Instituto Tecnológico de Tuxtla Gutiérrez

Proyecto de Residencia Profesional: "*Control de una silla de ruedas por medio de un dispositivo móvil Android***"**

Asesor: MA José Alberto Morales Mancilla

Hernández Cruz José Manuel

No. de Control: 07270577

Pérez Urquin Samuel Guadalupe

No. de Control: 07270579

Tuxtla Gutiérrez, Chiapas; 10 de Junio de 2013

ÍNDICE

RESÚMEN

El proyecto “Control de una silla de ruedas a través de un dispositivo móvil Android” pretende ayudar a personas que tienen problemas motrices debido a la discapacidad que tengan en una o más extremidades ayudándoles a desplazarse con una silla de ruedas controlada desde un teléfono móvil con Sistema Operativo Android.

El control de la silla será de manera inalámbrica, por lo que podrá ser manipulada tanto por quien use la silla, así como un tercero que esté a su cuidado en el caso de que quien padezca la discapacidad no pueda hacer uso del sistema.

1. INTRODUCCIÓN

[INEGI. Cuéntame… Población. [Ref. Marzo de 2013]. México. Disponible en Web: <http://cuentame.inegi.org.mx/poblacion/discapacidad.aspx?tema=P>]

En nuestro país existen, al año 2010, 5 millones 739 mil 270 personas con capacidades diferentes, lo cual representa un 5.1% de la población total. Existen diferentes tipos de discapacidades entre las que se encuentran: motriz (no poder caminar o moverse), de la vista, mentales, auditivas, comunicativas (hablar), de aprendizaje y auto-cuidado (que tiene limitaciones para atender por sí mismo su cuidado personal. De todas estas discapacidades es la primera (motriz) la que tiene los índices más altos (un 58% de la población con discapacidad)

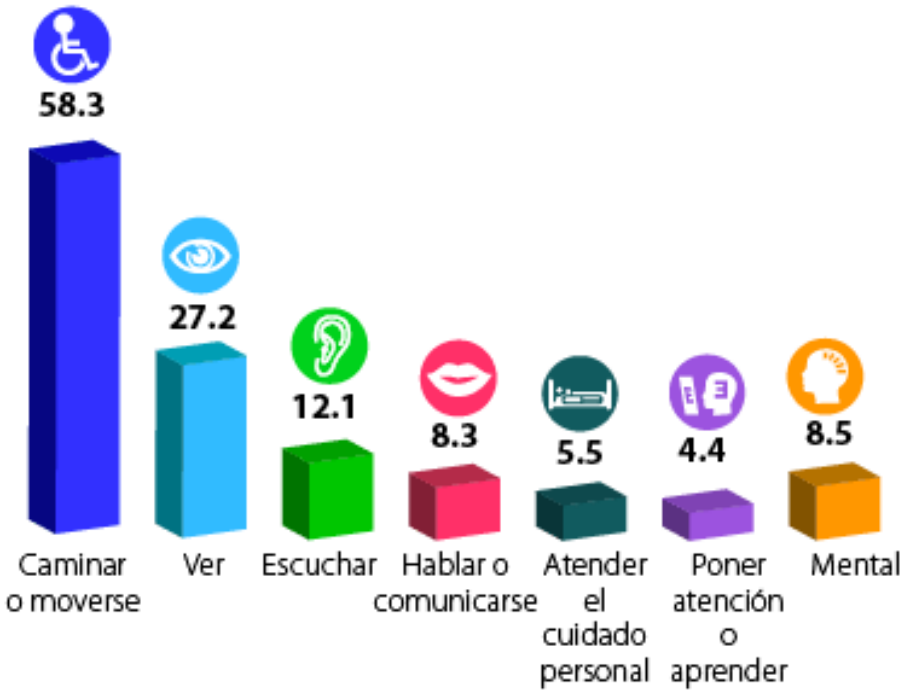

Figura 1.1 Porcentaje de la población con discapacidad según dificultad en la actividad (Año 2010)

Esta gráfica excede el 100% numérico y es debido a que hay personas que tienen más de una sola de las discapacidades mencionadas. La causa de la discapacidad varía y el INEGI lo clasifica en 4 grupos, puede ser por enfermedad, de nacimiento, accidentes o edad avanzada, siendo Enfermedad, la principal causa de discapacidades, seguida por edad avanzada, viniendo luego por nacimiento y por último accidentes. También pueden ser otros casos y algunos no especificados.

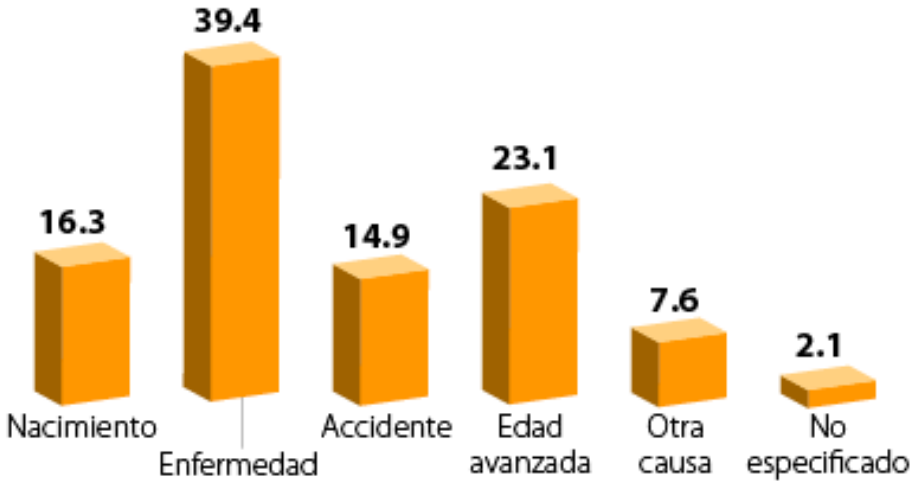

Figura 1.2. Porcentaje de la población con discapacidad según causa de la misma. (Año 2010)

Referente a la discapacidad motriz, que es dónde nos centramos para el desarrollo del proyecto hacemos referencia a las enfermedades conocidas como Artrogriposis Congénita Múltiple y Distrofia Muscular tipo Duchenne.

1.1. Antecedentes

[Pérez Martínez, Geovanny. *Sistema para el control de una silla de ruedas por medio de un dispositivo móvil para personas parapléjicas*. Ing. Eléctrica del Instituto Tecnológico de Tuxtla Gutiérrez. Tuxtla Gutiérrez, Chiapas, México. 17 de Diciembre de 2012. [Ref. Mayo de 2013]. Página 5-7]

Uno de los antecedentes más recientes sobre el que se tomaron las bases para el desarrollo de este proyecto, es uno similar controlado igualmente a través de un dispositivo móvil Android, con la diferencia de que el anterior es manejado a través de botones en lugar del sensor acelerómetro con el que este proyecto fue desarrollado.

[Montoto Paredes, Jesús Fernando; Gutiérrez Garrido, Antonio. *Silla de ruedas controlada por voz*. Ingeniería Mecatrónica de la Universidad Politécnica de Pachuca. Pachuca, Hidalgo. 08 de Noviembre de 2007. [Ref. Mayo de 2013]. Páginas 179-181]

Otro antecedente es una silla de ruedas controlada por voz hecha por alumnos de la Universidad Politécnica de Pachuca. Este sistema funciona comparando patrones contenidos en el sistema con los captados por el micrófono que está instalado en la silla de ruedas. A través de este reconocimiento de patrones de voz, parecido al que incluyen hoy día muchos aparatos electrónicos, la silla de ruedas se desplaza hacia dónde el usuario haya dado la orden.

1.2. Problemática

La principal dificultad de las personas con discapacidad motriz, es el desplazarse de un lugar a otro, ya que necesitan de una silla de ruedas para ese fin. Eso, hace que incluso el movimiento en su propia casa sea complicado. Se necesita que, en el lugar dónde vive, estudia y trabaja existan rampas que le permitan el acceso a esos lugares.

[Texas Scottish Rite Hospital. *Artrogriposis*. Patient Education. Dallas, Texas, Estados Unidos de América. [Ref. Marzo de 2013]. Páginas 1 y 2. Disponible en Web: <http://www.tsrhc.org/downloads/PDF/Amb-033a.pdf>]

Las personas que padecen de Duchenne y Artrogriposis Múltiple Congénita, nacen con estas enfermedades, por lo cual, conforme crecen estos padecimientos se agravan de manera gradual. Esto hace que necesiten de otra persona (ya sea los padres o un tercero) que los traslade porque al crecer van perdiendo la fortaleza en las extremidades. Esto hace que el acceso a espacios públicos como escuelas, oficinas, entre otros sea muy complicado ya que si tienen poca movilidad en los brazos, necesitan de una persona que los asista para poder desplazarse.

1.3. Estado del arte

[Wheelchair e-magazine. Historia de las Sillas de Ruedas. España. [Ref. Abril 2013]. Disponible en Web: <http://www.guiamovilidad.com/noticias/129-historia-de-las-sillas-de-ruedas.html>]

La primera representación conocida de una silla de ruedas, proveniente de la China, data del año 525 antes de NE. Mucho tiempo después, Felipe II de España, quien padecía de gota dolorosa, utilizó en 1595 el primer sillón de ruedas confortable, equipado con respaldo y reposapiés de inclinación variable. La propulsión de todos estos aparatos era llevada a cabo por otra persona.

En 1665, Farfler, un relojero alemán parapléjico, fabricó una especie de velocímano de tres ruedas accionado por manivelas que movilizaban mediante un engranaje la rueda delantera. Gracias a este aparato, por primera vez la persona minusválida pudo desplazarse de manera autónoma. En 1783, aparece la silla “Bath”, inventada por el fabricante John Dawson, en la ciudad inglesa de Bath, de dónde toma su nombre. No obstante, la silla “Bath” no era muy cómoda y durante el siguiente siglo fueron añadiéndole mejoras, pensando sobretodo en el confort del usuario, como respaldo y reposapiés ajustables. Entre 1867 y 1875, se siguieron añadiendo mejoras como los aros de propulsión y ruedas de goma aprovechando la invención de las bicicletas en esas épocas.

En 1900, se introdujeron las ruedas radiadas manuales y en 1916 se fabricó en Londres la primera silla de ruedas motorizada. A partir de la guerra de 1914-1918 se empezaron a construir las sillas de ruedas con tubos unidos mediante placas metálicas sobre las que se apoyaban los cojines.

En 1932, los estadounidenses Everest y Jennings concibieron una silla revolucionaria que ofrecía dos grandes ventajas: al ser poco voluminosa pasaba fácilmente por espacios estrechos, pero además era plegable, de manera que podía cargarse en el automóvil La silla de ruedas con propulsión eléctrica está actualmente en plena expansión. Sus últimas mejoras son el mando electrónico (en lugar del eléctrico), la ergonomía del proceso de desarmado en los modelos pequeños, las posiciones multifuncionales en los modelos más perfeccionados y el desarrollo de las interfaces de conducción. En el futuro se mejorarán probablemente los sistemas de asistencia a la conducción.

1.4. Sillas de ruedas controladas

[PowerChairsDirec.co.uk. *La silla de ruedas eléctrica: historia*. Power Chairs Direct. Cambourne, Cornwall, USA. [Ref. Junio de 2013]. Disponible en Web: <http://www.powerchairsdirect.co.uk/powerchair%20articles/powerchair-history.html>]

Sobre cuándo fue desarrollada la primera silla de ruedas eléctrica puede causar controversia. Canadá reclama haber desarrollado la primer silla de ruedas eléctrica en 1950, jactándose acerca de un trabajo llamado “Espíritu colaborativo que lleva a la invención de una silla de ruedas motorizada”.

Aunque eso sólo parece ser que las sillas de ruedas eléctricas se hayan empezado a usar desde antes. El hecho es que el registro que las primeras sillas de ruedas eléctricas datan de la época temprana del siglo XX. La primera silla de ruedas de este tipo parece haberse fabricado en 1912, cuando un motor de 1.75 caballos de fuerza fue añadido a un triciclo para inválidos.

Probablemente la primera silla de ruedas eléctrica que se desarrolló de manera comercial fue hecha en 1916, aunque era muy cara para las personas discapacitadas en ese entonces quienes decidieron mantenerse en el uso de sillas de ruedas manuales.

Sobre quien desarrolló la primera silla de ruedas motorizada aún sigue causando confusión, actualmente el crédito a menudo es dado a Ernest y Jennings, quienes fueron los primeros en manufacturar las sillas de ruedas eléctricas en varias escalas con producción en masa la cual comenzó en 1956.

