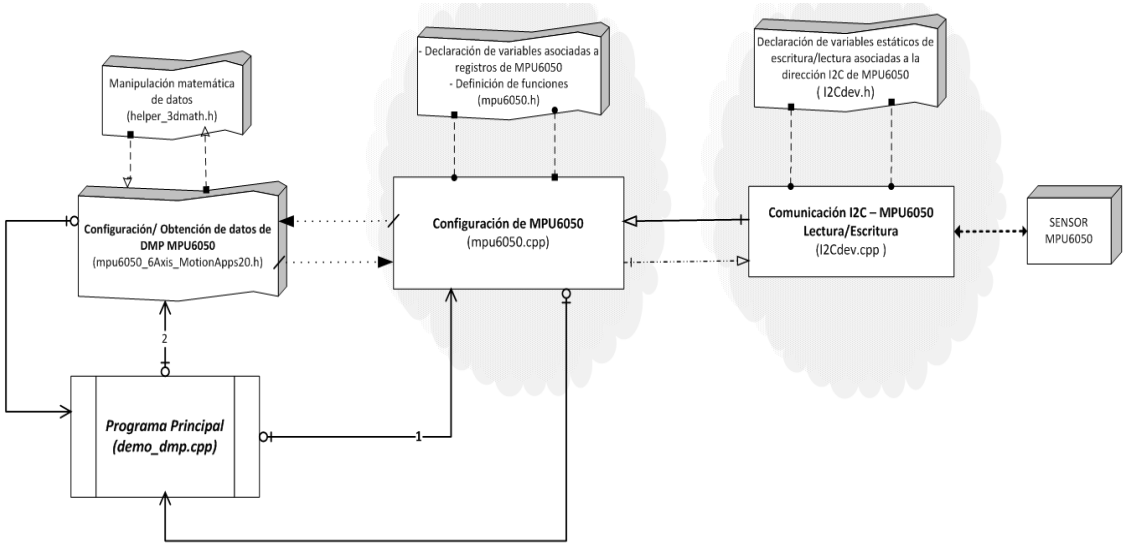
3.2. Algoritmo de adquisición

En la resolución de un problema demasiado complejo a través de programación por software, es necesario obtener un algoritmo para subdividir las tareas a ejecutarse dentro de varios subprogramas. De este modo, es mucho más fácil el ordenamiento y se evitan repeticiones de código innecesarias.

En la adquisición de datos del MPU6050 se utiliza una serie de programas para realizar diferentes tareas. Estos son:

- Programa Principal (demo_dmp.cpp)
- Configuración MPU6050 (mpu6050.cpp)
- Archivo de encabezado definición de registros y funciones (MPU6050.h)
- Configuración/obtención de datos DMP MPU6050 (mpu6050_6Axis_MotionApps20.h).
- Comunicación I2C (i2cdev.cpp)
- Archivo de encabezado (i2cdev.h)
- Manipulación matemática (helper_3dmath.h)

Figura 11. **Diagrama de flujo de módulos**



Fuente: elaboración propia, con programa Microsoft Visio 2010.

En el diagrama de arriba se muestra el flujo de comunicación que se da para la obtención de datos desde el sensor. A continuación se describe, uno a uno cada subprograma.

3.2.1. Demo_dmp.cpp

El programa demo_dmp.cpp es un archivo de implementación que llama a las funciones principales de configuración o verificación de estado del MPU6050 que han sido definidas en los otros archivos. Asimismo, en este se encuentra la configuración para el envío de datos a través de UART sobre un GPIO de la Raspberry Pi.

3.2.1.1. Inclusión de archivos de cabecera

Al inicio del programa se incluyen (#include) los siguientes archivos de cabecera:

- #include <stdio.h>: cabecera estándar de entrada y salida. Definiciones de macros, constantes, declaraciones de funciones.

- #include <stdlib.h>: biblioteca estándar de propósito general con prototipos de funciones de C++ para gestión de memoria dinámica y control de procesos.

- #include <unistd.h>: provee acceso al sistema POSIX (interfaz de SO portable) que define la API.

- #include <stdint.h>: sirve para declarar o predefinir tamaños de enteros, así como macros.

- #include <string.h>: contiene la definición de macros, constantes, funciones y tipos de utilidad para trabajar con cadenas de caracteres y algunas operaciones de manipulación de memoria.

- #include <math.h>: archivo de cabecera que contiene las operaciones matemáticas básicas.

- #include <fcntl.h>: se utiliza para definir los argumentos de las funciones fcntl() y open().

- #include <errno.h>: se definen los macros que presentan un informe de error a través de códigos de error.

- #include <termios.h>: contiene las definiciones para las interfaces de las terminales I/O.

- #include <sys/types.h>: contiene una colección de símbolo typedef y estructuras.

- #include <sys/stat.h>: devuelve la data o información de la funcion stat()

- #include "I2Cdev.h": librería creada para lectura/escritura por medio de I2C.

- #include "MPU6050_6Axis_MotionApps20.h": librería creada para la obtención de datos y estado del DMP MPU6050.

En el caso de las librerías I2Cdev.h y MPU6050_6Axis_MotionApps20.h se utiliza comillas, por que estas no son librerías estándar. Por defecto cuando se compila el programa, son buscadas dentro del directorio fuente donde se encuentra el archivo principal.

3.2.1.2. Declaración de variables

Se utiliza una serie de variables para diferentes objetivos, tales como:

- Variables booleanas para indicar si un procedimiento fue exitoso o no
  - bool dmpReady = false;
- Variables enteras sin signo de 8 *bits* y 16 *bits* de longitud

- uint8_t mpuIntStatus;
- uint8_t devStatus;
- uint16_t packetSize;
- uint16_t fifoCount;
- uint8_t fifoBuffer[64]

- Variable de la clase Quaternion
  - Quaternion q;
- Variables tipo float
  - float ypr[3];

3.2.1.3. Configuración inicial

La configuración inicial es una función de tipo *void*, esta se subdivide en el llamado a diferentes funciones dentro de los subprogramas que permiten realizar configuraciones en el sensor y determinar si un procedimiento se realizó con éxito. La sintaxis utilizada es mpu.función(), donde mpu es el objeto. A continuación se brinda mayor detalle:

- mpu.initialize(): define el reloj, sensibilidad de giroscopio (250º/s), escala de acelerómetro (+/- 2g) y deshabilita la función de dormir modificando el registro uno de manejo de energía del mpu6050.

- mpu.testConnection(): lee el registro 0x75 *bit* 6 al 1 que corresponde a la dirección i2c del MPU6050.

- mpu.dmpInitialize(): esta retorna un valor igual a cero si la carga de datos y configuración desde y hacia los registros del DMP y FIFO se realiza con éxito; de lo contrario, el resultado será un error.

- Si el valor de retorno en la función anterior es cero, se llama a las siguientes funciones:
    - mpu.setDMPEnabled(true);
    - mpu.getIntStatus();

Luego es modificada la variable booleana de estado de configuración del MPU, dmpReady con valor igual a cero, para permitir que se ejecute la función de *bucle* principal del programa.

3.2.1.4. Bucle principal del programa

El bucle principal del programa es una función de tipo *void* que verifica el estado de la FIFO, conteo de paquetes, borrado de paquetes y lectura de paquetes, que al mismo tiempo se cargan a las funciones que procesan los datos y entregan los valores de ángulos.

- Obtención de conteo de la FIFO
    - fifoCount = mpu.getFIFOCount();

- Reseteo de la FIFO si la variable fifoCount es 1024
    - mpu.resetFIFO();

- Lectura de paquetes si fifoCount es mayor o igual a 42
    - mpu.getFIFOBytes(fifoBuffer, packetSize);

Los valores obtenidos son cargados en las funciones que realizan los cálculos matemáticos (se encuentran en otros subprogramas). Para este proyecto solo interesa la obtención de los ángulos *yaw* y *roll*, por lo que solo se describe abajo las modificaciones al código base para la respectiva obtención.

- Obtención de los cuaterniones, gravedad y ángulos *yaw*, *pitch* y *roll*
    - mpu.dmpGetQuaternion(&q, fifoBuffer);
    - mpu.dmpGetGravity(&gravity, &q);
    - mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);

- Para el envío de datos a través del UART se utiliza una variable (strOut) a la que se carga el valor de *yaw* y *roll*, se obtiene su tamaño para ser enviado a través de la función estándar write, cuyos campos son: descriptor de archivo o salida (variable fd igual al puerto serial de la Raspberry Pi), variable a enviar (variable strOut), tamaño de variable (variable len).
    - sprintf(strOut, "%7.2f,%7.2f", ypr[0] * 180/M_PI, ypr[2] * 180/M_PI);
    - strcat(strOut, "\n");
    - int len = sizeof(strOut);
    - write(fd, strOut, len);

3.2.1.5. Función principal

La función principal int main() dentro del programa llama a las funciones *void* que se han descrito anteriormente y adicionalmente contiene la configuración del puerto serial.

- Llamado a la función de configuración inicial setup()

- Definición de puerto serial sobre variable fd
    - fd = open("/dev/ttyAMA0", O_WRONLY | O_NOCTTY | O_NDELAY | O_NONBLOCK);

- Configuración de la interfaz que controla el puerto de comunicación asíncrona. Se utiliza la estructura termios y se definen los valores de los campos que le componen.

- Bucle infinito (for (;;)) para llamar a la función loop() y realizar el envío uno a uno de los datos matemáticos por el puerto serial.

3.2.2. MPU6050.cpp

El archivo de implementación mpu6050 contiene una serie de funciones que realizan tareas de escritura o lectura en los registros del MPU6050. Estas pueden retornar un valor o no, de acuerdo al tipo de registro que se manipula.

3.2.2.1. Inclusión de archivos de cabecera

Se incluyen algunos archivos de cabecera estándar y el archivo MPU6050.h.

