# PRÁCTICAS DE GESTIÓN DE ENERGÍA EN APLICACIONES PARA DISPOSITIVOS ANDROID, SOPORTADAS EN PERIFÉRICOS ESPECÍFICOS


Universidad
del Cauca

Trabajo de Grado

**Manuel Fernando Fuentes Amaya**
**Johan Alberto Gómez Girón**

Director: PhD. Gustavo Adolfo Ramírez González

**Universidad del Cauca**
**Facultad de Ingeniería Electrónica y Telecomunicaciones**
**Departamento de Telemática**
**Línea de Investigación en Servicios avanzados en Telecomunicaciones**
**Popayán, Abril de 2013**

# Tabla de Contenido

# Listado de Figuras

# Listado de Tablas

# Capítulo 1

# Introducción

Durante los últimos años ha aumentado la necesidad y la utilización de los teléfonos móviles, los cuales se han convertido en dispositivos indispensables, integrándose a las actividades diarias (trabajo y entretenimiento especialmente) y facilitando la forma de comunicarse con el mundo.

Siguiendo las dinámicas del mercado y las necesidades actuales de un mundo globalizado y constantemente cambiante, los terminales de telefonía móvil han evolucionado integrando en ellos una serie de características que les dan utilidades y funcionalidades para ser usados en diferentes ambientes.

A partir de esta evolución surgen los móviles inteligentes, llamados "smartphones", los cuales contienen interfaces múltiples de comunicación, como interfaces WiFi y bluetooth; procesadores con altas frecuencias de funcionamiento, memorias de altas capacidades, y numerosos periféricos, incluyendo GPS, sensores de aceleración, brújula, cámaras, sistemas de altavoces, pantallas táctiles y/o teclados lo que ha aumentado las necesidades de energía [1, 2].

Este trabajo de grado se exploran las diferentes formas de programación que permitan generar un conjunto de prácticas de gestión de energía en los dispositivos Android, sobre unos periféricos específicos.

## 1.1. Motivación

La utilización de dispositivos móviles en la vida cotidiana de las personas ha tenido un gran crecimiento en los últimos años, debido a la integración de múltiples funcionalidades que han aumentado su tiempo de uso, haciendo necesaria una disponibilidad cada vez mayor, y por tanto una mayor cantidad de energía o un mejor uso de esta. Es por esto que en este trabajo se buscan prácticas en la forma de programación y/o uso de aplicaciones para mejorar la gestión y consumo de la energía en estos dispositivos.

## 1.2. Problema

La evolución de los teléfonos inteligentes, ha venido de la mano del desarrollo de un gran número de aplicaciones orientadas a diferentes segmentos del mercado, como la educación, el hogar, la banca, el entretenimiento, salud, el sector empresarial, entre otras. Esto ha permitido que los teléfonos inteligentes brinden facilidades a sus usuarios, integrando características de diferentes dispositivos (como computadores, televisores, radios, módems, entre otros) y servicios (como la banca, el correo, compras electrónicas, entre otros). Sin embargo algunas de estas facilidades requieren alto procesamiento, lo cual se ve reflejado en la duración de la batería, ya que hay un considerable aumento en el consumo de energía originado por la

mayor utilización del procesador y diversos periféricos que se han integrado en el dispositivo móvil [3][4].

Paralelamente, la capacidad de la batería crece a un ritmo mucho más lento que las necesidades de energía de los móviles inteligentes, haciendo necesario definir y utilizar alternativas para la gestión de energía, optimizando así, la duración de la misma [1, 5].

Por esta razón, se pretenden realizar procesos experimentales para encontrar prácticas adecuadas de gestión de energía para periféricos específicos. El estudio se realizara sobre el sistema operativo Android, el cual actualmente lidera el mercado de los Smartphone [6], con una tienda de aplicaciones que posee un poco más de 500.000, de las cuales 320.000 están activas para ser descargadas [7].