[Roach, John. Silla de ruedas robótica convierte las ruedas en piernas para escalar. NBC News. 15 de Octubre de 2012. [Ref. Junio de 2013]. Disponible en Web: <http://www.nbcnews.com/technology/robotic-wheelchair-turns-wheels-legs-climbs-steps-1C6443238>

En Japón, el Chiba Institute of Technology creó una silla de ruedas escaladora, cuyas ruedas se comportan como piernas que pueden dar pasos curvos y subir

escalones de baja altura. La tecnología desarrollada podría ser revolucionaria para personas con movilidad limitada.

Esta silla de ruedas está bajo el desarrollo del Instituto Tecnológico de Chiba en Japón. Luce como una silla de ruedas común controlada por un joystick que se mueve con el giro de sus cuatro ruedas, pero cuando se encuentra con algún obstáculo en el camino, un conjunto de sensores mide el obstáculo y la silla se ajusta de acuerdo a las maniobras y sus ruedas como si fuesen piernas se posicionan sobre este para evadirlo. Aunque es una tecnología bastante innovadora, su costo asciende a los 25,000 dólares.

[El Universal TV Computación. *Desarrollan silla de ruedas que puede ser controlada por ondas cerebrales*. El Universal. México, D. F. 29 de Septiembre de 2012. [Ref. Junio de 2013]. Disponible en Web: <http://www.eluniversaltv.com.mx/detalle.php?d=32616>

Toyota Motor Corp., anunció la creación de una silla de ruedas que detecta las ondas cerebrales de una persona sentada en ella y moverla sin hacer el más mínimo movimiento muscular o dar órdenes por voz.

El usuario se coloca un gorro que lee las ondas cerebrales y las transmite a un encefalograma que interpreta las órdenes, el único movimiento físico que tiene que realizar el usuario es para el frenado, lo que tiene qué hacer es inflar las mejillas para que un sensor detecte este movimiento y detenga el desplazamiento.

Este proyecto aún se encuentra en desarrollo por la compañía.

Silla de ruedas/Características	**Detección de obstáculos**	**Velocidad de movimiento**	**Costo**
Silla de ruedas controlada por botones y	No cuenta con detección de obstáculos	Rápido	Alrededor de 1200 dólares

dispositivo móvil			
Silla de ruedas escaladora	Sí cuenta con detección de obstáculos	Lenta/Medio (depende del tamaño del obstáculo)	25000 dólares
Silla de ruedas controlada por ondas cerebrales	En desarrollo	Rápido	En estimación
Silla de ruedas controlada por acelerómetro y dispositivo móvil	No cuenta con detección de obstáculos	Medio	Alrededor de 1200 dólares

Tabla 1. Tabla comparativa de las sillas de ruedas presentadas en este documento

2. JUSTIFICACIÓN

El proyecto ayudará a las personas que padecen de discapacidad motriz a proporcionarles acceso a los espacios públicos de una manera más cómoda e independiente ya que no necesitan de un tercero que los apoye o hacer mucho esfuerzo impulsando una silla de ruedas manual.

2.1. Alcances

Un punto más que sustenta el proyecto, es que las sillas eléctricas convencionales en el mercado, puede elevar su precio hasta los 90,000 pesos dependiendo de las características del modelo, además, con cada elemento agregado la silla es mucho más pesada. El prototipo planteado es más liviano y de bajo costo estimado en 15,000 pesos.

2.2. Limitaciones

Al ser una silla de ruedas, se necesita que donde el usuario la utilice haya rampas para discapacitados.

La silla de ruedas no tiene sensor para obstáculos, por lo que si el usuario no detiene la silla, podría chocar contra una pared, banquetas u otro obstáculo en el camino.

3. OBJETIVO GENERAL

Ayudar a una persona con discapacidad motriz a desplazarse utilizando una silla de ruedas controlada a través de un dispositivo móvil con SO Android.

3.1. Objetivos específicos

- Controlar una silla de ruedas a través del sensor acelerómetro contenido en los dispositivos móviles con SO Android.
- Frenar la silla de ruedas con el sensor del dispositivo.
- Mover la silla de ruedas hacia adelante y hacia atrás usando el sensor acelerómetro del dispositivo móvil
- Desplazar la silla de ruedas hacia la derecha e izquierda usando el sensor acelerómetro del dispositivo móvil

4. ÁREA DE TRABAJO

Para lograr la Residencia Profesional, se trabajó en conjunto con la Unidad de Orientación al Público de educación especial (UOP).

El programa Centros de Atención Múltiple de la UOP está planteado para brindar educación inicial y básica (preescolar y primaria); así como formación para el trabajo, a la población escolar que presenta discapacidad y necesidades educativas especiales. La atención en los CAM´S tiene cierto

carácter transitorio, dado que se espera la integración de los alumnos a escuelas regulares o al ámbito laboral competitivo. En todos, se implementan actividades que faciliten a los niños su desarrollo cognitivo, psicomotor, lingüístico y personal. Así como también, brinda atención a los padres de familia con apoyo psicológico y de orientación educativa, para involucrarlos en el proceso de enseñanza-aprendizaje de sus hijos.

4.1 Funciones de cada departamento

1. **Dirección general**. Asumir la responsabilidad inmediata de que la institución en general opere de forma adecuada.
2. **Departamento de planeación y seguimiento operativo**. Planea y evalúa las actividades para la atención educativa, así como el control sobre los recursos presupuestales requeridos para la operación del Instituto.
3. **Departamento de atención de grupos étnicos**. Organiza a los grupos étnicos existentes en el estado de Chiapas para evaluar el proceso de la educación que les brindara.
4. **Departamento de servicios educativos**. El Departamento de Servicios Educativos planea, organiza, dirige y evalúa los procesos de formación de todas las figuras que trabajan en el IDEA, así como las formas y estrategias de atención a educandos, de acuerdo con los lineamientos, modelos, materiales y contenidos académicos.
5. **Departamento de administración**. Proporciona recursos suficientes para la operación, previa autorización del Departamento de Planeación y la existencia o disponibilidad de los mismos, como son materiales educativos, didácticos, de apoyo, el pago de gratificaciones, etc.
6. **Departamento de acreditación**. Proporciona los servicios de acreditación y certificación a los educandos, lleva el control sobre el avance educativo de éstos.
7. **Departamento de informática**. Procesa la información cuantitativa y los procesos automatizados de control del avance académico, acreditación y certificación de educandos.

4.2. Organigrama

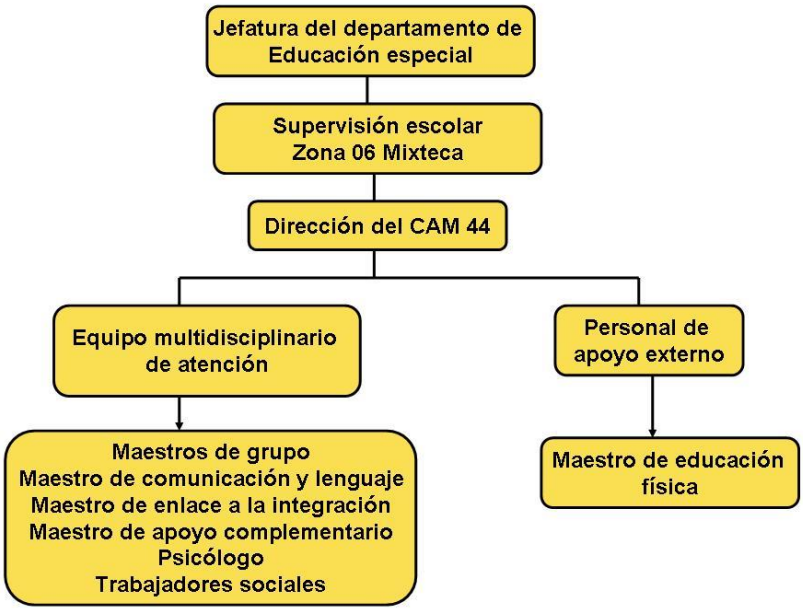

Figura 4.1. Oranigrama de la UOP

4.3. Ubicación

El CAM dónde se tomó la problemática para el desarrollo del proyecto, se encuentra ubicado en la Prolongación Norte de la Avenida Rosa del Poniente S/N en la colonia INFONAVIT Rosario en la ciudad de Tuxtla Gutiérrez, Chiapas.

5. FUNDAMENTO TEÓRICO

A continuación describiremos las herramientas y tecnologías de las que nos valdremos para la realización del proyecto de Residencia descrito en este documento.

5.1. Marco Teórico Conceptual

5.1.1. Bluetooth

[FreeBSD Handbook. *Chapter 32 Advandec Networking*. [Referencia Abril 2013]. Disponible en Web: <http://www.freebsd.org/doc/en/books/handbook/network-bluetooth.html>]

Bluetooth es una tecnología inalámbrica usada para la creación de redes personales (WPAN, *Wirelesss Personal Area Network*) operando en la banda libre de 2.4Ghz en un rango de 10 metros. Las redes son usualmente ad-hoc, formadas desde dispositivos portátiles como los teléfonos celulares y lap-tops. A diferencia de otras tecnologías inalámbricas populares como el Wi-Fi, Bluetooth ofrece un alto nivel de servicios de perfiles, los cuales describiremos más adelante.

5.1.1.1. Arquitectura Bluetooth

[Desarrollo de Aplicaciones para dispositivos móviles. *Apéndice A, Protocolo Bluetooth*. ANTS Research Group. Universidad de Murcia, España. [Referencia Abril 2013]. Páginas 169-174. Disponible en Web: <http://ants.dif.um.es/~felixgm/docencia/android/resources/ApendicesBibliografia.pdf>]

La unidad básica de un sistema Bluetooth es una picored, que costa de un nodo maestro y hasta siete nodos esclavos activos a una distancia de 10 metros. Varias picoredes pueden conectarse a través de nodos puente formando lo que se conoce como una *scattarnet* (red dispersa).

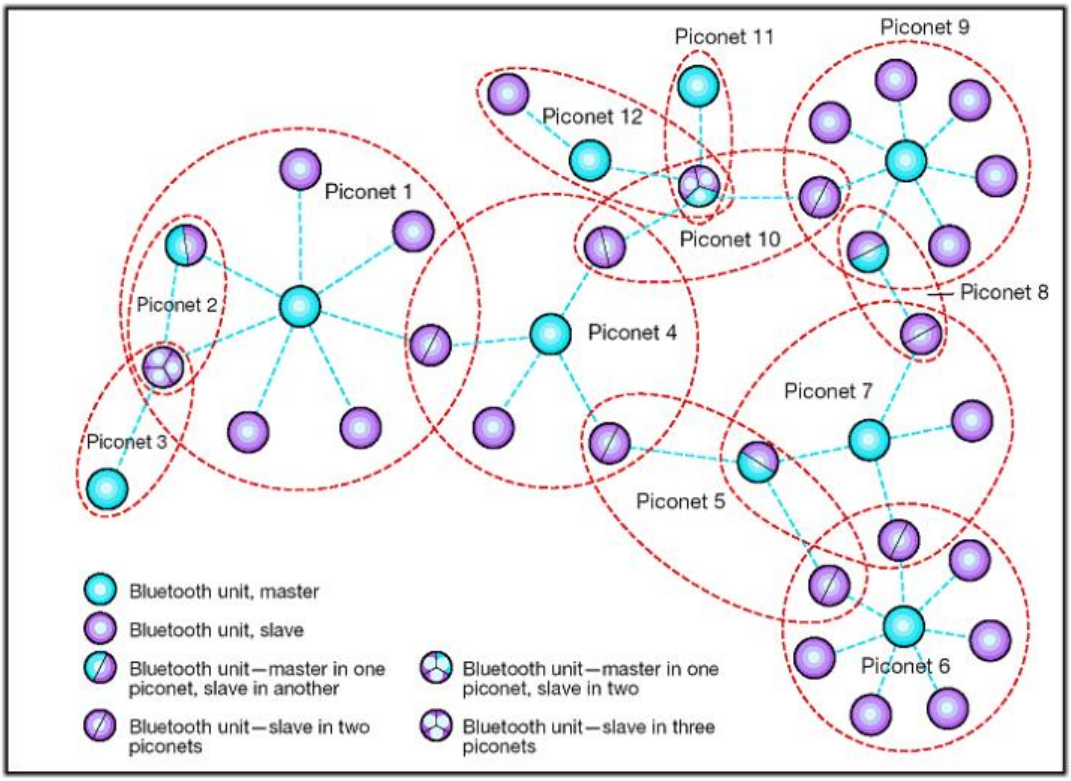

Figura 5.1. *Scatternet* formada por 12 picoredes

Además de los siete nodos esclavos activos en una picored, puede haber hasta 255 nodos estacionarios en la red. Éstos son dispositivos que el maestro ha cambiado a un estado de bajo consumo para reducir el desgate innecesario de sus baterías. Lo único que un dispositivo estacionario puede hacer es responder a una señal de activación por parte del maestro.