- #include <stdio.h>
- #include <stdlib.h>
- #include <unistd.h>
- #include <string.h>
- #include <stdint.h>
- #include "MPU6050.h"

3.2.2.2. Funciones de lectura y escritura registros de MPU6050

Las funciones de lectura y escritura, utilizan siempre un llamado a las funciones de lectura o escritura ya predefinidas en el subprograma I2Cdev.cpp. A continuación se muestra un listado con la descripción de las funciones de interés:

- Definición de dirección I2C de MPU6050. Se utiliza la dirección por defecto del bus.

- initialize(). Se utiliza para definir el reloj, la escala del giroscopio a rango completo (250 grados por segundo), escala del acelerómetro a +/- 2g, (donde g es el valor de la gravedad) y despierta el MPU6050. Esta función es utilizada en el programa principal para realizar la configuración inicial del MPU6050.

- Lectura y escritura sobre el registro divisor de muestreo (SMPLRT_DIV). Esto sirve para establecer la rata de muestreo.
    - Velocidad de muestreo = velocidad de salida de giroscopio / (1 + SMPLRT_DIV).

- Registro CONFIG: en este se configura la sincronización de trama externa del pin donde se realiza el muestreo (*bit* 3 al *bit* 5) y los parámetros para definir el filtro digital paso bajo que se utiliza en el acelerómetro y giroscopio.

Tabla II. **Configuración de *bits* para filtro paso bajo**

DLPF_CFG	Accelerometer (F_s = 1kHz)		Gyroscope		
	Bandwidth (Hz)	Delay (ms)	Bandwidth (Hz)	Delay (ms)	Fs (kHz)
0	260	0	256	0.98	8
1	184	2.0	188	1.9	1
2	94	3.0	98	2.8	1
3	44	4.9	42	4.8	1
4	21	8.5	20	8.3	1
5	10	13.8	10	13.4	1
6	5	19.0	5	18.6	1
7	RESERVED		RESERVED		8

Fuente: Invensense Inc. MPU-6000 and MPU-6050 Register Map and Descriptions. http://invensense.com/mems/gyro/documents/RM-MPU-6000A.pdf. Consulta: 25 de junio de 2013.

- Configuración y obtención de los rangos de escala completa que maneja el giroscopio.

Tabla III. **Rangos de escala completa de giroscopio**

FS_SEL	Full Scale Range
0	± 250 °/s
1	± 500 °/s
2	± 1000 °/s
3	± 2000 °/s

Fuente: *Invensense Inc.* MPU-6000 and MPU-6050 Register Map and Descriptions. http://invensense.com/mems/gyro/documents/RM-MPU-6000A.pdf. Consulta: 25 de junio de 2013.

- Realización de prueba automática en acelerómetro y la configuración de rango de escala completa.

Tabla IV. **Rangos de escala completa de acelerómetro**

AFS_SEL	Full Scale Range
0	± 2*g*
1	± 4*g*
2	± 8*g*
3	± 16*g*

Fuente: Invensense Inc. MPU-6000 and MPU-6050 Register Map and Descriptions. [en línea] http://invensense.com/mems/gyro/documents/RM-MPU-6000A.pdf. Consulta: 25 de junio de 2013.

- Configuración del umbral de detección para cuando se da un evento de caída libre y determinación de la duración del mismo.

- Configuración y obtención de detección de umbral cuando no hay movimiento. Esto sucede cuando en los tres ejes del sensor no se sobrepasa el umbral que se ha preestablecido anteriormente para determinar si hubo un movimiento.

- Obtención de la duración en que no se realiza algún movimiento. Existe un registro dentro del sensor que hace el modo de contador a una frecuencia de 16 Hz cuyo valor incrementa por cada ciclo si aún no se sobrepasa el umbral preestablecido para la detección de movimiento.

- Configuración del registro FIFO_EN: en este se determina qué mediciones de los sensores (giroscopio y acelerómetro) serán cargados.

- Configuración de I2C para modo maestro
    - Habilitación del bus I2C en modo multi-maestro
    - Habilitación de interrupción de envío de datos para ser enviados a los registros externos dedicados para guardar dicha información.
    - Cargar los datos desde los registros externos hacia la FIFO.
    - Obtención y configuración de la velocidad de reloj del dispositivo I2C maestro.

- Configuración de registros de I2C en modo esclavo
    - Obtención y configuración de la dirección I2C del esclavo
    - Obtención y configuración de registro interno del esclavo
    - Obtención y configuración del valor de habilitación del esclavo
    - Especificación del modo escritura o lectura de acuerdo a la modificación del *bit* dentro del registro en mención.
    - Obtención del tamaño de datos

- Configuración de registro de estado del maestro I2C

- Configuración de los modos de interrupción

- Configuración de registro de habilitación de señales de interrupción en los pines dedicados para esto.

- Habilitación de generación de interrupción por otras fuentes. Estas son:
    - Detección de movimiento
    - Desborde de datos en la FIFO
    - Habilitación en modo maestro del I2C
    - Buffer de datos lleno

- Obtención de los datos de salida de acelerómetro y giroscopio
    - Se realiza la lectura de los registros externos donde se encuentran almacenados los datos, estos se almacenan en variables haciendo corrimientos de los *bits* que se encuentran en otra variable de paso ya que cada lectura tiene un tamaño de 16 *bits* y se mueven en grupos de 8 *bits* (*byte* por *byte*).

- Detección de estado de interrupciones de movimiento en cada eje

- Reseteo del patrón de salida de giroscopio

- Control de usuario sobre la FIFO. Permite:
    - Habilitación o reseteo del buffer de la FIFO
    - Habilitación o reseteo del modo master e interfaz primaria I2C

- Configuración del registro uno de manejo de energía y fuente de reloj
    - Reseteo del dispositivo completo (MPU6050)
    - Configuración de los modos de dormir y despertar del MPU6050
    - Configuración del reloj del sensor. Se utiliza el oscilador interno de 8 MHz.

- Lectura y escritura desde registros externos de datos hacia la FIFO a través de la configuración del registro correspondiente. Los datos se almacenan en el *buffer* de la FIFO de acuerdo al número de registro, desde el más bajo hacia el más alto, esto sucede si las banderas de habilitación de la FIFO tiene valor igual a 1.

- Obtención y configuración del registro WHO_AM_I, en este se guarda la dirección del dispositivo (MPU6050), usualmente es 0x34.

- Configuración de registros de ajuste de desvío de ejes

- Lectura de estado del DMP

- Habilitación y reseteo del DMP

- Configuraciones de bancos de memoria

- Configuraciones en registros de memoria utilizada para lectura y escritura.

- Configuración de registro propios del DMP

Existe una serie de funciones adicionales dentro del programa que no se describen en el presente documento debido a que no son de interés para el proyecto.

3.2.3. MPU6050.h

El archivo de cabecera MPU6050.h contiene lo siguiente:

- Declaraciones de constantes que hacen referencia a los registros del MPU6050, como por ejemplo: #define MPU6050_RA_FIFO_EN 0x23.

- Declaraciones de constantes que hacen referencia a un *bit* dentro de un registro, como por ejemplo: #define MPU6050_TC_PWR_MODE_BIT 7.

- Definición de la clase MPU6050. Dentro de la misma se encuentra la declaración de las funciones públicas utilizadas en el archivo de implementación. Estas pueden subdividirse de la siguiente manera:
    - Funciones tipo *void*, como por ejemplo:
        - *void* setYGyroFIFOEnabled(bool enabled);
    - Funciones de tipo entero sin signo, como por ejemplo:
        - uint8_t getExternalSensorByte(int position);
        - uint16_t getExternalSensorWord(int position);
    - Funciones de tipo entero con signo, como por ejemplo:
        - int16_t getRotationY();
    - Funciones de tipo *boolean*, como por ejemplo:
        - bool getXGyroFIFOEnabled();
    - Funciones que utilizan punteros dentro del argumento
        - uint8_t dmpGetQuaternion(Quaternion *q, const uint8_t* packet=0);.

En general, este archivo contiene todas las declaraciones de constantes, variables y funciones que se necesitan en los otros archivos de implementación del módulo.

3.2.4. MPU6050_6Axis_MotionApps20.h

El archivo de cabecera MPU6050_6Axis_MotionApps20.h es un poco más complejo que los anteriores, ya que ahora, adicional al manejo de registros generales del MPU6050, también se realizan todas las configuraciones para el

uso de los datos matemáticos proporcionados por el DMP. Este puede subdividirse en lo siguiente:

- Programación de los bancos de memoria del DMP cada vez que se inicializa.

- Direcciones de configuración del DMP dentro de los espacios del banco de memoria.

- Función “dmpInitialize()”
    - Reseteo del dispositivo
    - Deshabilitar el modo de bajo consumo
    - Revisión del hardware del MPU
    - Obtención de desajustes en ejes de giroscopio
    - Configuración de dirección de esclavo I2C
    - Deshabilitar modo maestro del I2C esclavo
    - Carga del código del DMP en los bancos de memoria
    - Escritura de la configuración del DMP en los bancos de memoria
    - Establecimiento de valores a las funciones de configuración que se encuentran dentro del archivo MPU6050.
        - Configuración del reloj
        - Habilitación de interrupciones
        - Velocidad de muestreo igual a 200 Hz
        - Ancho de banda del filtro digital paso bajo igual 42 Hz
        - Habilitación del giroscopio a +/- 2 000 grados por segundo
        - Umbrales de detección de movimiento
        - Tamaño del *buffer* igual a 42 *bytes*
        - Tiempo de duración de detección de movimiento igual a 80 ms.

- Reseteo de la FIFO
- Lectura de contador de la FIFO
- Reseteo del DMP

- Funciones de obtención de los diferentes datos provenientes del DMP a través del uso de punteros donde se han almacenado estos, en el proceso de lectura I2C.
  - Obtención de datos de aceleración en los tres ejes
  - Obtención de cuaterniones, se guardan en el puntero q
  - Obtención de datos de giroscopio
  - Obtención de aceleraciones lineales
  - Obtención de ángulos de Euler
  - Obtención de ángulos *yaw, pitch* y *roll*

- Lectura y procesamiento de paquetes de la FIFO

- Obtención del tamaño del paquete de la FIFO

3.2.5. Helper_3dmath.h

El archivo de cabecera helper_3dmath.h contiene una serie de funciones que manipulan matemáticamente los cuaterniones provenientes del DMP, con el propósito de obtener los ángulos *yaw*, *pitch* y *roll*. Este contiene tres clases que a continuación se describen:

- Clase Quaternion. Dentro de esta clase se declaran las variables que contienen el valor escalar y los valores imaginarios del cuaternión. Entonces se realizan operaciones con estos valores para facilitar el cálculo de las conversiones a otros tipos de ángulos.