Aunque evidentemente uno de los periféricos que mas consume energía actualmente es el modem de datos usado en redes Wi-Fi, 3G o 4G es importante considerar otro tipo de periféricos que son de uso común entre los usuarios y que continuamente están involucrados en el uso de la mayoría de las aplicaciones que pueden encontrarse en la tienda de aplicaciones, o ser de interés para desarrollos específicos. Por esta razón el presente trabajo se enfoca en la investigación del consumo de energía sobre los periféricos: acelerómetro, brújula, sistema de audio (auricular y altavoz) y el teclado, los cuales no se han estudiado en profundidad en cuanto a su impacto en el consumo de energía; adicionalmente, se estudiara el periférico GPS que tiene un gran impacto sobre la duración de la batería.

De acuerdo a este contexto, la pregunta de investigación central de este trabajo de grado es: ¿Cuáles son las prácticas más adecuadas para la gestión de energía, reflejadas en diferentes formas de programación y/o uso de aplicaciones disponibles para dispositivos Android, soportadas en periféricos específicos[1]?

## 1.3. Objetivos

El objetivo principal de este trabajo es definir un conjunto de prácticas de gestión de energía basadas en formas de programación y/o uso de aplicaciones disponibles para dispositivos Android, soportadas en periféricos específicos. Como objetivos específicos se propone:

1. Proponer un modelo de evaluación de consumo de energía soportado en periféricos específicos sobre dispositivos móviles android.

2. Diseñar y ejecutar pruebas para medir los niveles de consumo de energía de cada periférico en diferentes escenarios, mediante programación específica y/o uso de aplicaciones disponibles en el Google Play.

3. Caracterizar el consumo de energía de cada periférico, en diferentes escenarios bajo el modelo de evaluación de consumo propuesto.

## 1.4. Hipótesis

Como hipótesis inicial para el desarrollo de este trabajo, se plantea: Es posible definir un conjunto de prácticas de gestión de energía basadas en formas de programación y/o uso de aplicaciones disponibles para dispositivos Android, soportadas en periféricos específicos.
Bajo esta suposición, el desarrollo de un conjunto de prácticas de gestión de energía pueden disminuir el consumo de energía en la utilización de los periféricos.

[1] En este caso: GPS, acelerómetro, brújula, sistema de audio (auricular y altavoz) y el teclado.

## 1.5. Experimentación

Como parte de la experimentación se diseñaron y ejecutaron pruebas en escenarios reales de uso de los dispositivos con el fin de encontrar prácticas de gestión de energía basadas en formas de programación y/o uso de aplicaciones. Para ello se creó un modelo de evaluación del consumo de energía el cual definirá los lineamientos para el diseño de pruebas procesamiento y análisis de resultados.

## 1.6. Conclusiones y divulgación

Se realiza la síntesis de los resultados más relevantes, la recolección de experiencias, lecciones aprendidas y elementos para ser tenidos en cuenta a futuro. se llevara a cabo la publicación de un artículo de divulgación que describe de manera precisa los logros alcanzados en el proyecto y las conclusiones del trabajo.

## 1.7. Metodología

El modelo para la investigación científica se tomará como referencia metodológica para el desarrollo general del proyecto. La figura 1 muestra las diferentes etapas del proceso. En la fase de experimentación, en caso


Figura 1.1: Etapas de desarrollo del proyecto.

de ser necesario el uso desarrollos puntuales, se utilizará el "Modelo para la Construcción de Soluciones" (MCS) componentes del Modelo Integral para el Profesional en Ingeniería [8].

## 1.8. Partes de la monografía

Este documento ha sido divido de la siguiente forma:

- Capitulo 1: Presenta la Introducción , el planteamiento del problema y la estructura del presente trabajo de grado.
- Capitulo 2: Denominado "Marco conceptual", hace referencia a los conceptos sobre las baterías sus parámetros de medición, las tecnologías y otras investigaciones que tratan la gestión de energía en dispositivos móviles.
- Capitulo 3: Denominado "Modelo para la evaluación de consumo de energía sobre dispositivos móviles", presenta un modelo para la evaluación del consumo de energía que es desarrollado.
- Capitulo 4: Denominado "Herramienta para la medición de consumo de energía en dispositivos móviles", presenta el análisis de las herramientas y métodos para la medición del consumo de energía utilizados en experiencias previas de otros investigadores para la implementan de una herramienta que se adapte al modelo de evaluación presentado en el capitulo 3.
- Capitulo 5: Denominado "Planificación de pruebas", se desarrolla la caracterización del consumo inicial y diseño de pruebas en base al "Modelo para la evaluación del consumo de energía".