5.1.1.2. Pila de Protocolos Bluetooth

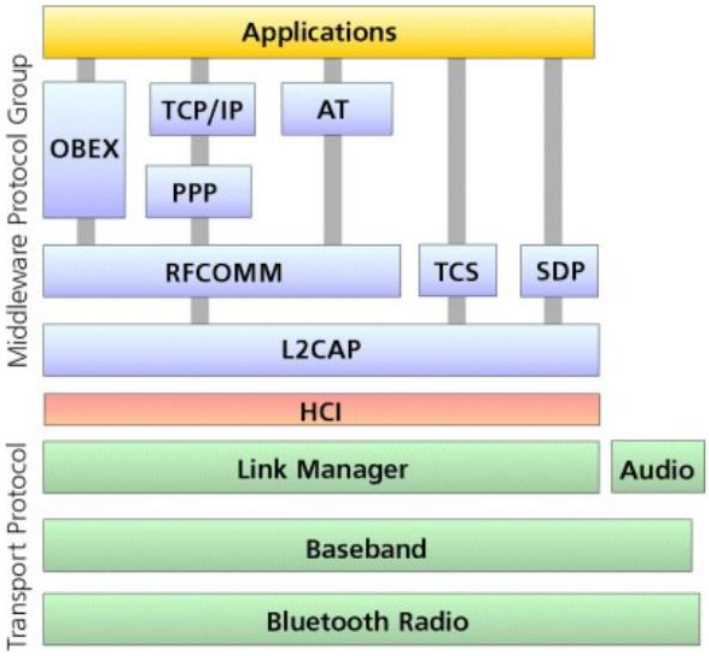

Figura 5.2. Pila de protocolos de Bluetooth

En esta imagen se puede apreciar la pila de protocolos de Bluetooth y a continuación se explicará el funcionamiento de cada una de estas capas.

a) **La capa de radio**. Esta capa define los requisitos para un transmisor-receptor de radio Bluetooth, el cual opera en la banda de 2.4GHz. Define los niveles de sensibilidad de dicho transmisor-receptor, establece los requisitos para utilizar frecuencias del espectro expandido y clasifica a los dispositivos Bluetooth en tres clases.

Clase	Potencia máxima permitida		Potencia mínima permitida		Distancia
	mW	dBm	mW	dBm	
Clase 1	100 mW	20 dBm	1 mW	0 dBm	100 metros
Clase 2	2'5 mW	4 dBm	0'25 mW	-6 dBm	10 metros
Clase 3	1 mW	0 dBm	N/A	N/A	1 metros

Tabla 2. Clases de dispositivos Bluetooth

Esta capa transfiere los bits del maestro al esclavo y viceversa. Es un sistema de baja potencia con un rango entre 1 y 100 metros. La banda se divide en 79 canales de 1MHz cada uno. La modulación es por división de

frecuencia, con 1 bit por Hz, lo cual da una tasa de datos aproximada de 1Mbps, pero gran parte de este espectro lo consume la sobrecarga.

Para asignar los canales de manera equitativa, el espectro da saltos de frecuencia, se utiliza a 1600 saltos por segundo y un tiempo de permanencia de 625µseg. Todos los nodos de una picored saltan de manera simultánea, y el maestro establece la secuencia de salto.

Debido a que tanto el 802.11 como Bluetooth operan en la banda ISM de 2.4GHz en los mismos 79 canales, interfieren entre sí. Puesto que Bluetooth salta mucho más rápido que 802.11, es más probable que un dispositivo Bluetooth dañe las transmisiones del 802.11 que en caso contrario.

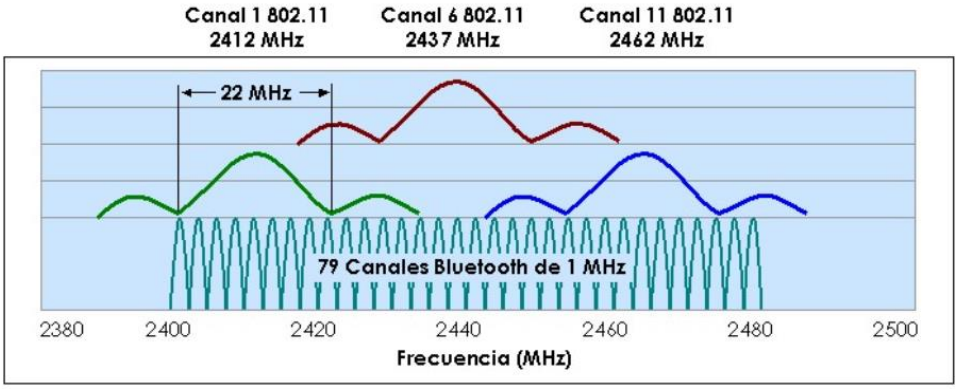

Figura 5.3. Coexistencia de Bluetooth y 802.11 en la banda de los 2.4GHz

b) **La capa de banda base**. Esta capa, representa la capa física de Bluetooth. Se usa como controladora de enlace, la cual trabaja con el *link manager* para llevar a cabo operaciones tales como la creación de conexiones de enlace con otros dispositivos.

   Esta capa es lo más parecido a una subcapa MAC. Esta convierte el flujo de bis puros en tramas. En la fórmula más sencilla, el maestro de cada picored define una serie de ranuras de tiempo de 625µseg y las transmisiones del maestro en las ranuras pares, y las de los esclavos en las impares. Esta es la tradicional multiplexación por división de tiempo, en la cual el maestro acapara la mitad de las ranuras y los esclavos

comparten la otra mitad. Las tramas pueden tener 1, 3 o 5 ranuras de longitud. Cada trama se transmite por un canal lógico llamado enlace entre el maestro y un esclavo. Hay dos tipos de enlaces

a. **ACL**. Se utiliza para datos conmutados en paquetes disponibles a intervalos irregulares. Estos datos provienen de la capa L2CAP en el nodo emisor y se entregan en la capa L2CAP en el nodo receptor. No hay garantías de entrega del tráfico de ACL. Las tramas se pueden perder y tienen qué retrasmitirse. Un esclavo sólo puede tener un enlace ACL con su maestro.
b. **SCO**. Se usa para datos en tiempo real, como ocurre en las conexiones telefónicas. A este tipo de canal se le asigna una ranura fija en cada dirección. Las tramas que se envían a través de SCO nunca se retransmiten. En vez de ello se puede usar la corrección de errores hacia adelante para conferir una alta fiabilidad al sistema. Un esclavo puede establecer hasta enlaces SCO con su maestro. Cada enlace de este tipo puede transmitir un canal de audio PCM de 64000 bps.

c) **Link Manager**. Es el administrador de enlaces o *Link Manager*, se encarga de establecer canales lógicos entre dispositivos, incluyendo administración de energía, autenticación y calidad de servicio.
d) **Host Controller Interface (HCI)**. El HCI permite el acceso mediante línea de comandos a la capa de banda base y al LMP para controlar y recibir información acerca del estado. Se compone de tres partes:
    a. **El firmware HCI**. También conocido como el programa oficial del fabricante, actualmente forma parte del hardware del Bluetooth.
    b. **El controlador HCI**. Es un driver que se encuentra en el software del dispositivo Bluetooth.
    c. **El Host Controller Transport Layer**. Este conecta el firmware con el driver.

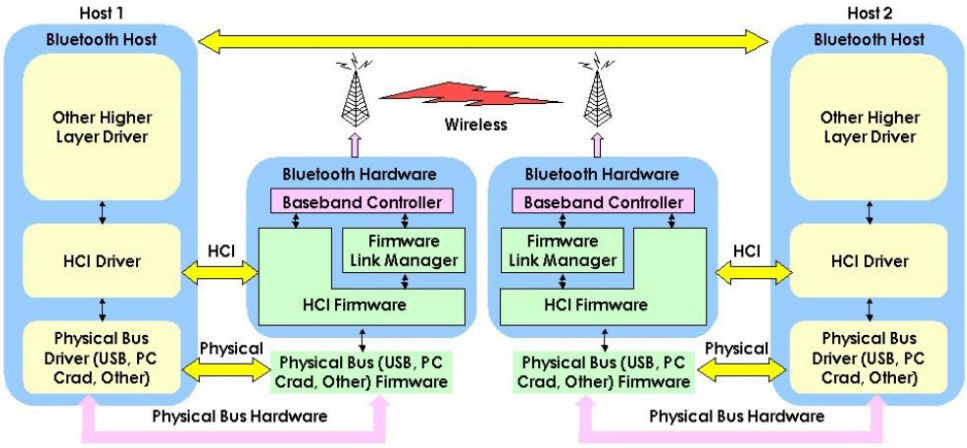

Figura 5.4. Host Controller Interface

e) **Protocolo de adaptación y control de enlaces lógicos L2CAP**. El protocolo L2CAP aísla a las capas superiores de los detalles de transmisión. Es análogo a la subcapa LLC del estándar 802, pero difiere de ésta en el aspecto técnico. Proporciona servicios orientados y no orientados a la conexión a las capas superiores. Este protocolo permite a los protocolos de alto nivel y a las aplicaciones enviar y recibir paquetes de datos de hasta 64Kb. Buena parte de su tiempo la dedica a la segmentación y reensamblado.

La capa L2CAP tiene tres funciones principales:

a. Acepta paquetes de hasta 64Kb provenientes de las capas superiores y los divide en tramas para transmitirlos. Las tramas se reensamblan nuevamente en paquetes en el otro extremo.
b. Maneja la multiplexación desmultiplexación de múltiples fuentes de paquetes. Cuando se reensambla un paquete, la capa L2CAP determina qué protocolo de las capas superiores lo manejará. Por ejemplo la capa RFCOMM o el de telefonía.
c. Se encarga de la calidad del servicio, tanto al establecer los enlaces como durante la operación normal. Asimismo, durante el establecimiento de los enlaces se negocia el tamaño máximo de carga útil permitido, para evitar que un dispositivo que envíe paquetes grandes sature a uno que recibe paquetes pequeños.

f) **REFCOMM**. RFCOMM es el protocolo que emula el puerto serie estándar RS232 de los ordenadores para la conexión de teclados, ratones y módems, entre otros dispositivos. Su propósito es que dichos dispositivos no tengan por qué saber nada acerca de Bluetooth.

g) **TCS y SDP**. El protocolo de telefonía TCS es un protocolo de tiempo real y está destinado a los tres perfiles orientados a voz (HFP, HSP e ICP). También se encarga del establecimiento y terminación de llamadas. Por su parte, el protocolo de descubrimiento de servicios (SDP) se emplea para la detección automática de dispositivos dentro de la red, así como servicios ofrecidos por dichos dispositivos.

5.1.1.3. Estructura de la trama de Bluetooth

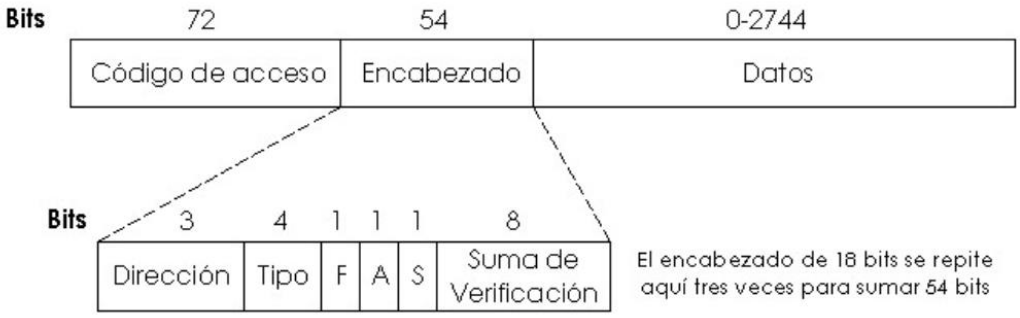

Figura 5.5. Estructura de la trama de Bluetooth

En la figura anterior se puede apreciar el formato de una trama de Bluetooth. Las tramas comienzan con un código de acceso, el cual identifica al maestro de la picored y cuyo propósito es que los esclavos que se encuentren en el rango de alcance de dos maestros sepan qué tráfico está destinado a ellos.

Después de este código de acceso, viene un encabezado de 54bits que contiene campos comunes de la subcapa MAC. Y por último está el campo de datos, de hasta 2744bits (para la transmisión de 5 ranuras). Para una sola ranura de tiempo, el formato es el mismo excepto que el campo de datos es de 240bits.

Describiendo un poco más a detalle el encabezado, el campo *Dirección* identifica a cuál de los ocho dispositivos activos está destinada la trama. El

campo *Tipo* indica el tipo de trama (ACL, SCO, de sondeo o nula), el tipo de corrección de errores que se utiliza en el campo de datos y cuántas ranuras de longitud tiene la trama.