- GetProduct(Quaternion q): esta función realiza multiplicaciones entre los valores del cuaternión.
- Obtención del conjugado del cuaternión
- Obtención de la magnitud
- Normalización del cuaternión

- Clase VectorInt16. Dentro de esta clase se obtiene un nuevo cuaternión a partir de la rotación del cuaternión inicial.

- Clase VectorFloat. Se utiliza para obtener un vector con los valores de gravedad asociados a cada eje.

3.2.6. I2Cdev.cpp

El archivo I2C contiene todas las funciones de lectura y escritura hacia el MPU6050. Pueden clasificarse a estas, de acuerdo a la cantidad de datos que se manipulan.

- Funciones de lectura. Dentro de los argumentos de estas se incluye: la dirección I2C del dispositivo, el nombre del registro a leer, tiempo de lectura y estado de lectura, entre otros aspectos, como el tamaño del dato a leer. Las funciones declaradas son las siguientes:
  - Lectura de un solo *bit* desde un registro de 8 *bits*
  - Lectura de un solo *bit* desde un registro de 16 *bits*
  - Lectura de múltiples *bits* desde un registro de 8 *bits*
  - Lectura de múltiples *bits* desde un registro de 16 *bits*
  - Lectura de un único *byte* desde un registro de 8 *bit*s
  - Lectura de una sola palabra (dos *bytes*) desde un registro de 16 *bits*.

- Lectura de múltiples *bytes* desde un registro de 8 *bits*
- Lectura de múltiples palabras desde un registro de 16 *bits*

- Funciones de escritura. Dentro de los argumentos de estas se incluye; la dirección I2C del dispositivo, nombre del registro a escribirle y estado de lectura, entre otros aspectos como el tamaño del dato a escribir. Las funciones declaradas para esto son las siguientes:
  - Escritura de un solo *bit* hacia un registro de 8 *bits*
  - Escritura de un solo *bit* hacia un registro de 16 *bits*
  - Escritura de múltiples *bits* hacia un registro de 8 *bits*
  - Escritura de múltiples *bits* hacia un registro de 16 *bits*
  - Escritura de un único *byte* hacia un registro de 8 *bits*
  - Lectura de una sola palabra hacia un registro de 16 *bits*
  - Lectura de múltiples *bytes* hacia un registro de 8 *bits*
  - Lectura de múltiples palabras hacia un registro de 16 *bits*

3.2.7. I2Cdev.h

El archivo de cabecera I2Cdev.h contiene la clase I2Cdev. En esta última se encuentra la declaración de las funciones de lectura y escritura.

- Las funciones de lectura se declaran como tipo entero de 8 *bits* estáticas, como por ejemplo:
  - static int8_t readBytes(uint8_t devAddr, uint8_t regAddr, uint8_t length, uint8_t *data, uint16_t timeout=I2Cdev::readTimeout);

- Las funciones de escritura se declaran como tipo booleano estáticas, como por ejemplo:

- static bool writeBytes(uint8_t devAddr, uint8_t regAddr, uint8_t length, uint8_t *data);

La combinación de todos los archivos descritos anteriormente hace posible el flujo de datos (ver figura 11), para la manipulación y entrega a través del puerto serial hacia la interfaz programada en Python, encargada de analizarlos para llevar a cabo el movimiento del cursor en la pantalla.

4. PROCESAMIENTO DE DATOS EN PYTHON

Luego de la obtención de los datos matemáticos proporcionados por el DMP es necesario un último proceso para poder manipular el cursor. Utilizando la herramienta de programación Python se realiza la adquisición y análisis de los datos en tiempo real obtenidos por medio del UART para llevar a cabo la rutina de calibración con el usuario y posterior manipulación del cursor, entre otras características que se adicionan con el objetivo de facilitar el manejo.

4.1. Recepción en tiempo real por medio de UART

La obtención de datos en tiempo real se hace a través del uso del UART. Dentro de Python se utiliza el módulo pySerial para la configuración y manejo del puerto serial mediante las funciones básicas que se encuentran ya preestablecidas. La lectura de datos en el puerto serial es una de las tareas que deberán ejecutarse reiteradamente cuando se realice la rutina de calibración, o bien cuando ya se esté moviendo el cursor.

4.1.1. Configuración del puerto serial

Previo a utilizar el puerto serial es necesario realizar lo siguiente:

- Importar el módulo utilizando *import serial* (serial es el nombre en Python de dicho módulo).

- Definición de la clase serial utlizando *s=serial.Serial('puerto',velocidad en baudios)* sobre el objeto *s*.

- Se utiliza como puerto el ttyAMA0 a una velocidad de 115 200 baudios.

- *s.isOpen()* y *s.close()* para desconectar el puerto serial.

4.1.2. Lectura de datos en el puerto serial

La lectura de datos se realiza leyendo la información que se encuentra en el *buffer*: para esto se definió una función a la que pudiese llamarse en cada ocasión que se necesitara obtener datos desde el puerto serial. A continuación se detalla de forma genérica a esta:

- Definición de la función de lectura de puerto serial: *readSerial()*

- Definición de las variables globales y locales

- Lectura línea por línea haciendo instanciación de la función *.readline()* al objeto *s* (definición del puerto serial).

- Concatenación de los datos obtenidos en una única variable. Se remueven los espacios en blanco.

- Separación de los datos ya concatenados a través de la función *.split* para formar un nuevo vector.

- Obtención de los datos necesarios, en este caso el valor *yaw* y *roll* dentro del vector. Estos se cargan en las posiciones cero y uno de un nuevo vector, *ang[0]* y *ang[1]* respectivamente.

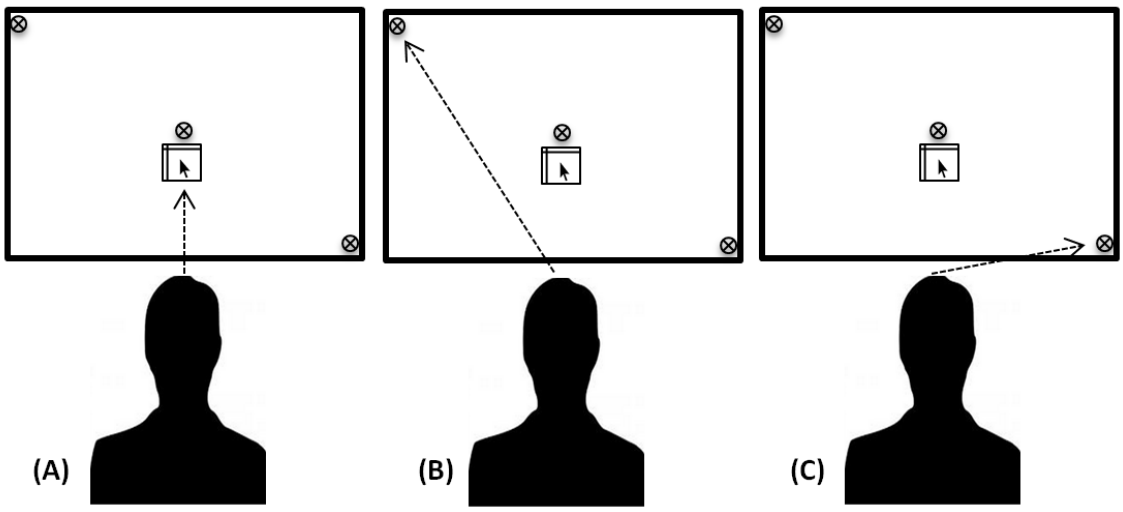
- Retorno de los ángulos *yaw* y *roll* en grados decimales

Cada que vez que se inicia la lectura del puerto serial es necesario limpiar el *buffer*: para esto se define la función *clearBuffer(),* la cual hace instanciación de la función .*flushInput* al objeto *s.*

4.2. Rutina de calibración

Durante la rutina de calibración el usuario debe dirigir la vista y cabeza hacia los puntos que se le indiquen (ver figura 12). Esto proporcionará los valores de ángulos (*yaw* y *roll)* máximos, para el caso de la esquina superior izquierda de la pantalla y mínimos, para el caso de la esquina inferior derecha de la pantalla.

Figura 12. **Rutina de calibración**



Fuente: elaboración propia.

La rutina de calibración hace uso de un GUI que despliega un botón, al que debe hacerse clic cuando se esté observando la posición indicada (ver figura 12). Adicionalmente, se hace uso de un sintetizador de voz que se dirige al usuario. El desarrollo de la GUI y la detección de clic se detallan más adelante.

Se puede dividir a la rutina de calibración en tres fases, A, B y C. A continuación se describe que funciones se ejecutan en cada una de ellas y la finalidad:

- Posición A
  - Al encender la Raspberry Pi automáticamente se inicializará la rutina de calibración, por lo que el usuario debe dirigir su vista hacia el centro de la pantalla hasta serle indicado que proceda a la posición B.
  - Se llama a la función clearBuffer() para eliminar cualquier dato innecesario previo a dirigirse a la posición B.

- Posición B
  - Se indicará dirigirse a la posición B, por medio de un audio que se ha realizado utilizando el sintetizador de voz.
  - Cuando el usuario se encuentre observando hacia la posición B deberá hacer clic (el cursor se encontrará sobre el botón). Esto automáticamente llamará a la función readSerial() almacenando dentro de las variables (maxX y maxY) los valores que definen los ángulos máximos de giro para observar la esquina superior izquierda.