- Capitulo 6: Denominado “Ejecución y análisis de pruebas ”, se aplican los procesos para la ejecución de pruebas y para el análisis de mediciones descritos en el “Modelo para la evaluación del consumo de energía”.
- Capitulo 7: presenta las conclusiones, lecciones aprendidas y trabajos futuros, derivados de este trabajo de grado
- Finalmente, los “Anexos” donde se encuentra material complementario como manuales de usuario, documentación de modelado e información que sera referenciada a lo largo del documento.

# Capítulo 2

# Marco conceptual

Este capítulo se recopilan los conceptos y tecnologías en los que se fundamenta este trabajo y presenta las experiencias de otros investigadores que constituyen el conocimiento base de la temática de investigación y un marco para identificar los posibles enfoque que pueden ser utilizados, las limitaciones que existen y los campos por explorar identificando los posibles aportes .

## 2.1. Conceptos fundamentales

### 2.1.1. Parámetros de las baterías en los dispositivos móviles

Desde que aparecieron los dispositivos móviles [9], se ha tenido la necesidad de utilizar baterías que suplan las necesidades energéticas de estos, teniendo que aumentar sus capacidades de carga, incrementar su duración, disminuir su tamaño y mitigar su impacto de contaminación al planeta. Por eso se han venido desarrollando diversos tipos de baterías con el fin de acercarse a estos requerimientos.

**Densidad de energía:** Es la propiedad que permite conocer cuánta energía es posible almacenar por Kg de batería, Se mide en Wh/kg [10].

**Capacidad (Carga eléctrica almacenada):** Es la cantidad de carga eléctrica que es posible almacenar en la batería, es el número aproximado de electrones que quedan atrapados en la batería al someterse a carga. Se mide en Amperios*hora (Ah) [10] [11].

**Ciclo de vida:** Es la estimación del número de ciclos carga-descarga que puede soportar una batería antes de perder el 100 % de su capacidad de carga [10].

**Efecto de memoria:** El efecto memoria hace alusión a la reducción de la capacidad de carga de una batería al someterla a una carga incompleta, es decir, que la batería debe descargarse por completo antes de cargarse de nuevo, si esto no se hace, la duración de la carga se verá disminuida considerablemente. Por ejemplo, si la batería es cargada aun teniendo un 25 % de la carga, luego de ser cargada usará solo el 75 % de capacidad total [10].

### 2.1.2. Tipos de baterías

Desde la aparición de los dispositivos móviles se han utilizado diversos tipos de baterías recargables, las más usadas han sido: Níquel - Cadmio (Ni-Cd), Níquel e Hidruro metálico (Ni-MH), Ion de Litio (Li-Ion) y Polímero de Litio (Li-Po). A continuación se describen:

**Batería de Níquel-Cadmio (Ni-Cd):** Fueron las primeras baterías comerciales para dispositivos móviles, pero su utilización ha disminuido considerablemente en los últimos años. Cuando esta batería es recargada sin estar totalmente descargada ocurre un efecto llamado memoria, en el cual se crean cristales dentro de la batería en consecuencia a una reacción química cuando la batería se calienta, lo que produce auto-descarga o corto circuito. Esta puede perder alrededor del 10 % de su energía en las posteriores 24 horas de su no utilización. Además son perjudiciales para el medioambiente por contener metales tóxicos [12].

**Batería de Níquel e Hidruro metálico (Ni-MH):** Este tipo utiliza el hidrogeno para el proceso de producción de energía, permitiendo un mayor almacenamiento de energía, aproximadamente entre un 40 % más que en una Ni-Cd del mismo tamaño [12]. Este tipo de batería contiene materiales no tóxicos, su tiempo de carga es mayor que en las Ni-Cd, no sufren del efecto de memoria, pero si presentan el efecto de auto-descarga siendo la tasa mayor que en las Ni-Cd [12].

**Baterías de ion de litio (Li-Ion):** Basada en iones de litio, estas baterías no sufren del efecto de memoria y cuentan con una gran capacidad específica de carga. Alcanza hasta tres veces la capacidad de almacenamiento de las Ni-Cd. Presenta una tasa de auto-descarga baja y un voltaje por célula alto, cada batería proporciona 3,7 voltios, que equivale a 3 baterías de Ni-Cd (cada batería de Ni-Cd maneja 1,2 voltios aproximadamente) [12][13]. El tiempo de carga es más lento y a temperaturas muy bajas éstas ofrecen un rendimiento menor que las de Níquel [13].