Un esclavo establece el bit *F* (de flujo) cuando su búfer está lleno y no puede recibir más datos. Ésta es una forma primitiva de control de flujo. El bit *A* (de confirmación de recepción o *Acknowledgement*) se utiliza para incorporar un ACK en la trama. Por último, el bit *S* (de secuencia) sirve para numerar las tramas con el propósito de detectar retransmisiones. El protocolo es de parada y espera, por lo que 1 bit es suficiente.

A continuación se encuentra el encabezado *Suma de verificación de 8* bits. Todo el encabezado de 18 bits se repite tres veces para formar el encabezado de 54bits. En el receptor se comprueban las tres copias de cada bit y si coinciden, se acepta. De lo contrario se impone la opinión de la mayoría.

5.1.2. Acelerómetro

[Carletti, Eduardo J. *Acelerómetros, sensores de vibración. Descripción y funcionamiento*. Sensores. Robots Argentina. Buenos Aires, Argentina. [Ref. Mayo de 2013]. Disponible en Web: <http://robots-argentina.com.ar/Sensores_acelerometros.htm>]

Un acelerómetro es un dispositivo que permite medir el movimiento y las vibraciones a las que está sometido el sistema al que está unido en su modo de medición dinámico, y la inclinación (con respecto a la gravedad) en su modo estático.

De los antiguos acelerómetros mecánicos de tamaños grandes y dificultosos de construir, porque incluían imanes, resortes y bobinas (en algunos modelos), se ha pasado en esta época a dispositivos integrados, con los elementos sensibles creados sobre los propios microcircuitos.

Uno de los acelerómetros integrados más conocidos es el ADXL202, muy pequeño, versátil y de costo accesible. Este es un acelerómetro de bajo

consumo y salida digital, integrado en un chip monolítico, mide aceleraciones hasta una escala máxima de ± 2g, soporta golpes de hasta 1000g. Puede medir aceleración dinámica (como por ejemplo la vibración) y también la aceleración estática, como por ejemplo la atracción de la gravedad.

Este circuito integrado tiene salidas digitales en forma de pulsos repetidos cuyo ancho varía en relación con la medición. Estas salidas en forma de pulsos se pueden medir con microcontroladores sin necesidad de contar con una entrada para la conversión analógica/digital. El ritmo de repetición del pulso es ajustable de 0,5 a 10ms por medio de un resistor. Un ciclo de relación 50% significa una aceleración de 9g.

El ruido de la señal es muy bajo, lo que permite hacer mediciones menores a 2mg, a una frecuencia de 50Hz. El ancho de banda de respuesta se puede determinar por medio de capacitares de filtro conectados en ambos circuitos X e Y.

5.1.3. Eclipse Índigo

[Eclipse Org. http://www.eclipse.org/org/]
Eclipse Índigo es un compilador orientado a Java de distribución libre de la serie Eclipse. Se pueden compilar diversas aplicaciones para Java, incluidas aquellas destinadas para funcionar bajo un Sistema Operativo Android contenido en muchos teléfonos móviles actuales.

5.1.4. Arduino Uno

El Arduino Uno es una placa de microcontrolador basada en el ATmega238 (*datasheet*). Tiene 14 pines que pueden ser usados como entrada o salida (de entre los cuales 6 pueden ser usados como salidas PWM), 6 entradas análogas, resonador cerámico de 16MHz, conexión USB, jack para alimentación de corriente, un “encabezado” ICSP (*In-Circuit Serial Programming*, Programación en Circuito Serial) y un botón para reset. Contiene todo lo necesario para el soporte de microcontrolador; simplemente hay que conectarlo a una entrada

USB a una computadora o encenderlo con el adaptador AC/DC o una batería para inicializarlo.

El Uno se diferencia de todos las otras placas que le preceden en que no usa el chip de driver FTDI USB a serial. En su lugar, su característica es el ATmega16U2 programado como un conversor USB a serial.

5.1.4.1. Características del Arduino Uno

- Microcontrolador ATmega328
- Voltaje de operación a 5V
- Voltaje de entrada recomendado de 7-12V
- Voltaje de entrada límite de 6-20V
- 14 pines digitales de entrada/salida (de los cuales 6 proveen salida PWM)
- 6 pines de entrada análoga
- 40mA de CD por pin de entrada/salida
- 32kb de memoria flash de los cuales 0.5kb son usados por el bootloader
- 2kb para la SRAM
- 1kb de memoria EEPROM
- Velocidad de reloj a 16MHz

5.1.4.2. Comunicación del Arduino Uno

El Arduino uno tiene muchas facilidades para comunicarse con una computadora, otros Arduinos u otros microcontroladores. El ATmega328 provee una comunicación serial UART TTL a 5V, la cual está disponible en los pines digitales 0 (RX) y 1 (TX). Un ATmega16U2 en la placa canaliza esta comunicación serial sobre el USB y aparece como un puerto COM virtual en el software de la computadora. El firmware del ‘16U2 usa los estándares de los drivers COM de USB y no se necesita un driver externo. Sin embargo, bajo Windows es necesario incluir un archivo .inf.

El software Arduino incluye un monitor que permite que un simple dato de texto sea enviado hacia y desde la placa Arduino. Los LEDs de los pines RX y TX se encenderán cuando haya datos en transmisión a través del chip USB a serial y la conexión USB a la computadora

5.2. Marco Teórico Específico

5.2.1. Artrogriposis Múltiple Congénita

[Álamos Asociación. *¿Qué es la Artrogriposis Múltiple Congénita?* Asociación de Discapacidades Físicas y Sensoriales. Alcalá La Real Jaén, Andalucía, España. [Ref. Marzo de 2013]. Páginas 1 y 2. Disponible en Web: <http://asociacionalamos.es/img/enfermedades/40artrogriposis.pdf>]

La Artrogriposis Múltiple Congénita es un síndrome caracterizado por las contracturas congénitas no progresivas de muchas articulaciones. Esta característica presente en más de 200 síndromes no es el síntoma principal, sino uno más dentro de un desorden o enfermedad más generalizada que a menudo se hereda. En algunos de estos síndromes las anomalías (cardíacas, renales, pulmonares, etcétera) son a veces tan graves que hacen la vida casi imposible.

La causa de la enfermedad puede deberse a varios factores. Una tercera parte de los casos registrados son por herencia genética, aunque independientemente de si es heredada o no, esta enfermedad se desarrolla en el útero mientras el feto se desarrolla.

[Texas Scottish Rite Hospital. *Artrogriposis*. Patient Education. Dallas, Texas, Estados Unidos de América. [Ref. Marzo de 2013]. Páginas 1 y 2. Disponible en Web: <http://www.tsrhc.org/downloads/PDF/Amb-033a.pdf>]

Aunque se sabe que esto se desarrolla durante el embarazo, en la mayoría de los casos, esta enfermedad no puede ser identificada. Las articulaciones o coyunturas que carecen de movilidad antes del nacimiento pueden resultar en contracturas de las mismas. Cuando las articulaciones o coyunturas no tienen

movimiento por un período de tiempo, se desarrolla un tejido conectivo extra. Este tejido fija la articulación o coyuntura en una posición rígida o cerrada. Esto también ocasiona que los tendones que conectan estos mismos no se estiren a su medida normal, haciendo el difícil el movimiento de las articulaciones.

Las causas para la limitación del movimiento de articulaciones son:

- Los músculos no se desarrollan apropiadamente
- Trastornos musculares
- Fiebre durante el embarazo y virus que podrían dañar las células que transmiten los impulsos nerviosos
- Disminución de la cantidad de líquido amniótico
- El Sistema Nervioso Central y la médula espinal no se formaron correctamente
- Los tendones, huesos, articulaciones o coyunturas o el revestimiento de estas pueden desarrollarse anormalmente
- Una causa genética en el 30% de los casos
    - Algunos modelos genéticos han sido identificados, pero la mayoría de los casos son raros y el número de recurrencias varía con el tipo de trastorno genético.

5.2.2. Distrofia Muscular

[Institutos Nacionales de la Salud (NIH). *MedlinePlus, Información de salud para usted*. Biblioteca Nacional de Medicina de Estados Unidos de América. Bethesda, Meryland, Estados Unidos de América. [Ref. Marzo de 2013]. Disponible en Web: <http://www.nlm.nih.gov/medlineplus/spanish/ency/article/001190.htm>]

La Distrofia Muscular es una enfermedad hereditaria que puede afectar a un grupo específico de músculos o todos a la vez como los que están alrededor de la pelvis, los hombros o la cara. La distrofia Muscular puede afectar a adultos, pero las formas severas tienden a ocurrir en la primera infancia. Los tipos de síntomas varían dependiendo el tipo de Distrofia. En este trabajo, hablaremos de la Distrofia tipo Duchenne.

5.2.2.1. Distrofia Muscular Tipo Duchenne

[Baumgarther Manfred. *Distrofia Muscular de Duchenne*. Revista Médica de Costa Rica y Centroamérica. Traducción: Argüello Ruiz, Daniel. Publicación trimestral. San José, Costa Rica, Centroamérica. [Ref. Marzo de 2013]. Página 316 ISSN: 0034-9909.]

La Distrofia Muscular de Tipo Duchenne es la más frecuente de la niñez. Es un desorden de carácter hereditario recesivo ligado al cromosoma X, caracterizado por la debilidad muscular rápidamente progresivo, la cual empieza por los músculos de la pelvis y proximales de las piernas y luego afecta a todo el cuerpo, con un pronóstico de vida de no mayor a tres décadas.

Es una enfermedad hereditaria recesiva ligada al cromosoma del sexo. Afecta principalmente a hombres en relación a 1 por cada 3000 o 4000 nacidos vivos. Esta enfermedad es causada por una variedad de mutaciones, estas mutaciones conllevan a la ausencia total o parcial de distrofina, una proteína que forma parte del complejo disfrofina-glicoproteína, componente esencial del músculo esquelético.

Los síntomas por lo general aparecen a los 6 años de edad, pero también pueden darse desde la infancia temprana. Hay debilidad muscular en la pelvis y piernas, la cual se asocia a pérdida de masa muscular. La debilidad muscular también se presenta en los brazos, cuello y otras áreas, pero no tan severamente ni tan temprano con en la mitad inferior del cuerpo. Inicialmente los músculos de la pantorrilla se agrandan, los cuáles finalmente son remplazados por grasa y tejido conectivo. También se presentan contracturas musculares principalmente en los talones y piernas, produciendo incapacidad de utilizar los músculos debido al acortamiento de las fibras musculares y a la fibrosis del tejido conjuntivo. Hacia la edad de 10 años se requieren de prótesis para poder caminar y a la edad de 12 años, la mayoría de los pacientes están confinados a una silla de ruedas.

6. MODELO CONCEPTUAL

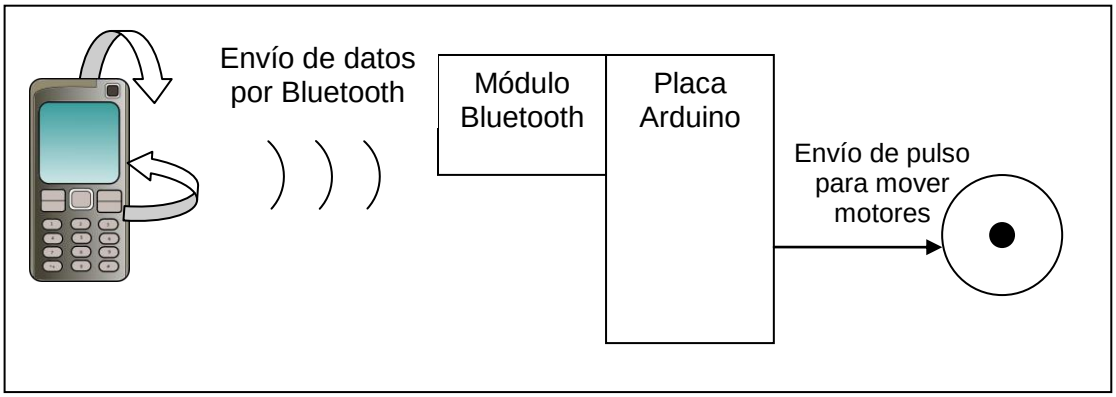

Figura 6.1. Modelo conceptual del proyecto

En la imagen anterior se puede apreciar el modelo conceptual básico del proyecto. Comenzando del lado izquierdo, tenemos el teléfono móvil, el cual se hará girar hacia la derecha, izquierda, adelante o atrás para que los valores del sensor acelerómetro se modifiquen y estos al llegar a un determinado valor (ya sea 4 para adelante o -4 hacia atrás) enviarán el dato numérico hacia el módulo Bluetooth que está conectado a la placa Arduino para su interpretación y enviarle el pulso requerido a los motores para su movimiento.