- Posición C
    - Al igual que en la posición B, se indicará por medio de otro audio (sintetizador de voz) que el usuario dirija la vista hacia la posición C.
    - Cuando el usuario se encuentre observando hacia la posición C deberá hacer clic (el cursor se encontrará sobre el botón), esto automáticamente llamará a la función readSerial() almacenando dentro de las variables (minX y minY) los valores que definen los ángulos mínimos de giro para observar la esquina inferior derecha.

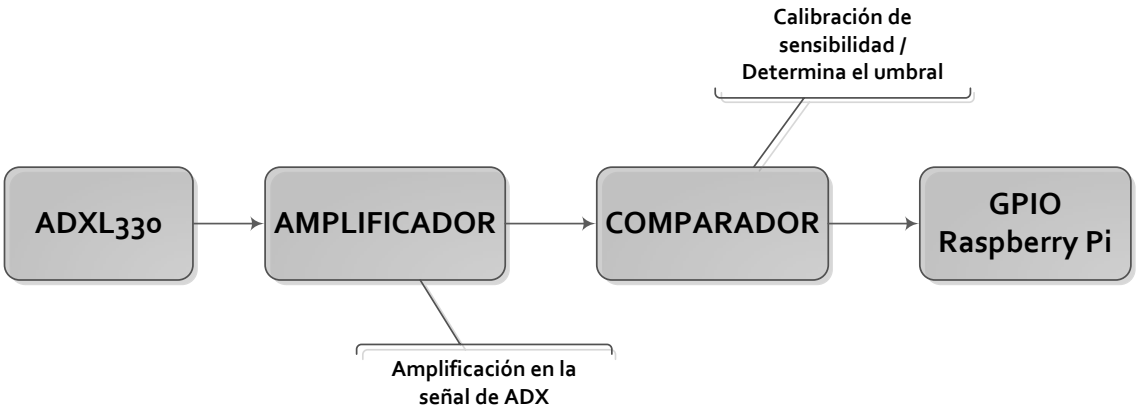
4.2.1. Detección de clic

La detección de clic se hace mediante un circuito adicional colocado sobre el hombro del usuario que envía una señal a la Raspberry Pi cuando se produce un movimiento hacia arriba.

4.2.1.1. Circuito electrónico de detección

El ADXL330 es un acelerómetro que proporciona una salida de voltaje de acuerdo a la aceleración que experimenta. Para ayudar a explicar de mejor manera el funcionamiento del circuito se agrega un diagrama de bloques.

Figura 13. **Diagrama de bloques**



Fuente: elaboración propia, con programa Microsoft Visio 2010.

El ADXL330 posee tres ejes: se utiliza el eje z, por lo que al colocar el sensor debe asegurarse que este se encuentre perpendicular al plano formado por el hombro. Su función principal radica en dar una señal de voltaje proporcional a la aceleración que experimenta.

El amplificador permite, como el nombre lo indica, dar amplificación controlada a la señal proveniente del ADXL330: se ha utilizado un amplificador operacional en configuración de tipo no inversor.

Por último el comparador permite la calibración del umbral de voltaje máximo que determinará si la señal a enviar es un uno o cero hacia el GPIO de la Raspberry Pi. El circuito utiliza un amplificador operacional y una resistencia variable que determina el voltaje de umbral en la pata inversora, por ende cuando la señal proveniente del amplificador sobrepase el voltaje de umbral el comparador enviará la señal que será interpretada como un clic.

4.2.1.2. Interpretación de señal en Python

Luego de obtener la señal desde el circuito, es necesario interpretarla para que se realice el clic en el ordenador. Para esto es necesario configurar el GPIO a donde llega la señal, definir el evento clic y realizar una función que monitoree dicho *pin*.

- Importación de módulos que se utilizan
  - Se importa el módulo RPi.GPIO que hace referencia a la distribución de pines de la Raspberry Pi.
  - Se importa el módulo PyMouse para manipular eventos propios del ratón.
  - Se importa la librería *time* para ser utilizada en la actividad de monitoreo.

- Configuraciones principales
  - Configuración del tipo de numeración que se asigna a los pines, utilizando el sistema de numeración de la tarjeta física *GPIO.setmode(GPIO.BOARD).*
  - Configuración del *pin* como entrada con resistencia *pull down*, *GPIO.setup(12, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)*
  - Instanciación del módulo *PyMouse().*

- Detección del evento en el *pin* doce, por medio de la función *GPIO.input(12).* Esto genera el llamado a la función que realiza el clic en la posición de acuerdo a los valores obtenidos desde las otras funciones que se han creado para esto.

- Por último se creó una función que llama continuamente a la función de detección de evento clic reiteradamente con un retraso de 5 ms.

4.2.2. GUI

Los botones que aparecen durante la rutina de calibración (ver figura 12) fueron creados utilizando el módulo *Tkinter:* este es la interfaz estándar de las herramientas para crear GUIs (interfaces gráficas de usuario) *Tk* en Python.

Cada uno de los botones se realizó por medio del dibujo de un GUI con ciertas propiedades y funciones que a continuación se describen:

- Al inicio del programa para crear el GUI es necesario importar el módulo *Tkinter.*

- Crear la ventana asociada a una variable para mejor manipulación de las propiedades, *var=tk.Tk()*

- Creación del frame dentro de la ventana, *frame=tk.Frame(var)*

- Creación del botón dentro del frame

    - *button=tk.button(frame, text= BOTON, command= .destroy()*
    - *Command* llama a la función *.destroy()* al dar clic para eliminar el botón, instanciando la ventana.

- Dibujar el botón, *button.pack()*

- Dibujar el frame, *frame.pack()*

- Realizar un update a todos los cambios que se desean en la ventana, utilizando la función *update()* instanciando a la ventana.

- Obtención del tamaño de la ventana por medio de las funciones estándar: *winfo_width()* y *winfo_height().*

- Obtención del tamaño de la pantalla por medio de las funciones estándar: *winfo_screenwidth()* y *winfo_screenheight().*

- Los valores obtenidos en los dos puntos anteriores se utilizan para determinar la posición en la que se desea dibujar el botón.

- Se dibuja el objeto en la posición que se desea utilizando la función *geometry(posición)*.

- Para finalizar, se utiliza la función *mainloop()* instanciando a la ventana (*var*), lo que despliega todos los objetos (ventana, frame y botón) con las propiedades previamente definidas.

El procedimiento descrito antes, se utiliza para crear dos botones: el primero cuando se obtiene la posición superior izquierda (función *botSupIzq()*)y el segundo para la posición inferior derecha (función *botInfDer()*).

4.2.3. Síntesis de voz

Para ayudar al usuario en los pasos de la rutina de calibración se hace uso del programa sintetizador de voz eSpeak. Una función determinada crea el nuevo archivo objeto (*os.popen(eSpeak y propiedades).read()*), en este caso un archivo *espeak* con las propiedades que se desea.

Entre las propiedades que se definieron se encuentran las siguientes:

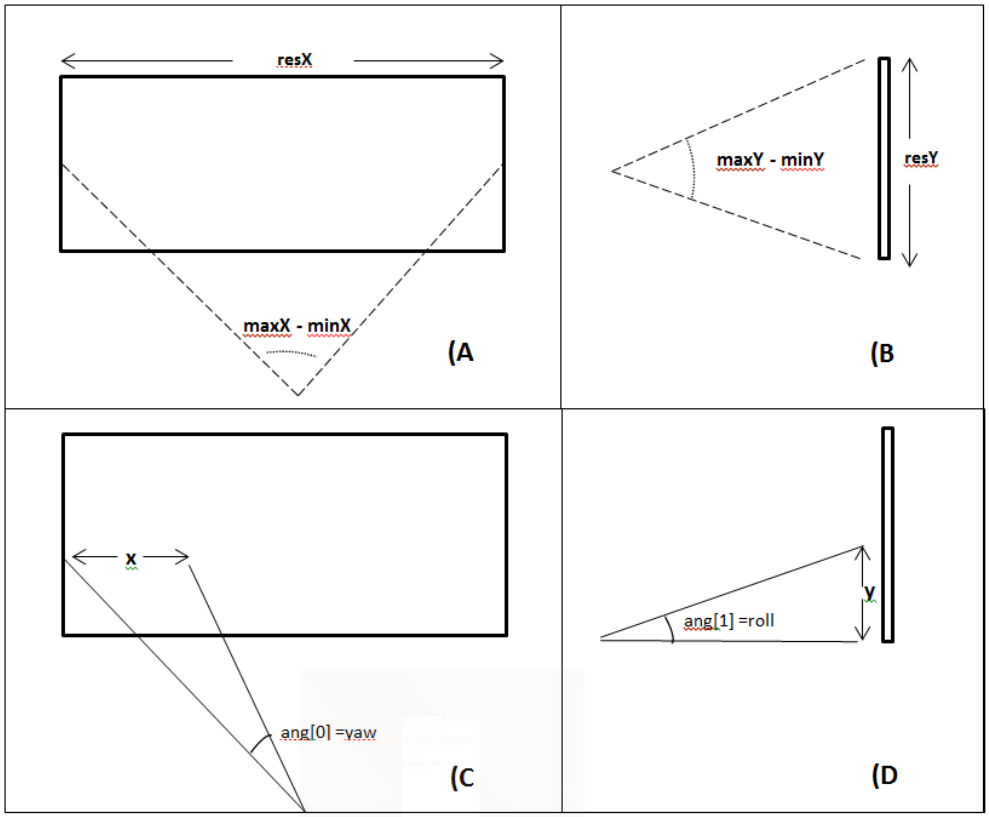
- Texto a leer
- Idioma español = *–ves-la*
- Voz femenina = *+f3*
- Volumen = 100
- Velocidad= 150

4.3. Manipulación del cursor

Se realiza mediante una serie de programas y funciones que ejecutan tareas específicas capaces de determinar los valores o parámetros que serán de utilidad en la obtención de la posición x e y donde se colocará el mismo.

A continuación se agrega una representación de los parámetros utilizados y la referencia en la pantalla (ver figura 14) para ayudar a explicar de mejor manera cada función.

Figura 14. **Representación gráfica de parámetros**



Fuente: elaboración propia.

4.3.1. Resolución de pantalla

La resolución de la pantalla se realiza con el objetivo de obtener las dimensiones de la pantalla, *resX* y *resY* (ver figura 14, A y B). Esto se hace de la siguiente manera:

- Utilizando el comando *xrandr* se obtiene la resolución de la pantalla, quien proporciona una sentencia con los valores (ancho y alto), que a la vez se carga sobre una variable (*screen).*

- Interesa obtener por separado los valores de ancho y alto, por lo que se utiliza la función *.split()[]* sobre la variable *screen*.
    - El ancho se obtiene con : *resX= screen.split()[7]*
    - El alto se obtiene con : *resY=screen.split()[9][:-1]*

4.3.2. Valores máximos y minimos de *yaw* y *roll*

Los valores máximos y mínimos de *yaw* y *roll* se obtienen desde la rutina de calibración, en la manipulación del cursor interesan; el cambio en el ángulo respecto al ángulo máximo (ver figura 14, C y D) que se da en cada movimiento y la proporcionalidad entre el máximo angulo o abertura y la resolución (ver figura 14, A y B).

- *(maxX – minX) ~ resX*
- *(maxY – minY) ~ resY*

4.3.3. Lectura de movimiento

Debido a que no es posible determinar cuándo el usuario querrá mover el cursor, es necesario mantener una lectura constante, así como la interpretación de los datos que se están recibiendo desde el puerto serial en tiempo real.

Por esta razón se crea una rutina que está siempre llamando a la función de lectura del puerto serial *readSerial()* que brinda el vector *ang[]*, desde donde *ang[0]*= ángulo *yaw* y *ang[1]*= ángulo *roll* (ver figura 14, C y D). Estos valores recaban el cambio en el ángulo que ha ocurrido de forma horizontal y vertical respectivamente.

4.3.4. Posición *x* e *y* del cursor

La posición x,y a la que se mueve el cursor se determina a partir del cálculo sobre los siguientes datos, cuya obtención se describió anteriormente.

- Resolución de la pantalla, ancho y alto: *resX, resY*

- Ángulos máximos y mínimos que describen los límites de la pantalla en cada eje.
  - Eje x, *maxX y minX*
  - Eje y, *maxY y min*Y

- Valores de ángulos *yaw* o *ang[0]* y *roll* o *ang[1]* que indican el movimiento realizado.

Como ya se conoce la correspondencia entre estos valores, el valor de *x* e *y* se obtiene haciendo una regla de tres de la siguiente manera:

- *x = ((-ang[0]+maxX)*resX)/(maxX – minX)*
- *y = ((-ang[1]+maxY)*resY)/(maxY – minY)*

Las expresiones *(-ang[0]+maxX)* y *(-ang[1]+maxY)* son equivalentes al cambio propio en el ángulo, ya que debe recordarse que los valores máximos y mínimos no se miden desde un valor cero, sino desde la izquierda hacia la derecha para, el caso del eje x y desde arriba hacia abajo para el caso del eje y.

4.3.5. Posicionamiento del cursor

Después de calcular los valores *x* e *y* se envían hacia la función que mueve el cursor.

- Se instancia sobre una variable (*m)* el módulo PyMouse()
- Utilizando el comando *.move* se coloca al cursor en la posición *x,y*.
    - *m.move(x,y)*

CONCLUSIONES

1. La determinación del cuaternión es más precisa, si ambos sistemas de ejes donde se mide la aceleración y velocidad angular se encuentran en el mismo punto de origen.

2. El cambio de movimiento vertical y horizontal se determina por: la variación en el valor del ángulo *yaw* y *roll* respectivamente.

3. En el manejo del MPU6050 debe manipularse uno a uno cada registro, haciéndolo un procedimiento complejo.

4. La Raspberry Pi es la computadora más apropiada debido al bajo costo, dimensiones pequeñas y alto rendimiento.

5. Python y los diferentes módulos son herramientas de programación que permiten manipular fácilmente las propiedades del cursor.

6. La determinación de la razón de proporcionalidad entre el movimiento de cabeza y el movimiento del cursor solo se pueden obtener por medio de la rutina de calibración.

RECOMENDACIONES

1. Para futuros proyectos con fines similares a este, puede emplearse también una computadora comercial, utilizando a la Raspberry Pi para obtener los datos del MPU6050 y enviarlos utilizando el puerto serial, puerto USB, etc.

2. Agregar una aplicación dentro de la computadora, donde el paciente pueda manipular la sensibilidad del circuito de detección de clic por medio de software, para evitar circuitería o cableado adicional.

3. Debido a que este proyecto se limita a la manipulación del cursor, es necesario el desarrollo de programas de aplicación de fácil uso, que permitan al paciente realizar tareas como escritura, lectura, navegación por internet, entre otros.

4. Establecer un procedimiento adicional que verifique si el usuario desea recalibrar el sistema.

BIBLIOGRAFÍA

1. *ADXL330*. [en línea] http://www.analog.com/static/imported-files/data_sheets/ADXL330.pdf. [Consulta: 15 de junio de 2013.]

2. *Ángulos yaw, roll y pitch*. [en línea] http://es.wikipedia.org/wiki/%C3%81ngulos_de_navegaci%C3%B3n. [Consulta: 18 de mayo de 2013.]

3. *C++.* [en línea] https://es.wikipedia.org/wiki/C%2B%2B. [Consulta: 21 de mayo de 2013.]

4. Configuración I2C en Raspberry Pi [en línea] http://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-setup/configuring-i2c. [Consulta: 01 de junio de 2013.]

5. *Cuaterniones*. [en línea] https://es.wikipedia.org/wiki/Cuaterniones_y_rotaci%C3%B3n_en_el_espacio. [Consulta: 18 de mayo de 2013.]

6. *eSpeak*. [en línea] http://espeak.sourceforge.net/. [Consulta: 28 de junio de 2013.]

7. *HDMI*. [en línea] http://es.wikipedia.org/wiki/High-Definition_Multimedia_Interface. [Consulta: 30 de junio de 2013.]

8. *I2C*. [en línea] http://es.wikipedia.org/wiki/I%C2%B2C [Consulta: 30 de junio de 2013.]

9. *Instalación de Raspbian en SD.* [en línea] http://sobrebits.com/montar-un-servidor-casero-con-raspberry-pi-parte-1-instalar-raspbian-en-una-tarjeta-sd/. [Consulta: 29 de mayo de 2013.]

10. Librerías estándar C++. [en línea] https://www2.opengroup.org/ogsys/jsp/publications/mainPage.jsp. [Consulta: 21 de mayo de 2013.]

11. *MPU6050.* [en línea] http://www.invensense.com/mems/gyro/mpu6050.html. [Consulta: 01 de mayo de 2013.]

12. MPU6050 Pi Demo. [en línea] http://www.raspberrypi.org/phpBB3/viewtopic.php?t=22266. [Consulta: 20 de mayo de 2013.]

13. MPU6050 Product Specification. [en línea] http://www.invensense.com/mems/gyro/documents/PS-MPU-6000A.pdf. [Consulta: 01 de mayo de 2013.]

14. MPU6050 Register Map. [en línea] http://www.invensense.com/mems/gyro/documents/RM-MPU-6000A.pdf. [Consulta: 01 de mayo de 2013.]

15. *pyMouse.* [en línea] https://github.com/pepijndevos/PyMouse/wiki/Documentation. [Consulta: 10 de junio de 2013.]

16. *pySerial.* [en línea] http://pyserial.sourceforge.net/pyserial.html#overview. [Consulta: 04 de junio de 2013.]

17. *Python*. [en línea] http://www.python.org/. [Consulta: 21 de mayo de 2013.]

18. *Raspberry Pi.* [en línea] http://www.rasperry.org. [Consulta: 28 de mayo de 2013.]

19. *UART.* [en línea] http://es.wikipedia.org/wiki/Universal_Asynchronous_Receiver-Transmitter. [Consulta: 30 de junio de 2013.]

20. Win32 Disk Imager. [en línea] http://sourceforge.net/projects/win32diskimager/. [Consulta: 29 de mayo de 2013.]

APÉNDICE

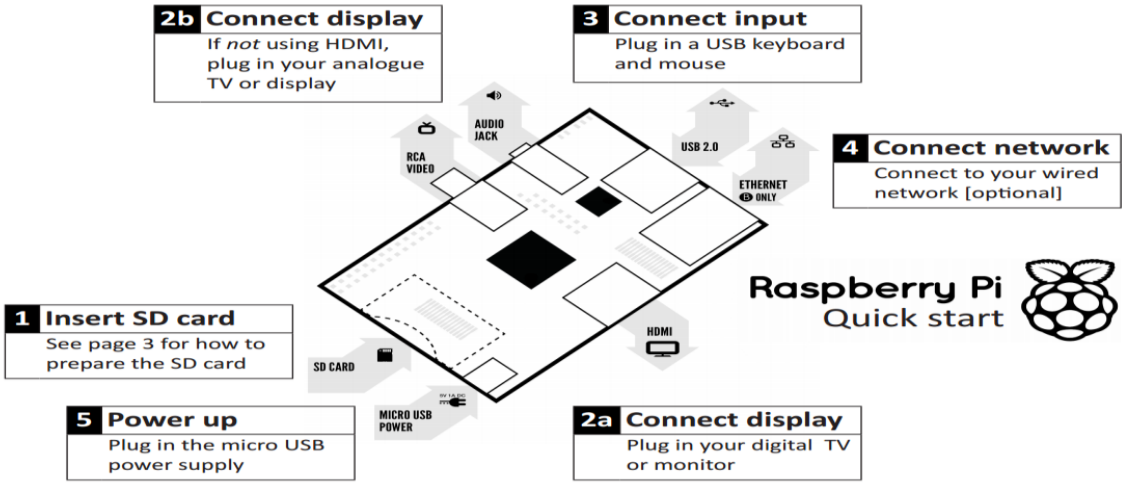
MANUAL DEL USUARIO

En este apartado se presentan las instrucciones necesarias para que el usuario final pueda utilizar el dispositivo diseñado en esta tesis de manera correcta.