Para este módulo del proyecto, sólo se enfoca en la parte de comunicación. La sección de potencia (cómo interpreta Arduino los datos y de qué manera envía los pulsos a los motores) se describirán en otro trabajo.

6.1. Procedimiento de comunicación

El proceso para poder controlar la silla de ruedas por medio del dispositivo android se da a través de dos medios. Por botones y por acelerómetro.

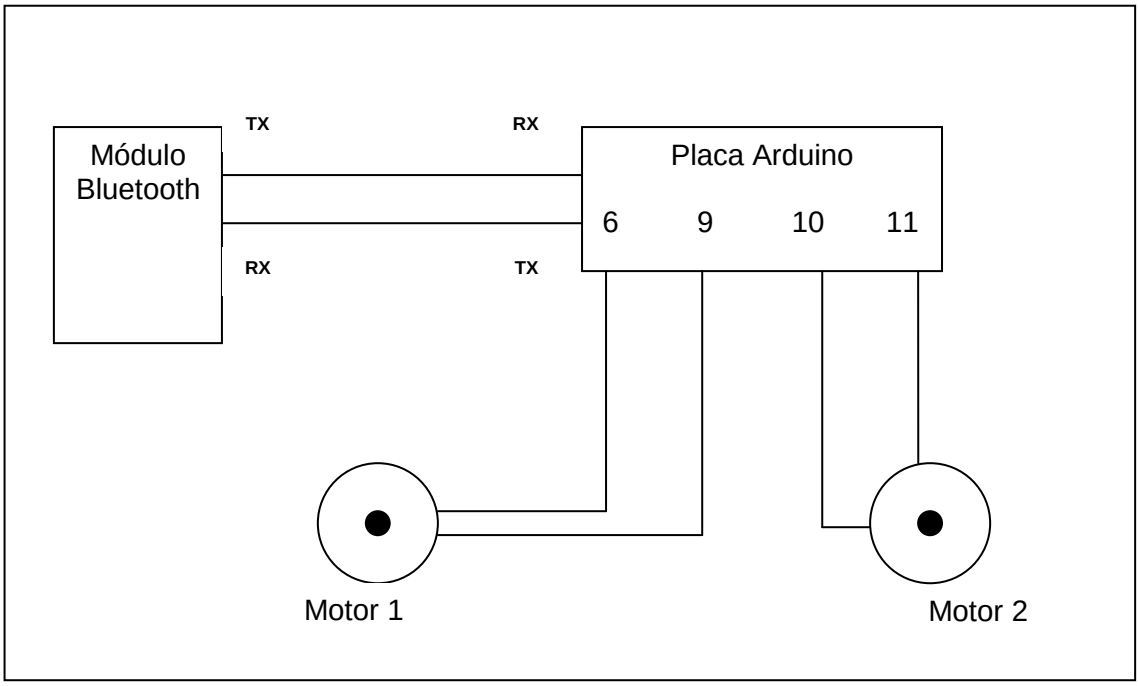

Figura 6.2. Diagrama unificar de conexiones

Como podemos apreciar en la Figura 6.2, el módulo Bluetooth recibe la señal de manera inalámbrica desde el teléfono móvil (Figura 6.1), a través de las terminales TX y RX, el módulo Bluetooth envía los datos hacia la placa Arduino para que se lean e interpreten y a su vez, la placa Arduino envía los pulsos hacia los motores a través de los puertos 6 y 9 para el Motor1, y 10 y 11 para el Motor 2 respectivamente.

A continuación explicaremos la manera en que las actividades del programa para Android ejecutan las instrucciones para el envío de los datos vía Bluetooth.

6.1.1. Botones

El sistema que se desarrollo esta al pendiente de que botones se presionan para saber que dato enviará al modulo arduino para que este envie la acción hacia los motores. Cuando en el sistema se presiona el botón adelante, este envía un numero 1 por medio del bluetooth hacia el arduino, para el botón atrás envía el numero 2, para el botón izquierda envía el numero 4 , para el botón derecha envía el numero 5 y para stop envía el numero 0.

El modulo arduino recibe cualquier de estos valores y analiza cual es, y dependiendo de esto envía pulsos a través de sus puertos de salida hacia los motores, los cuales, al recibir estos pulsos empiezan a girar. Los puertos digitales de salida del modulo arduino son: 6, 9, 10 y 11.

6.1.2. Acelerómetro

En este caso el sistema esta pendiente de la inclinación del teléfono, y cuando llega a un valor de inclinación, efectua el envio de un dato dependiendo sobre que eje se hizo la inclinación y que valor de inclinación alcanzo. Los valores leidos para determinar un movimiento en la silla son: para adelante el valor de inclinación sobre el eje y tiene que ser mayor o igual a -3, para atrás el valor sobre el eje y tiene que ser igual o mayor a 3, para la izquierda el valor en el eje x tiene que ser igual o mayor a 3, para la derecha el valor del eje x tiene que ser mayor o igual a -3 y para parar el movimiento los valores tanto en el eje x y en el eje y tienen que ser menores a 3.

Una vez que se detecto un valor de inclinación y se ha detectado la dirección que lleva esta inclinación, se procede a enviar los mismos datos que se envían en la modalidad de control por medio de botones hacia el modulo arduino.
Al igual que en la modalidad de control por medio de botones, el arduino interpreta el dato enviado y hace el envio de pulsos hacia los motores para hacer que muevan en la dirección seleccionada.

7. PROCEDIMIENTO Y PRUEBAS

A continuación describiremos los procedimientos y las pruebas que obtuvimos a lo lardo del desarrollo de este proyecto. Como primera instancia, se empezó con el desarrollo de la interfaz como se muestra en la figura siguiente. Los botones son para el arranque del sistema y abajo se aprecian unos cuadros de texto cuyo contenido cambia con la inclinación del teléfono.

7.1. Casos de uso

Para desarrollar la aplicación, recurrimos a los diagramas del Lenguaje de Modelado Universal (UML por sus siglas en inglés) para tener una guía sobre la construcción del software y como primer paso se estructuró el diagrama de Casos de Uso que se muestra en la imagen siguiente.

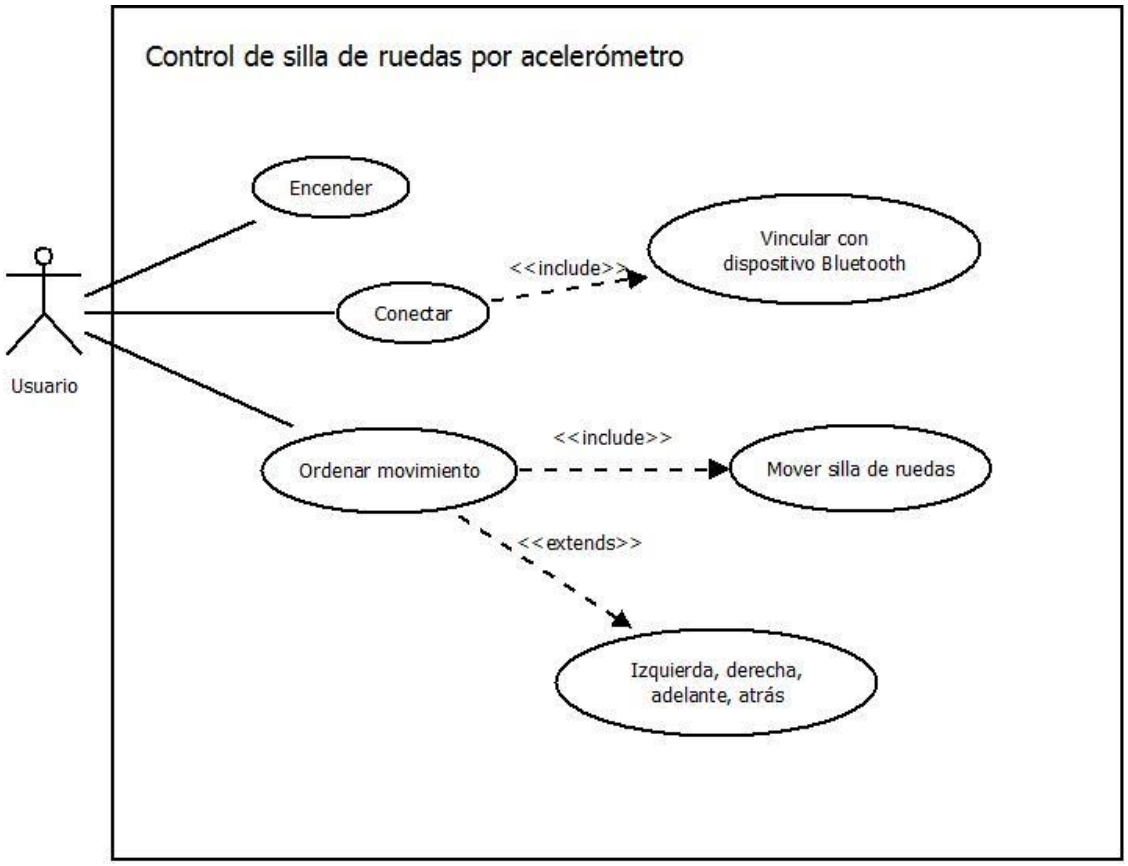

Figura 7.1. Diagrama de casos de uso

El sistema consta de tres órdenes básicas que son Encender, Conectar y Ordenar movimiento; las cuales se describen en la siguiente documentación

DOCUMENTO DE DESCRIPCIÓN DE CASO DE USO

ELABORADO POR: SAMUEL PÉREZ URQUIN FECHA: 12/09/2012

Nombre:	Encender	
Actor:	Usuario	
Descripción:	Proceso cuando el usuario pulsa el botón de encendido	
Precondiciones:	Se haya abierto la aplicación en el teléfono móvil.	
Flujo principal:	**Eventos ACTOR**	**Eventos SISTEMA**

	1. Usuario corre la aplicación	3. Se enciende el Bluetooth del teléfono
Flujo alternativo:		2. Si el Bluetooth del teléfono está encendido envía mensaje de encendido
Post-condiciones:	Usuario enciende el Bluetooth	
Notas:	NO	

Tabla 3. Documentación de caso de uso Encender

DOCUMENTO DE DESCRIPCIÓN DE CASO DE USO

ELABORADO POR: SAMUEL PÉREZ URQUIN FECHA: 12/09/2012

Nombre:	Conectar	
Actor:	Usuario	
Descripción:	Proceso cuando el usuario pulsa el botón de conectar	
Precondiciones:	Se haya encendido el Bluetooth. Se haya encendido el circuito de la silla de ruedas	
Flujo principal:	**Eventos ACTOR**	**Eventos SISTEMA**
	1. Usuario se conecta con el dispositivo de la silla de ruedas	4. El teléfono se conecta con el dispositivo de la silla de ruedas
Flujo alternativo:		2. Si el circuito de la silla de ruedas está apagado no se reconoce dispositivo
	3. Usuario enciende el circuito	
Post-condiciones:	Usuario conectó de manera exitosa	
Notas:	NO	

Tabla 4. Documentación de caso de uso Conectar

DOCUMENTO DE DESCRIPCIÓN DE CASO DE USO

ELABORADO POR: SAMUEL PÉREZ URQUIN FECHA: 12/09/2012

Nombre:	Ordenar movimiento	
Actor:	Usuario	
Descripción:	Proceso cuando el usuario da una orden para movimiento (izquierda, derecha, adelante y atrás)	
Precondiciones:	Se haya conectado con el dispositivo de la silla de ruedas	
Flujo principal:	**Eventos ACTOR**	**Eventos SISTEMA**
	1. Usuario inclina el teléfono móvil enviando la orden de movimiento	4. Se envía la orden de movimiento hacia el dispositivo de la silla de ruedas
Flujo alternativo:		2. Si el dispositivo no está conectado, la orden no se envía
	3. Usuario se conecta con dispositivo de la silla	
Post-condiciones:	Usuario conectó de manera exitosa	
Notas:	NO	

Tabla 5. Documentación de caso de uso Ordenar movimiento

7.2. Diagrama de clases

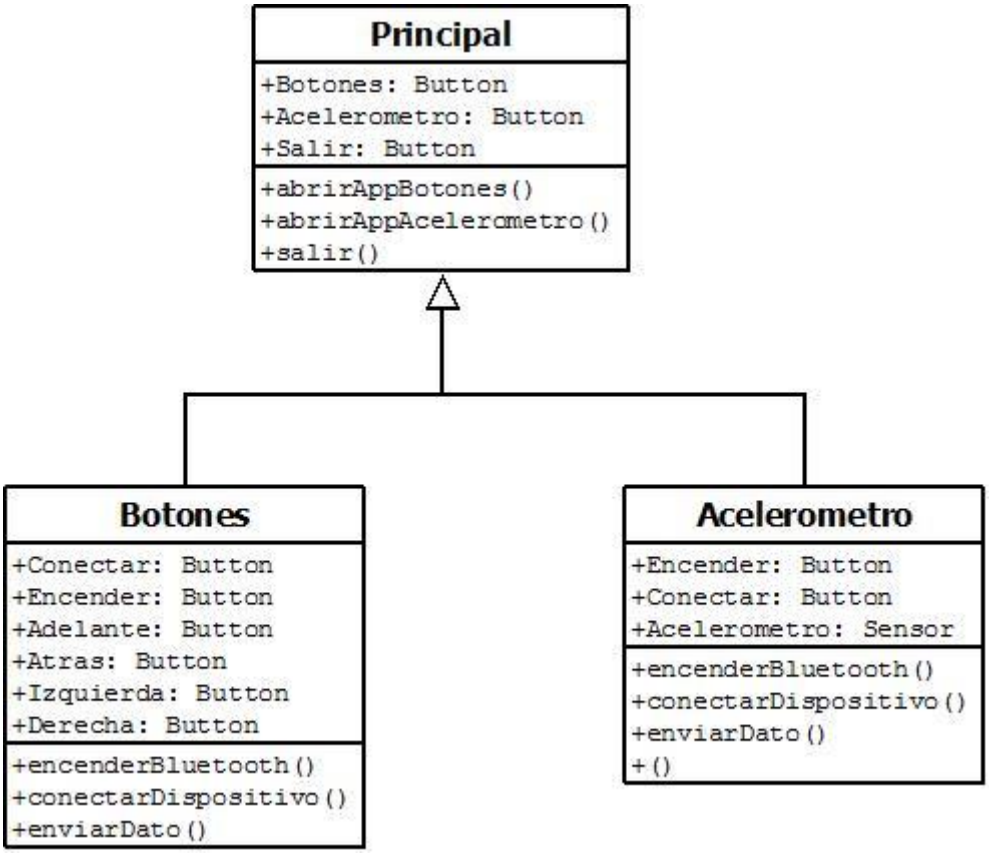

Figura 7.2. Diagrama de clases

7.3 Observaciones

Dentro del proceso de pruebas se observó que el circuito integrado utilizado para la conversión del voltaje de 5V a 12V se calienta con el uso y puede llegar a sobrecalentarse demasiado con el uso prolongado, por lo que se recomienda utilizar un disipador de calor en el conversor o en dado caso, elegir un modelo diferente que no se caliente en demasía.

También se observó que la aplicación es compatible con cualquier versión de Android, el problema es la compatibilidad de los dispositivos móviles con el módulo Bluetooth utilizado para este proyecto por lo que en algunos casos, la aplicación se cerrará de manera repentina o no podrá establecerse la conexión con el dispositivo conectado a la placa Arduino.

8. RESULTADOS

En este apartado presentamos los resultados obtenidos en la realización del proyecto, tanto las pruebas iniciales realizadas sobre una protoboard con leds hasta la prueba final hecha sobre la placa que se instaló en la silla de ruedas para su entrega final.

8.1. Funcionamiento de la silla de ruedas

Se describirá el funcionamiento de la silla de ruedas en una serie de puntos que expondrán paso a paso como trabaja el sistema de movimiento:

1. El usuario ejecuta la aplicación desde su teléfono móvil, el cual presenta una interfaz en la que se elige cómo se operará la silla, ya sea con botones o con acelerómetro que en esta segunda opción fue la que se aplicó para la residencia.
2. Seleccionando acelerómetro, el teléfono se girará hacia adelante, atrás, izquierda o derecha, dependiendo de a dónde quiere ir el usuario para que el teléfono, vía Bluetooth, envíe las órdenes a un módulo que recibirá los datos y como estará conectado a una placa Arduino Uno, se transferirán directamente los datos para que se interpreten.
3. La placa Arduino con el programa que se haya grabado en memoria, se recibirán, leerán e interpretarán los datos enviados desde el módulo Bluetooth para que a través de una comparación de estos, se envíen las órdenes a la parte de potencia.
4. La placa de potencia dónde están conectados los motores, recibirán los pulsos desde Arduino para que los transistores se encarguen de nivelar la corriente y los motores impulsen la silla hacia donde el usuario desee ir.

8.2. Pruebas efectuadas sobre la silla de ruedas

Las pruebas que se efectuaron sobre la silla fueron exitosas y la movilidad es adecuada para una persona que tienen escasa movilidad o no la tienen en la mayoría de sus extremidades. Presentamos una imagen de las pruebas,

igualmente se subió un video que puede ser visto a través de la página de YouTube: [<http://www.youtube.com/watch?v=nGmH1jKwIIQ>]

Figura 8.1. Pruebas de la aplicación sobre la silla de ruedas

8.3. Estabilidad de acelerómetro

En el apartado de Marco Teórico hemos dado detalles sobre el sensor de acelerómetro y su funcionamiento. Gracias a su sensibilidad, es un sensor que puede ser utilizado para la aplicación presentada. En la aplicación de prueba, se envían los datos a tres elementos de texto que nos arrojan los valores cambiantes del acelerómetro y en los cuales podemos ver su sensibilidad al igual su estabilidad cuando el teléfono está fijo o con muy poco movimiento, lo cual es bastante adecuado para lograr una correcta funcionalidad para evitar movimientos no requeridos en caso de que el usuario mueva ligeramente el teléfono.

9. CONCLUSIÓN

Como conclusión, podemos decir que nuestra aplicación cumple con lo establecido en el acuerdo que se presentó al iniciar la residencia, dando como resultado un producto que funciona de manera correcta.

La sensibilidad y estabilidad del sensor acelerómetro en los dispositivos móviles son Sistema Operativo Android, hacen que la aplicación pueda ejecutarse de manera adecuada y cómoda para los usuarios. Los motores están adaptados para que el arranque y detención de la silla no sea brusca para que el usuario pueda ir con comodidad hacia donde quiera llegar.

Al utilizar el compilador Eclipse Índigo que es de distribución libre, hace que los costos de programación se abaraten, lo cual hace la aplicación accesible para todo público. Igualmente en la utilización de Arduino, al ser un hardware que utiliza software libre hace que sea más barato que si se usara otro tipo de tecnología o construir toda la placa de circuitos desde cero.