Al inicio el usuario debe asegurarse de contar con todos los dispositivos adicionales que se conectan a la Raspberry Pi. De acuerdo con la figura 23, estos son:

- Memoria SD (1)
- Pantalla (TV) por HDMI (2a)
- Teclado (3), de ser necesario.
- Cable Ethernet o adaptador inalámbrico para Internet (4)
- Cable microusb para energía (5)

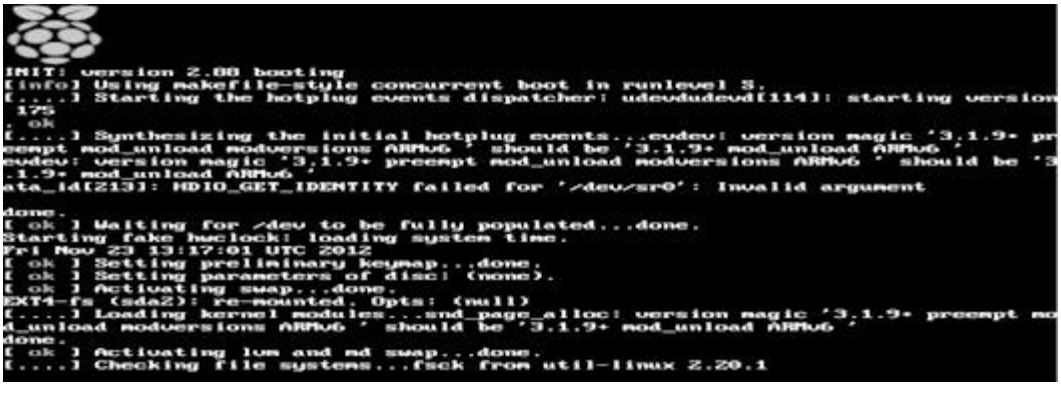
Figura A. **Diagrama Raspberry Pi**



Fuente: Raspberry Pi. http://www.raspberrypi.org/wp-content/uploads/2012/04/quick-start-guide-v2_1.pdf. Consulta: 28 de julio de 2013.

Luego de energizar la Raspberry Pi, en la pantalla el usuario observará como se carga el sistema operativo (ver figura 24).

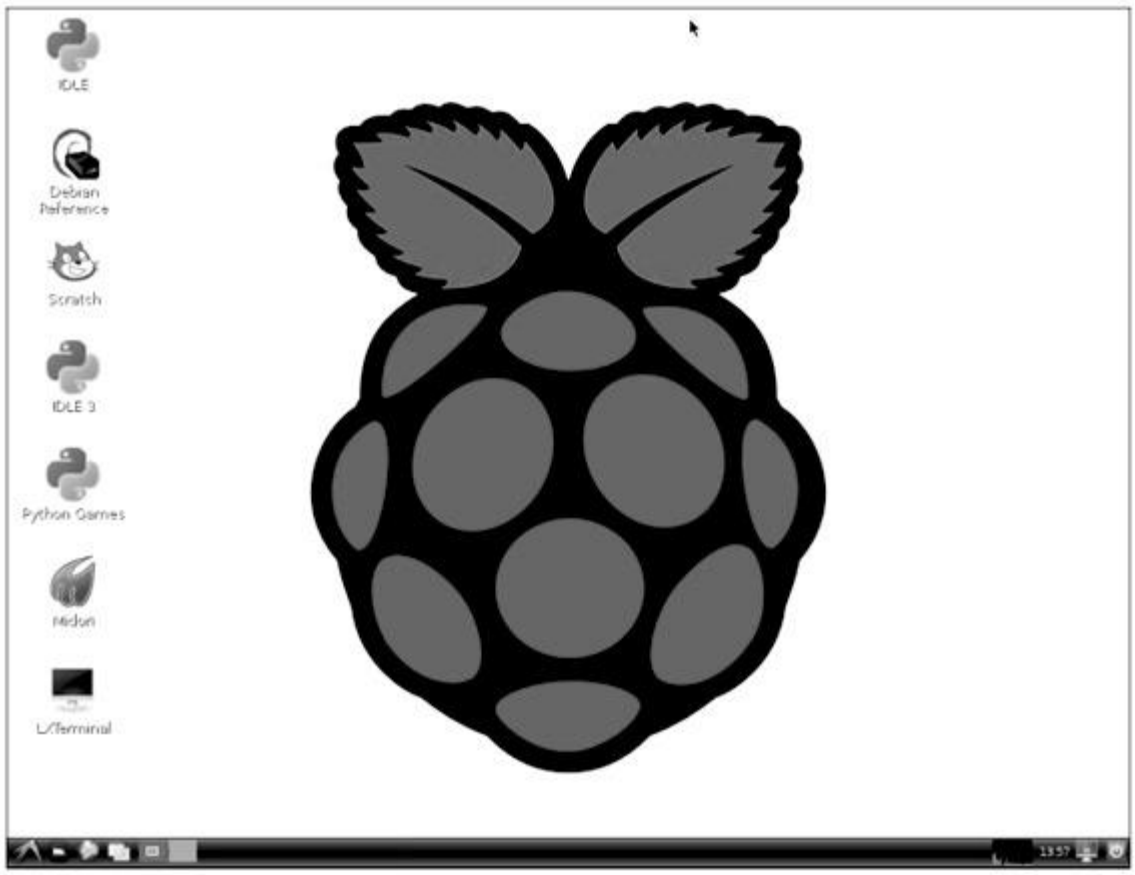
Figura B. **Inicialización de Raspbian**



Fuente: Arranque Raspberry Pi. http://reviews.cnet.co.uk/desktops/how-to-get-started-with-the-raspberry-pi-50009845/. Consulta: 06 de junio de 2013.

Seguido escuchará un audio indicándole que la rutina de calibración ha iniciado, la pantalla mostrará el escritorio de la Raspberry Pi (ver Figura 25).

Figura C. **Escritorio de Raspberry Pi**



Fuente: Escritorio Raspberry Pi. http://reviews.cnet.co.uk/desktops/how-to-get-started-with-the-raspberry-pi-50009845/. Consulta: 28 de julio de 2013.

Entonces la rutina de calibración se dará por iniciada y el usuario debe seguir los siguientes pasos, de acuerdo con la figura 26.

- El usuario debe mover su cabeza para observar hacia el centro de la pantalla, aparecerá un botón tal como lo muestra la figura 26, A.

- Seguido un audio le indicará que mueva la cabeza para observar la esquina superior izquierda (ver figura 26, B).

- Entonces debe levantar su hombro para dar clic sobre el botón que ha aparecido al centro de su pantalla.

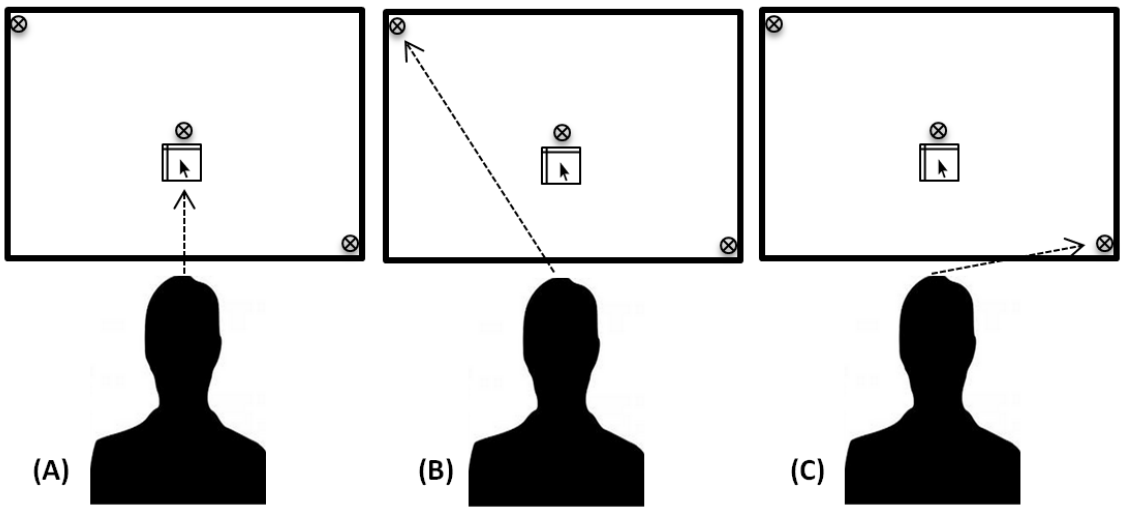
- Luego otro audio le indicará que mueva la cabeza para observar la esquina inferior izquierda (ver Figura 26, C).

- De nuevo debe levantar su hombro para dar clic sobre el botón al centro de su pantalla.

- Por último debe esperar unos segundos hasta que un audio le indique que la rutina de calibración ha sido finalizada.

- El cursor está listo para que usted lo mueva y maneje la computadora.

Figura D. **Diagrama de calibración**



Fuente: elaboración propia.