10. ANEXOS

Anexo 1. Pruebas iniciales en protoboard

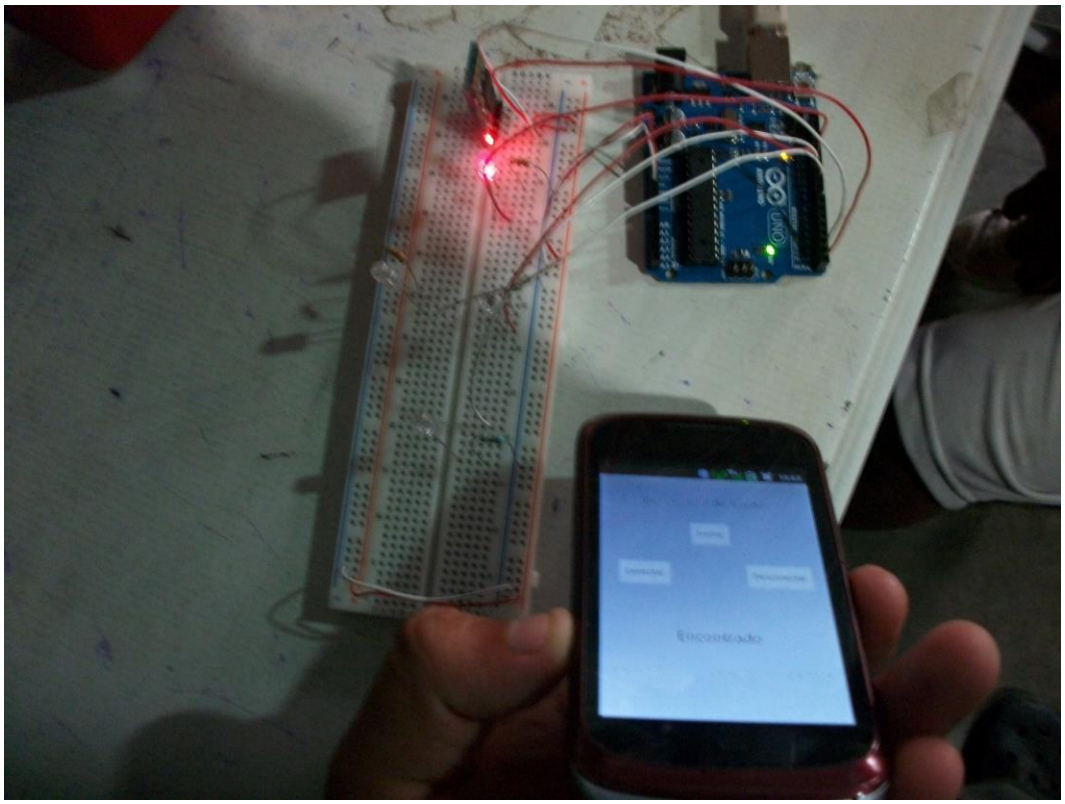

Figura 10.1. Inclinación hacia adelante

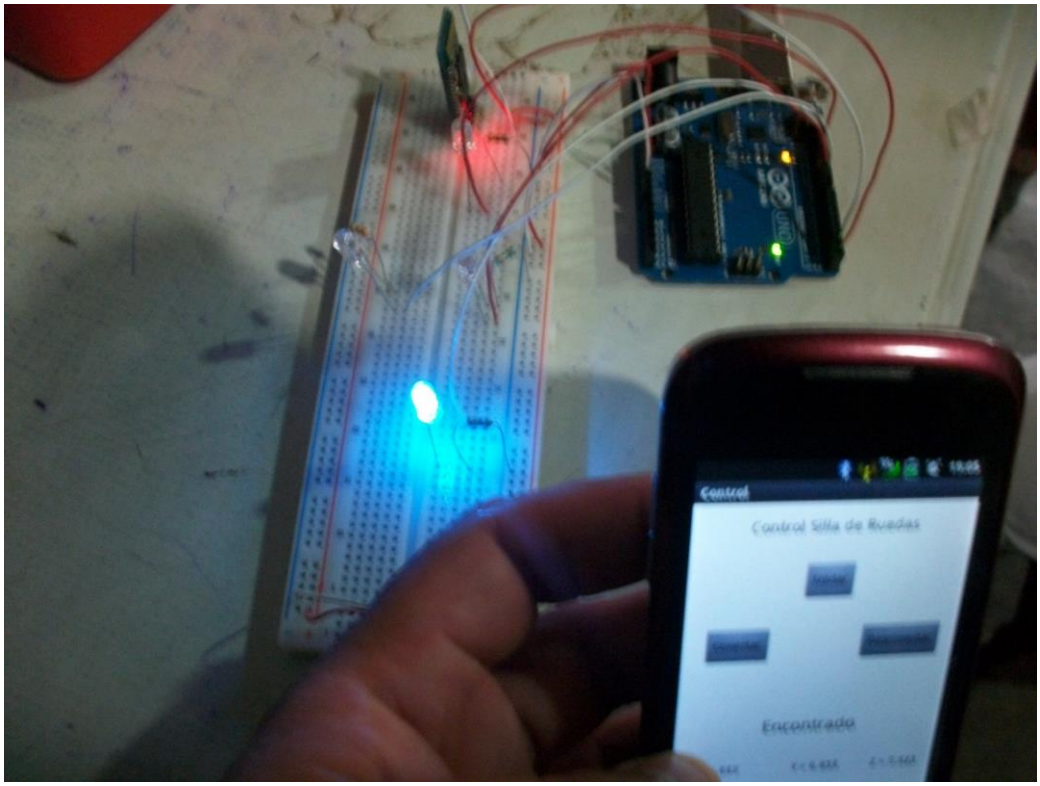

Figura 10.2. Inclinación hacia atrás

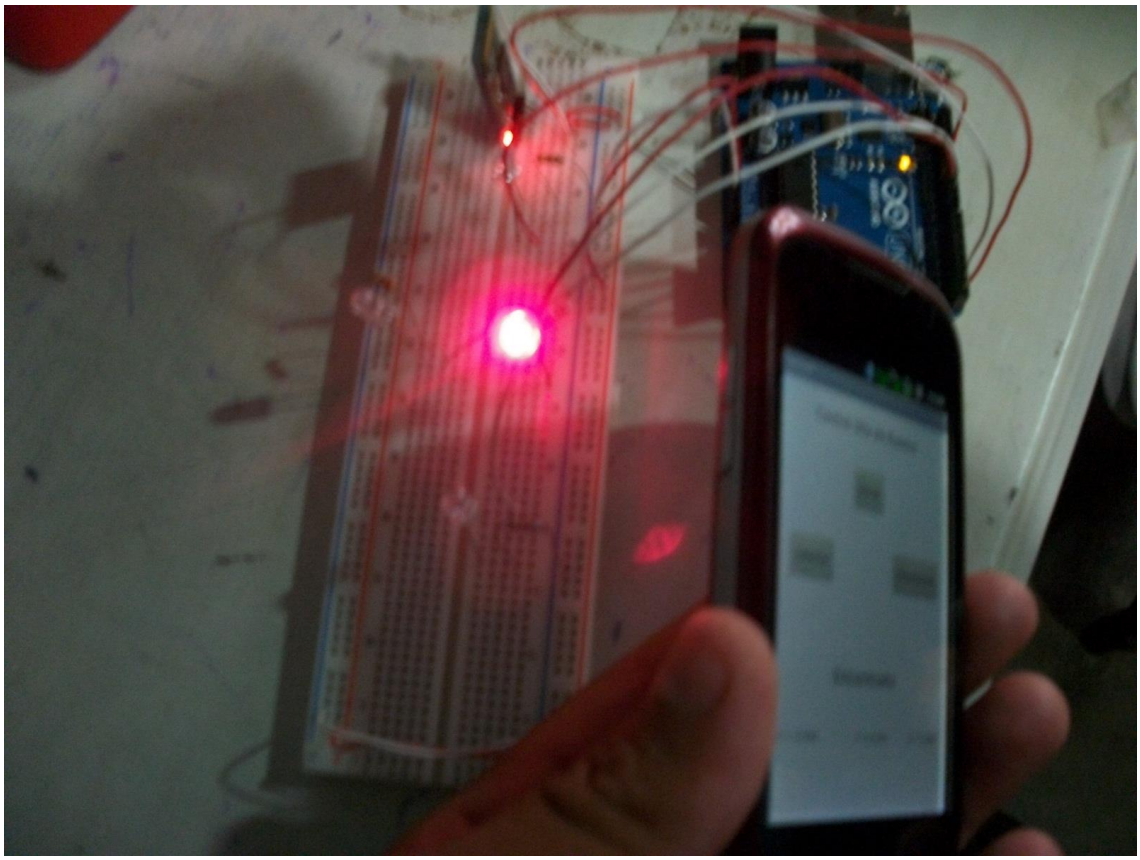

Figura 10.3. Inclinación hacia la derecha

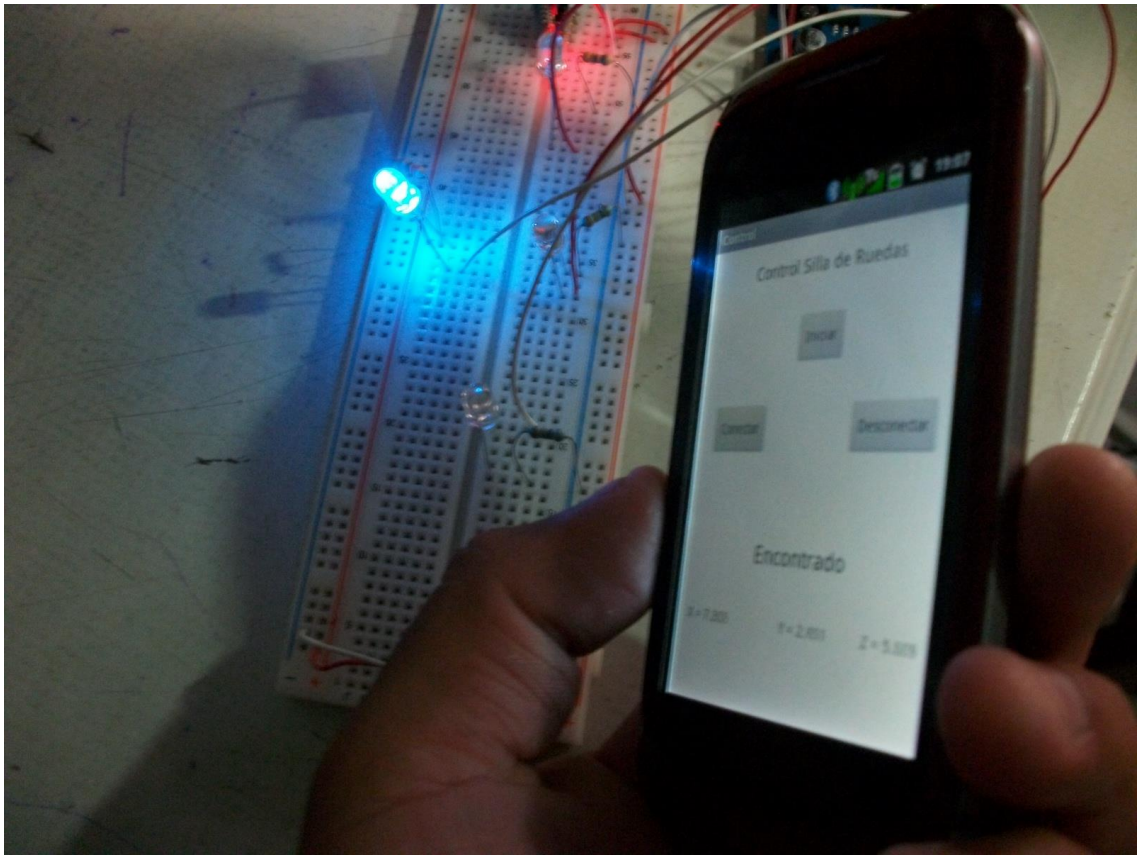

Figura 10.4. Inclinación hacia la izquierda

Anexo 2. Capturas de pantalla de la aplicación

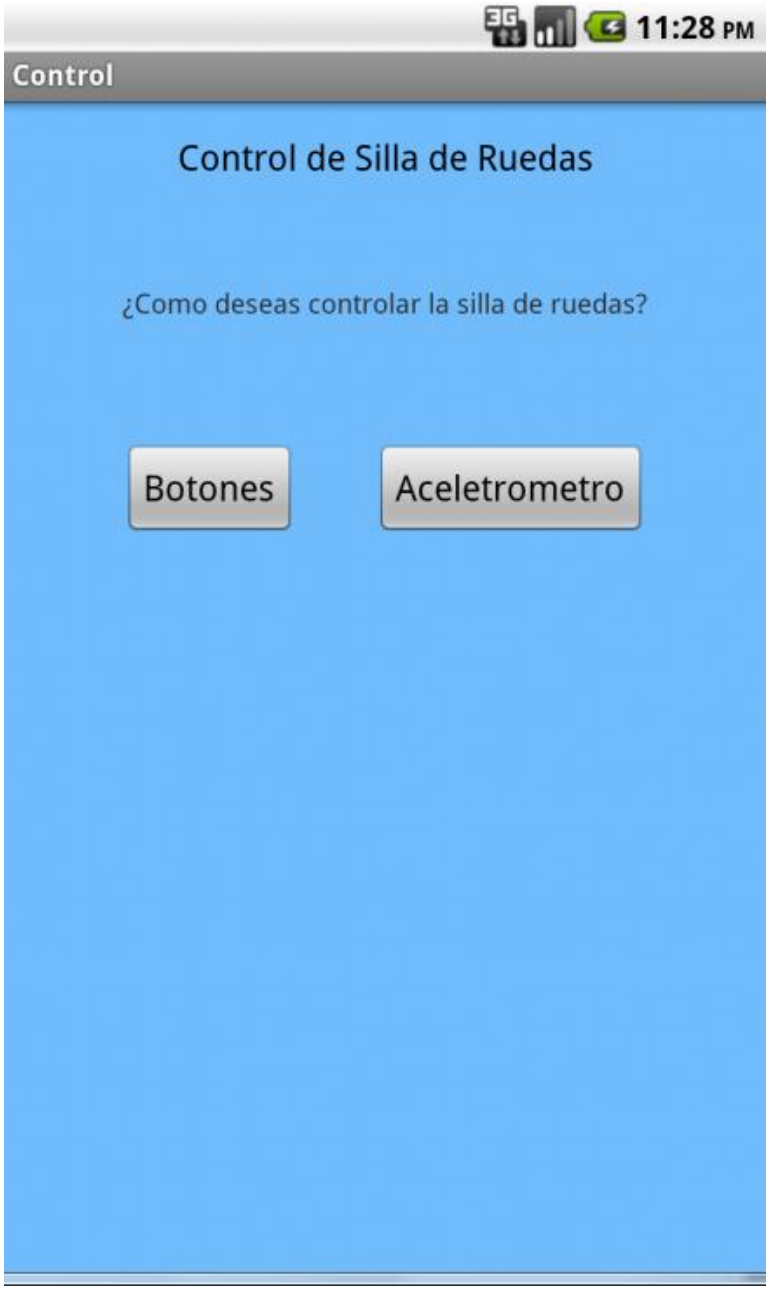

Figura 10.5. Pantalla Principal del Sistema

Figura 10.6. Pantalla de Control por medio de Botones.

Figura 10.7. Pantalla de Control por medio de Acelerometro.