ANEXOS

Tabla A. **Especificaciones del giroscopio de MPU6050**

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
GYROSCOPE SENSITIVITY						
Full-Scale Range	FS_SEL=0		±250		°/s	
	FS_SEL=1		±500		°/s	
	FS_SEL=2		±1000		°/s	
	FS_SEL=3		±2000		°/s	
Gyroscope ADC Word Length			16		bits	
Sensitivity Scale Factor	FS_SEL=0		131		LSB/(°/s)	
	FS_SEL=1		65.5		LSB/(°/s)	
	FS_SEL=2		32.8		LSB/(°/s)	
	FS_SEL=3		16.4		LSB/(°/s)	
Sensitivity Scale Factor Tolerance	25°C	-3		+3	%	
Sensitivity Scale Factor Variation Over Temperature			±2		%	
Nonlinearity	Best fit straight line; 25°C		0.2		%	
Cross-Axis Sensitivity			±2		%	
GYROSCOPE ZERO-RATE OUTPUT (ZRO)						
Initial ZRO Tolerance	25°C		±20		°/s	
ZRO Variation Over Temperature	-40°C to +85°C		±20		°/s	
Power-Supply Sensitivity (1-10Hz)	Sine wave, 100mVpp; VDD=2.5V		0.2		°/s	
Power-Supply Sensitivity (10 - 250Hz)	Sine wave, 100mVpp; VDD=2.5V		0.2		°/s	
Power-Supply Sensitivity (250Hz - 100kHz)	Sine wave, 100mVpp; VDD=2.5V		4		°/s	
Linear Acceleration Sensitivity	Static		0.1		°/s/*g*	
SELF-TEST RESPONSE						
Relative	Change from factory trim	-14		14	%	1
GYROSCOPE NOISE PERFORMANCE	**FS_SEL=0**					
Total RMS Noise	DLPFCFG=2 (100Hz)		0.05		°/s-rms	
Low-frequency RMS noise	Bandwidth 1Hz to10Hz		0.033		°/s-rms	
Rate Noise Spectral Density	At 10Hz		0.005		°/s/ √ Hz	
GYROSCOPE MECHANICAL FREQUENCIES						
X-Axis		30	33	36	kHz	
Y-Axis		27	30	33	kHz	
Z-Axis		24	27	30	kHz	
LOW PASS FILTER RESPONSE						
	Programmable Range	5		256	Hz	
OUTPUT DATA RATE						
	Programmable	4		8,000	Hz	
GYROSCOPE START-UP TIME	**DLPFCFG=0**					
ZRO Settling (from power-on)	to ±1°/s of Final		30		ms	

Fuente: MPU6050 Product Specification. [en línea]
http://www.invensense.com/mems/gyro/documents/PS-MPU-6000A.pdf.
[Consulta: 10 de mayo de 2013.]

Tabla B. **Especificaciones del acelerómetro de MPU6050**

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
ACCELEROMETER SENSITIVITY						
Full-Scale Range	AFS_SEL=0		±2		*g*	
	AFS_SEL=1		±4		*g*	
	AFS_SEL=2		±8		*g*	
	AFS_SEL=3		±16		*g*	
ADC Word Length	Output in two's complement format		16		bits	
Sensitivity Scale Factor	AFS_SEL=0		16,384		LSB/*g*	
	AFS_SEL=1		8,192		LSB/*g*	
	AFS_SEL=2		4,096		LSB/*g*	
	AFS_SEL=3		2,048		LSB/*g*	
Initial Calibration Tolerance			±3		%	
Sensitivity Change vs. Temperature	AFS_SEL=0, -40°C to +85°C		±0.02		%/°C	
Nonlinearity	Best Fit Straight Line		0.5		%	
Cross-Axis Sensitivity			±2		%	
ZERO-G OUTPUT						
Initial Calibration Tolerance	X and Y axes		±50		*mg*	1
	Z axis		±80		*mg*	
Zero-G Level Change vs. Temperature	X and Y axes, 0°C to +70°C		±35			
	Z axis, 0°C to +70°C		±60		*mg*	
SELF TEST RESPONSE						
Relative	Change from factory trim	-14		14	%	2
NOISE PERFORMANCE						
Power Spectral Density	@10Hz, AFS_SEL=0 & ODR=1kHz		400		µg/ √ Hz	
LOW PASS FILTER RESPONSE						
	Programmable Range	5		260	Hz	
OUTPUT DATA RATE						
	Programmable Range	4		1,000	Hz	
INTELLIGENCE FUNCTION INCREMENT			32		mg/LSB	

Fuente: MPU6050 Product Specification. [en línea]
http://www.invensense.com/mems/gyro/documents/PS-MPU-6000A.pdf.
[Consulta: 10 de mayo de 2013.]

Tabla C. **Descripción de pines de salida y tipos de señal en MPU6050**

Pin Number	MPU-6000	MPU-6050	Pin Name	Pin Description
1	Y	Y	CLKIN	Optional external reference clock input. Connect to GND if unused.
6	Y	Y	AUX_DA	I^2C master serial data, for connecting to external sensors
7	Y	Y	AUX_CL	I^2C Master serial clock, for connecting to external sensors
8	Y		/CS	SPI chip select (0=SPI mode)
8		Y	VLOGIC	Digital I/O supply voltage
9	Y		AD0 / SDO	I^2C Slave Address LSB (AD0); SPI serial data output (SDO)
9		Y	AD0	I^2C Slave Address LSB (AD0)
10	Y	Y	REGOUT	Regulator filter capacitor connection
11	Y	Y	FSYNC	Frame synchronization digital input. Connect to GND if unused.
12	Y	Y	INT	Interrupt digital output (totem pole or open-drain)
13	Y	Y	VDD	Power supply voltage and Digital I/O supply voltage
18	Y	Y	GND	Power supply ground
19, 21	Y	Y	RESV	Reserved. Do not connect.
20	Y	Y	CPOUT	Charge pump capacitor connection
22	Y	Y	CLKOUT	System clock output
23	Y		SCL / SCLK	I^2C serial clock (SCL); SPI serial clock (SCLK)
23		Y	SCL	I^2C serial clock (SCL)
24	Y		SDA / SDI	I^2C serial data (SDA); SPI serial data input (SDI)
24		Y	SDA	I^2C serial data (SDA)
2, 3, 4, 5, 14, 15, 16, 17	Y	Y	NC	Not internally connected. May be used for PCB trace routing.

Fuente: MPU6050 Product Specification. [en línea] http://www.invensense.com/mems/gyro/documents/PS-MPU-6000A.pdf. [Consulta: 10 de mayo de 2013.]

Figura A. **Diagrama de bloques ADXL330**

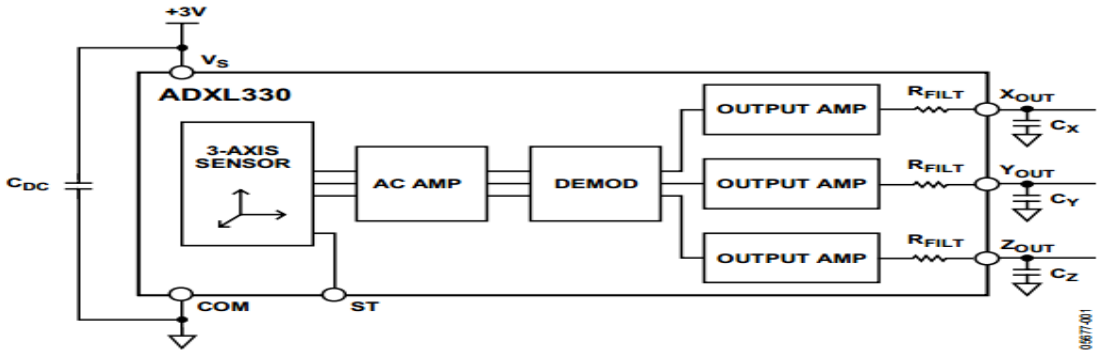


Figure 1.

Fuente: ADXL330. [en línea] http://www.analog.com/static/imported-files/data_sheets/ADXL330.pdf. [Consulta: 10 de junio de 2013.]

Tabla D. **Mapa de registros MPU6050**

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0D	13	SELF_TEST_X	R/W	XA_TEST[4-2]			XG_TEST[4-0]				
0E	14	SELF_TEST_Y	R/W	YA_TEST[4-2]			YG_TEST[4-0]				
0F	15	SELF_TEST_Z	R/W	ZA_TEST[4-2]			ZG_TEST[4-0]				
10	16	SELF_TEST_A	R/W	RESERVED		XA_TEST[1-0]		YA_TEST[1-0]		ZA_TEST[1-0]	
19	25	SMPLRT_DIV	R/W	SMPLRT_DIV[7:0]							
1A	26	CONFIG	R/W	-	-	EXT_SYNC_SET[2:0]			DLPF_CFG[2:0]		
1B	27	GYRO_CONFIG	R/W	-	-	-	FS_SEL [1:0]		-	-	-
1C	28	ACCEL_CONFIG	R/W	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]				
1F	31	MOT_THR	R/W	MOT_THR[7:0]							
23	35	FIFO_EN	R/W	TEMP _FIFO_EN	XG _FIFO_EN	YG _FIFO_EN	ZG _FIFO_EN	ACCEL _FIFO_EN	SLV2 _FIFO_EN	SLV1 _FIFO_EN	SLV0 _FIFO_EN
24	36	I2C_MST_CTRL	R/W	MULT _MST_EN	WAIT _FOR_ES	SLV_3 _FIFO_EN	I2C_MST _P_NSR	I2C_MST_CLK[3:0]			
25	37	I2C_SLV0_ADDR	R/W	I2C_SLV0 _RW	I2C_SLV0_ADDR[6:0]						
26	38	I2C_SLV0_REG	R/W	I2C_SLV0_REG[7:0]							
27	39	I2C_SLV0_CTRL	R/W	I2C_SLV0 _EN	I2C_SLV0 _BYTE_SW	I2C_SLV0 _REG_DIS	I2C_SLV0 _GRP	I2C_SLV0_LEN[3:0]			
28	40	I2C_SLV1_ADDR	R/W	I2C_SLV1 _RW	I2C_SLV1_ADDR[6:0]						
29	41	I2C_SLV1_REG	R/W	I2C_SLV1_REG[7:0]							
2A	42	I2C_SLV1_CTRL	R/W	I2C_SLV1 _EN	I2C_SLV1 _BYTE_SW	I2C_SLV1 _REG_DIS	I2C_SLV1 _GRP	I2C_SLV1_LEN[3:0]			
2B	43	I2C_SLV2_ADDR	R/W	I2C_SLV2 _RW	I2C_SLV2_ADDR[6:0]						
2C	44	I2C_SLV2_REG	R/W	I2C_SLV2_REG[7:0]							
2D	45	I2C_SLV2_CTRL	R/W	I2C_SLV2 _EN	I2C_SLV2 _BYTE_SW	I2C_SLV2 _REG_DIS	I2C_SLV2 _GRP	I2C_SLV2_LEN[3:0]			
2E	46	I2C_SLV3_ADDR	R/W	I2C_SLV3 _RW	I2C_SLV3_ADDR[6:0]						
2F	47	I2C_SLV3_REG	R/W	I2C_SLV3_REG[7:0]							
30	48	I2C_SLV3_CTRL	R/W	I2C_SLV3 _EN	I2C_SLV3 _BYTE_SW	I2C_SLV3 _REG_DIS	I2C_SLV3 _GRP	I2C_SLV3_LEN[3:0]			
31	49	I2C_SLV4_ADDR	R/W	I2C_SLV4 _RW	I2C_SLV4_ADDR[6:0]						
32	50	I2C_SLV4_REG	R/W	I2C_SLV4_REG[7:0]							
33	51	I2C_SLV4_DO	R/W	I2C_SLV4_DO[7:0]							
34	52	I2C_SLV4_CTRL	R/W	I2C_SLV4 _EN	I2C_SLV4 _INT_EN	I2C_SLV4 _REG_DIS	I2C_MST_DLY[4:0]				
35	53	I2C_SLV4_DI	R	I2C_SLV4_DI[7:0]							
36	54	I2C_MST_STATUS	R	PASS _THROUGH	I2C_SLV4 _DONE	I2C_LOST _ARB	I2C_SLV4 _NACK	I2C_SLV3 _NACK	I2C_SLV2 _NACK	I2C_SLV1 _NACK	I2C_SLV0 _NACK
37	55	INT_PIN_CFG	R/W	INT_LEVEL	INT_OPEN	LATCH _INT_EN	INT_RD _CLEAR	FSYNC_ INT_LEVEL	FSYNC _INT_EN	I2C _BYPASS _EN	-
38	56	INT_ENABLE	R/W	-	MOT_EN	-	FIFO _OFLOW _EN	I2C_MST _INT_EN	-	-	DATA _RDY_EN

Fuente: MPU6050 Register Map. [en línea]
http://www.invensense.com/mems/gyro/documents/RM-MPU-6000A.pdf.
[Consulta: 10 de mayo de 2013.]

Tabla E. **Mapa de registros MPU6050**

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3A	58	INT_STATUS	R	-	MOT_INT	-	FIFO _OFLOW _INT	I2C_MST _INT	-	-	DATA _RDY_INT
3B	59	ACCEL_XOUT_H	R	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT_L	R	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT_H	R	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT_L	R	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT_H	R	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT_L	R	ACCEL_ZOUT[7:0]							
41	65	TEMP_OUT_H	R	TEMP_OUT[15:8]							
42	66	TEMP_OUT_L	R	TEMP_OUT[7:0]							
43	67	GYRO_XOUT_H	R	GYRO_XOUT[15:8]							
44	68	GYRO_XOUT_L	R	GYRO_XOUT[7:0]							
45	69	GYRO_YOUT_H	R	GYRO_YOUT[15:8]							
46	70	GYRO_YOUT_L	R	GYRO_YOUT[7:0]							
47	71	GYRO_ZOUT_H	R	GYRO_ZOUT[15:8]							
48	72	GYRO_ZOUT_L	R	GYRO_ZOUT[7:0]							
49	73	EXT_SENS_DATA_00	R	EXT_SENS_DATA_00[7:0]							
4A	74	EXT_SENS_DATA_01	R	EXT_SENS_DATA_01[7:0]							
4B	75	EXT_SENS_DATA_02	R	EXT_SENS_DATA_02[7:0]							
4C	76	EXT_SENS_DATA_03	R	EXT_SENS_DATA_03[7:0]							
4D	77	EXT_SENS_DATA_04	R	EXT_SENS_DATA_04[7:0]							
4E	78	EXT_SENS_DATA_05	R	EXT_SENS_DATA_05[7:0]							
4F	79	EXT_SENS_DATA_06	R	EXT_SENS_DATA_06[7:0]							
50	80	EXT_SENS_DATA_07	R	EXT_SENS_DATA_07[7:0]							
51	81	EXT_SENS_DATA_08	R	EXT_SENS_DATA_08[7:0]							
52	82	EXT_SENS_DATA_09	R	EXT_SENS_DATA_09[7:0]							
53	83	EXT_SENS_DATA_10	R	EXT_SENS_DATA_10[7:0]							
54	84	EXT_SENS_DATA_11	R	EXT_SENS_DATA_11[7:0]							
55	85	EXT_SENS_DATA_12	R	EXT_SENS_DATA_12[7:0]							
56	86	EXT_SENS_DATA_13	R	EXT_SENS_DATA_13[7:0]							
57	87	EXT_SENS_DATA_14	R	EXT_SENS_DATA_14[7:0]							
58	88	EXT_SENS_DATA_15	R	EXT_SENS_DATA_15[7:0]							
59	89	EXT_SENS_DATA_16	R	EXT_SENS_DATA_16[7:0]							
5A	90	EXT_SENS_DATA_17	R	EXT_SENS_DATA_17[7:0]							
5B	91	EXT_SENS_DATA_18	R	EXT_SENS_DATA_18[7:0]							
5C	92	EXT_SENS_DATA_19	R	EXT_SENS_DATA_19[7:0]							
5D	93	EXT_SENS_DATA_20	R	EXT_SENS_DATA_20[7:0]							
5E	94	EXT_SENS_DATA_21	R	EXT_SENS_DATA_21[7:0]							
5F	95	EXT_SENS_DATA_22	R	EXT_SENS_DATA_22[7:0]							
60	96	EXT_SENS_DATA_23	R	EXT_SENS_DATA_23[7:0]							
63	99	I2C_SLV0_DO	R/W	I2C_SLV0_DO[7:0]							
64	100	I2C_SLV1_DO	R/W	I2C_SLV1_DO[7:0]							

Fuente: MPU6050 Register Map. [en línea]
http://www.invensense.com/mems/gyro/documents/RM-MPU-6000A.pdf.
[Consulta: 10 de mayo de 2013.]

Tabla F. **Mapa de registros MPU6050**

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
65	101	I2C_SLV2_DO	R/W	I2C_SLV2_DO[7:0]							
66	102	I2C_SLV3_DO	R/W	I2C_SLV3_DO[7:0]							
67	103	I2C_MST_DELAY_CTRL	R/W	DELAY_ES_SHADOW	-	-	I2C_SLV4_DLY_EN	I2C_SLV3_DLY_EN	I2C_SLV2_DLY_EN	I2C_SLV1_DLY_EN	I2C_SLV0_DLY_EN
68	104	SIGNAL_PATH_RESET	R/W	-	-	-	-	-	GYRO_RESET	ACCEL_RESET	TEMP_RESET
69	105	MOT_DETECT_CTRL	R/W	-	-	ACCEL_ON_DELAY[1:0]		-		-	
6A	106	USER_CTRL	R/W	-	FIFO_EN	I2C_MST_EN	I2C_IF_DIS	-	FIFO_RESET	I2C_MST_RESET	SIG_COND_RESET
6B	107	PWR_MGMT_1	R/W	DEVICE_RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		
6C	108	PWR_MGMT_2	R/W	LP_WAKE_CTRL[1:0]		STBY_XA	STBY_YA	STBY_ZA	STBY_XG	STBY_YG	STBY_ZG
72	114	FIFO_COUNTH	R/W	FIFO_COUNT[15:8]							
73	115	FIFO_COUNTL	R/W	FIFO_COUNT[7:0]							
74	116	FIFO_R_W	R/W	FIFO_DATA[7:0]							
75	117	WHO_AM_I	R	-	WHO_AM_I[6:1]						-

Fuente: MPU6050 Register Map. [en línea]
http://www.invensense.com/mems/gyro/documents/RM-MPU-6000A.pdf.
[Consulta: 10 de mayo de 2013.]

Tabla G. **Especificaciones del ADXL330**

Parameter	Conditions	Min	Typ	Max	Unit
SENSOR INPUT	Each axis				
Measurement Range		±3	±3.6		*g*
Nonlinearity	% of full scale		±0.3		%
Package Alignment Error			±1		Degrees
Interaxis Alignment Error			±0.1		Degrees
Cross Axis Sensitivity[1]			±1		%
SENSITIVITY (RATIOMETRIC)[2]	Each axis				
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	V_S = 3 V	270	300	330	mV/*g*
Sensitivity Change Due to Temperature[3]	V_S = 3 V		±0.015		%/°C
ZERO *g* BIAS LEVEL (RATIOMETRIC)	Each axis				
0 *g* Voltage at X_{OUT}, Y_{OUT}, Z_{OUT}	V_S = 3 V	1.2	1.5	1.8	V
0 *g* Offset vs. Temperature			±1		mg/°C
NOISE PERFORMANCE					
Noise Density X_{OUT}, Y_{OUT}			280		µg/√Hz rms
Noise Density Z_{OUT}			350		µg/√Hz rms
FREQUENCY RESPONSE[4]					
Bandwidth X_{OUT}, Y_{OUT}[5]	No external filter		1600		Hz
Bandwidth Z_{OUT}[5]	No external filter		550		Hz
R_{FILT} Tolerance			32 ± 15%		kΩ
Sensor Resonant Frequency			5.5		kHz
SELF TEST[6]					
Logic Input Low			+0.6		V
Logic Input High			+2.4		V
ST Actuation Current			+60		µA
Output Change at X_{OUT}	Self test 0 to 1		−150		mV
Output Change at Y_{OUT}	Self test 0 to 1		+150		mV
Output Change at Z_{OUT}	Self test 0 to 1		−60		mV
OUTPUT AMPLIFIER					
Output Swing Low	No load		0.1		V
Output Swing High	No load		2.8		V
POWER SUPPLY					
Operating Voltage Range		1.8		3.6	V
Supply Current	V_S = 3 V		320		µA
Turn-On Time[7]	No external filter		1		ms
TEMPERATURE					
Operating Temperature Range		−25		+70	°C

Fuente: ADXL330. [en línea] http://www.analog.com/static/imported-files/data_sheets/ADXL330.pdf. [Consulta: 10 de junio de 2013.]