Anexo 3. Código fuente de la aplicación

Código fuente de control por medio de acelerómetro

```
package com.example.control;

import java.io.DataOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.IOException;
import java.util.List;
import java.util.UUID;

import android.os.Bundle;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.view.Menu;
import android.widget.Button;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;
import android.content.Context;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.view.View;
import android.view.View.OnClickListener;
import android.graphics.Color;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.PowerManager;

public class AccelerometerActivity extends Activity implements SensorEventListener {
	//Creacion de variables necesarias para usar BT
		public static final int REQUEST_ENABLE_BT = 0;
		private BluetoothAdapter Adaptador;
		BluetoothSocket mmSocket;
```

```
        BluetoothDevice mmDevice;
    OutputStream mmOutputStream;
    InputStream mmInputStream;
    //Creacion de botones y textos
    Button iniciar,conectar,desconectar;
    TextView estado,encabezado;
    TextView x,y,z;
    private Sensor mAccelerometer;
    float xvalue,yvalue,zvalue;
    int band = 0;
    int band2 = 0;
    int xact, xant, yact, yant;
    int xvalue2, yvalue2, zvalue2;
    protected PowerManager.WakeLock wakelock;


@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_accelerometer);
    RelativeLayout relativelayout01 = (RelativeLayout) findViewById(R.id.layout03);
        relativelayout01.setBackgroundColor(Color.argb(255,112,189,255));
         final PowerManager
pm=(PowerManager)getSystemService(Context.POWER_SERVICE);
      this.wakelock=pm.newWakeLock(PowerManager.SCREEN_DIM_WAKE_LOCK,
"etiqueta");
       wakelock.acquire();
        sensores();

this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
}
void prender(){
//Obtener si el dispositivo usa bluetooth y si esta encendido
Adaptador = BluetoothAdapter.getDefaultAdapter();
if(Adaptador == null){
        estado.setText("No Existe");
}
        if(!Adaptador.isEnabled()){
                Intent encenderBT = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
```

```
                    startActivityForResult(encenderBT,REQUEST_ENABLE_BT);
                }

                String address = "00:11:11:29:01:51";
                mmDevice = Adaptador.getRemoteDevice(address);
    estado.setText("Encontrado");
  }

        void abrirBT() throws IOException{
                //Se crea el socket y se conectan ambos dispositivos
                        UUID uuid = UUID.fromString("00001101-0000-1000-8000-
00805F9B34FB");
                        mmSocket = mmDevice.createRfcommSocketToServiceRecord(uuid);
            mmSocket.connect();
            mmOutputStream = mmSocket.getOutputStream();
            mmInputStream = mmSocket.getInputStream();
            estado.setText("Conectado");
            band = 1;
                }

            void cerrarBT()throws IOException{
                //Se cierran las conexiones
                        mmOutputStream.close();
            mmInputStream.close();
            mmSocket.close();
            estado.setText("Conexion Cerrada");
            band = 0;
                }
            void enviarDatos(int a)throws IOException{
                //Se crea una instancia para crear un socket de salida
                        //byte[] buffer;
                        try {
                                mmOutputStream = new
DataOutputStream(this.mmSocket.getOutputStream());
                        } catch (IOException e) {
                                Toast.makeText(this, "OutputFallido!",
Toast.LENGTH_LONG).show();
                        }
                        //Se fija los valores a enviar y se convierten a bytes y se escriben en el
socket de salida
```

```
                    try {
                            mmOutputStream.write(a);
                            //mmOutputStream.flush();

                    } catch (IOException e) {
                            // TODO Auto-generated catch block
                            Toast.makeText(this, "Envio fallido",
Toast.LENGTH_LONG).show();
                    }
            }

        private void sensores(){
            //Vinculacion de variables con los elementos de pantalla
                    encabezado = (TextView) findViewById(R.id.textView1);
                    estado = (TextView) findViewById(R.id.textView2);
                    iniciar = (Button) findViewById(R.id.button1);
                    conectar = (Button) findViewById(R.id.button2);
                    desconectar = (Button) findViewById(R.id.button3);

                    iniciar.setOnClickListener(new OnClickListener(){
                            @Override
                            public void onClick(View v) {
                                    prender();
                            }
                    });

                    conectar.setOnClickListener(new OnClickListener(){
                            @Override
                            public void onClick(View v) {
                                    try {
                                            abrirBT();
                                    } catch (IOException e) {
                                            // TODO Auto-generated catch block
                                            e.printStackTrace();
                                    }
                            }
                    });

                    desconectar.setOnClickListener(new OnClickListener(){
                            @Override
```

```
                    public void onClick(View v) {
                        try {
                            cerrarBT();
                        } catch (IOException e) {
                            // TODO Auto-generated catch block
                            e.printStackTrace();
                        }
                    }
                });
        }

        protected void onResume(){
            super.onResume();
            wakelock.acquire();
            SensorManager sm = (SensorManager)
getSystemService(SENSOR_SERVICE);
            List<Sensor> sensors = sm.getSensorList(Sensor.TYPE_ACCELEROMETER);
            if(sensors.size() > 0) //Se busca que haya un sensor en el telefono
            {
                //Si encuentra uno se pide que sea acelerometro
                sm.registerListener(this, sensors.get(0),
SensorManager.SENSOR_DELAY_NORMAL);
            }
        }

        @SuppressLint("Wakelock")
            protected void onDestroy(){
           super.onDestroy();
           this.wakelock.release();
        }

        protected void onPause(){
            SensorManager mSensorManager = (SensorManager)
getSystemService(SENSOR_SERVICE);
            mSensorManager.unregisterListener(this, mAccelerometer);
            super.onPause();
        }

        protected void onStop(){
```

```
                    SensorManager mSensorManager = (SensorManager)
getSystemService(SENSOR_SERVICE);
                mSensorManager.unregisterListener(this, mAccelerometer);
                super.onStop();
                }
            public void onSaveInstanceState(Bundle icicle) {
               super.onSaveInstanceState(icicle);
               this.wakelock.release();
            }
            @Override
            public boolean onCreateOptionsMenu(Menu menu) {
                // Inflate the menu; this adds items to the action bar if it is present.
                getMenuInflater().inflate(R.menu.accelerometer, menu);
                        return true;
            }
            @Override
                public void onAccuracyChanged(Sensor arg0, int arg1) {
                // TODO Auto-generated method stub

            }
          @Override
          public void onSensorChanged(SensorEvent event) {
                xvalue = event.values[SensorManager.DATA_X];
                yvalue = event.values[SensorManager.DATA_Y];
                zvalue = event.values[SensorManager.DATA_Z];

                xvalue2 = (int)xvalue;
                yvalue2 = (int)yvalue;
                zvalue2 = (int)zvalue;

                /*this.x.setText("X = " + xvalue +"");
                this.y.setText("Y = " + yvalue +"");
                this.z.setText("Z = " + zvalue +"");*/

                /*if(band2 == 0){
                        xant = (int)xvalue;
                        yant = (int)yvalue;
                        band2 = 1;
                }
```

```
xact = (int)xvalue;
yact = (int)yvalue;
*/

if (band == 1){

	if(yvalue <= -3){
		try {
			enviarDatos(1);
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}
	else if(yvalue >= 3){
		try {
			enviarDatos(2);
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}
	else if(xvalue >= 3){
		try {
			enviarDatos(4);
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}
	else if(xvalue <= -3){
		try {
			enviarDatos(3);
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}
	else{
		try {
```

```
                    enviarDatos(0);
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
            }
        }

}
```

Layout del control por medio de acelerómetro

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/layout03"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".AccelerometerActivity" >

        <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:text="@string/Control"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="32dp"
        android:text="@string/Iniciar"
        android:textSize="18sp" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/button1"
        android:layout_marginTop="32dp"
        android:layout_toLeftOf="@+id/button1"
        android:text="@string/Conectar"
```

```
        android:textSize="18sp" />

    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/button2"
        android:layout_alignBottom="@+id/button2"
        android:layout_alignParentRight="true"
        android:text="@string/Desconectar"
        android:textSize="18sp" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/button2"
        android:layout_marginTop="122dp"
        android:layout_toRightOf="@+id/button2"
        android:text="@string/Estado"
        android:textAppearance="?android:attr/textAppearanceLarge" />

</RelativeLayout>
```

Código fuente del control por medio de botones.

```
package com.example.control;

import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.UUID;

import android.os.Bundle;
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.graphics.Color;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;

public class ButtonActivity extends Activity {
        public static final int REQUEST_ENABLE_BT = 0;
        private BluetoothAdapter Adaptador;
        BluetoothSocket mmSocket;
        BluetoothDevice mmDevice;
    OutputStream mmOutputStream;
    InputStream mmInputStream;
    Button iniciar,conectar,desconectar,izquierda,adelante,derecha,atras,stop;
    TextView estado;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_button);
                RelativeLayout relativelayout01 = (RelativeLayout) findViewById(R.id.layout02);
```

```
        relativelayout01.setBackgroundColor(Color.argb(255,112,189,255));

        estado = (TextView) findViewById(R.id.textoView1);
        iniciar = (Button) findViewById(R.id.boton1);
        conectar = (Button) findViewById(R.id.boton2);
        desconectar = (Button) findViewById(R.id.boton3);
        izquierda = (Button) findViewById(R.id.boton4);
        adelante = (Button) findViewById(R.id.boton5);
        derecha = (Button) findViewById(R.id.boton6);
        atras = (Button) findViewById(R.id.boton7);
        stop = (Button) findViewById(R.id.boton8);

        conectar.setOnClickListener(new OnClickListener() {

                @Override
                public void onClick(View arg0) {
                        try {
                                abrirBT();
                        } catch (IOException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
                }
        });
iniciar.setOnClickListener(new OnClickListener() {

                @Override
                public void onClick(View v) {
                                prender();
                }
        });
desconectar.setOnClickListener(new OnClickListener() {

                @Override
                public void onClick(View v) {
                                try {
                                        cerrarBT();
                                } catch (IOException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
```

```
                }
            }
        });

izquierda.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                try {
                    enviarDatos(4);
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        });
adelante.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                try {
                    enviarDatos(1);
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        });
derecha.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                try {
                    enviarDatos(3);
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        });
```

```
atras.setOnClickListener(new OnClickListener(){
	@Override
	public void onClick(View v) {
		try {
			enviarDatos(2);
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}

});
stop.setOnClickListener(new OnClickListener(){
	@Override
	public void onClick(View v) {
		try {
			enviarDatos(0);
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}
});
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
	// Inflate the menu; this adds items to the action bar if it is present.
	getMenuInflater().inflate(R.menu.button, menu);
	return true;
}

void prender(){
//Obtener si el dispositivo usa bluetooth y si esta encendido
Adaptador = BluetoothAdapter.getDefaultAdapter();
if(Adaptador == null){
	estado.setText("No Existe");
}
	if(!Adaptador.isEnabled()){
```

```
                    Intent encenderBT = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                    startActivityForResult(encenderBT,REQUEST_ENABLE_BT);
                }

                String address = "00:11:11:29:01:51";
                mmDevice = Adaptador.getRemoteDevice(address);
    estado.setText("Encontrado");
  }

        void enviarDatos(int a)throws IOException{
        //Se crea una instancia para crear un socket de salida
                //byte[] buffer;
                try {
                        mmOutputStream = new
DataOutputStream(this.mmSocket.getOutputStream());
                } catch (IOException e) {
                        Toast.makeText(this, "OutputFallido!", Toast.LENGTH_LONG).show();
                }
                //Se fija los valores a enviar y se convierten a bytes y se escriben en el socket
de salida
                try {
                        mmOutputStream.write(a);
                        //mmOutputStream.flush();

                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        Toast.makeText(this, "Envio fallido", Toast.LENGTH_LONG).show();
                }
        }

  void abrirBT() throws IOException{
        //Se crea el socket y se conectan ambos dispositivos
                UUID uuid = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
                mmSocket = mmDevice.createRfcommSocketToServiceRecord(uuid);
    mmSocket.connect();
    mmOutputStream = mmSocket.getOutputStream();
    mmInputStream = mmSocket.getInputStream();
    estado.setText("Conectado");
        }
```

```
    void cerrarBT()throws IOException{
        //Se cierran las conexiones
                mmOutputStream.close();
        mmInputStream.close();
        mmSocket.close();
        estado.setText("Conexion Cerrada");
            }

}
```

Layout del control por medio de botones

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/layout02"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".ButtonActivity" >

     <Button
        android:id="@+id/boton1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="30dp"
        android:layout_marginTop="34dp"
        android:text="@string/Iniciar" />

    <Button
        android:id="@+id/boton2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/boton1"
        android:layout_alignBottom="@+id/boton1"
        android:layout_alignParentRight="true"
        android:layout_marginRight="47dp"
        android:text="@string/Conectar" />

    <Button
        android:id="@+id/boton3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/boton1"
        android:layout_marginTop="22dp"
        android:layout_toRightOf="@+id/boton1"
```

```
    android:text="@string/Desconectar" />

<Button
    android:id="@+id/boton5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/boton3"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="19dp"
    android:text="Adelante"
    android:textSize="16sp" />

<TextView
    android:id="@+id/textoView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignRight="@+id/boton8"
    android:layout_marginBottom="14dp"
    android:text="@string/Estado"
    android:textAppearance="?android:attr/textAppearanceMedium" />

<Button
    android:id="@+id/boton4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/boton5"
    android:layout_marginTop="23dp"
    android:text="Izquierda"
    android:textSize="16sp" />

<Button
    android:id="@+id/boton8"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/boton4"
    android:layout_alignBottom="@+id/boton4"
    android:layout_centerHorizontal="true"
    android:text="Parar"
```

```
        android:textSize="16sp" />

    <Button
        android:id="@+id/boton6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/boton8"
        android:layout_alignBottom="@+id/boton8"
        android:layout_alignParentRight="true"
        android:text="Derecha"
        android:textSize="16sp" />

    <Button
        android:id="@+id/boton7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/textoView1"
        android:layout_below="@+id/boton8"
        android:layout_marginTop="30dp"
        android:text="Atras"
        android:textSize="16sp" />

</RelativeLayout>
```

Anexo 4. Código fuente de módulo arduino

```
//Distribucion de pines en la placa
int llanta1     = 6;
int llanta11    = 9;
int llanta2     = 10;
int llanta22    = 11;


char val;  // variable que recibe datos desde el puerto serial

void setup() {

  // inicializar los pines de salida
  pinMode(llanta1, OUTPUT);
  pinMode(llanta11, OUTPUT);
  pinMode(llanta2, OUTPUT);
  pinMode(llanta22, OUTPUT);

  Serial.begin(9600);      // pa la comunicacion de inicio del bluetooth
}
void w() {
  digitalWrite(llanta1,HIGH);
  digitalWrite(llanta11,LOW);
  digitalWrite(llanta2,HIGH);
  digitalWrite(llanta22,LOW);
  Serial.println("Sasha Grey");
}

void s(){
  digitalWrite(llanta1,LOW);
  digitalWrite(llanta11,HIGH);
  digitalWrite(llanta2,LOW);
  digitalWrite(llanta22,HIGH);
  Serial.print("Foxy_Love");
}

void a() {
  digitalWrite(llanta1,HIGH);
  digitalWrite(llanta11,LOW);
```

```
  digitalWrite(llanta2,LOW);
  digitalWrite(llanta22,HIGH);
  Serial.println("Dayyanna");
}

void d(){
  digitalWrite(llanta1,LOW);
  digitalWrite(llanta11,HIGH);
  digitalWrite(llanta2,HIGH);
  digitalWrite(llanta22,LOW);
  Serial.print("Pamela Anderson");
}

void pff() {
  digitalWrite(llanta1,LOW);
  digitalWrite(llanta11,LOW);
  digitalWrite(llanta2,LOW);
  digitalWrite(llanta22,LOW);
  Serial.println("Stop");
}

// Leer el puerto serial para realizar comando
void performCommand() {
  if (Serial.available()) {
    val = Serial.read();
  }
    if (val == 1) { // adelante
      w();
    }
    else if (val == 2) { // atras
      s();
    }
    else if (val == 4) { // girar lado izquierda
      a();
    }
    else if (val == 3) { // girar lado derecha
      d();
    }
    else if (val == 0) { // parar
      pff();
```

```
    }
}
void loop() {
  performCommand();
}
```