**Batería de polímero de litio (Li-Po):** Estas baterías no sufren el efecto de memoria, son más estables que las de Li-Ion y su densidad de energía es entre 5 y 12 veces mayor que las de níquel (Ni-Cd y Ni-MH) si posee el mismo peso [13]. El tiempo de carga de las baterías de Li-Po, con respecto a las de Ni-Cd, es más lento, además éstas baterías no producen el pico típico que producen las de Níquel al alcanzar la máxima carga, por lo que los cargadores para las baterías de Li-Po deben ser especiales y no ser cargadas con cargadores diseñados para las baterías de Ni-Cd o Ni-MH porque se corre el riesgo de deteriorar su capacidad. Cada batería de Li-Po posee un voltaje de 3.7V, y no deben dejarse descargar hasta niveles muy bajos, ya que esto puede causar una disminución y deterioro en su capacidad de carga que puede llegar a ser irreversible [13]. La tabla 2.1 muestra las características de los tipos de baterías mencionados

| Características / Tipos | Níquel - Cadmio (Ni-Cd) | Níquel e Hidruro Metálico (Ni-MH) | Ion de Litio (Li-Ion) | Polímero de litio (Li-Po) |
|---|---|---|---|---|
| Ciclos de carga -descarga | 1500 | 300-500 | 500-1000 | 300-500 |
| Capacidad por celda | 1.2v | 1.2v | 3.7v | 3.6 |
| Efecto de memoria | Si | No | No | No |
| Densidad (Wh/kg) | 45-80 | 60-120 | 110-160 | 100-130 |
| Temperatura de funcionamiento (solo descarga) | -40 a 60 °C | -20 a 60°C | -20 a 60°C | 0 a 60°C |

Tabla 2.1: Características baterías para móviles.Adaptado de University Battery [12].

## 2.2. Tecnologías relacionadas

### 2.2.1. Periféricos específicos

El desarrollo de este trabajo se enfocará sobre el consumo de energía asociado a los periféricos GPS, acelerómetro, brújula, sistema de audio (auricular y altavoz) y el teclado. Teniendo en cuenta las limitaciones. A continuación se definen algunos conceptos pertinentes al funcionamiento de los periféricos.

#### GPS

El GPS es uno de los periféricos que mas usa con frecuencia en los dispositivos móviles, en especial en smartphones. Su principal función es proporcionar la ubicación del dispositivo utilizando datos de una serie de satélites para realizar los cálculos de la posición terrestre. La primera inicialización es costosa en cuanto al tiempo, ya que se debe encontrar los satélites más cercanos para tener mayor precisión y así utilizar los datos de los satélites para triangular la posición [14]. Además el GPS puede funcionar de forma asistida la cual se denomina A-GPS.

El A-GPS mejora la ubicación en lugares cerrados, acelerando la velocidad de inicio y garantizando una respuesta más rápida, con un menor consumo de energía. El A-GPS agrega nuevas características software al GPS, por tanto no es una tecnología distinta y puede funcionar como un GPS convencional [15].

Cuando se enciende el A-GPS el teléfono envía la información de la identificación de la antena más cercana a un servidor por medio de la red (3G, 4G, Wi-Fi), el servidor posee una base de datos con la información de las posiciones de los satélites, lo cual le permite retornar al móvil la información de los satélites más cercanos a su ubicación por medio de http [15]. De ahí en adelante el funcionamiento es similar al de GPS.

#### Acelerómetro

Un acelerómetro es un periférico que mide las dos fuerzas de aceleración, la estática y la dinámica. La estática es la fuerza constante de la gravedad con dirección a la tierra, mientras que la dinámica es la fuerza causada por el movimiento o vibración del dispositivo [16].

Al medir la cantidad de aceleración estática, se puede averiguar el ángulo de inclinación del dispositivo en relación con la tierra. Al detectar la cantidad de aceleración dinámica, se puede analizar la forma en que el dispositivo está en movimiento [17]. Este sensor nos proporciona tres valores que corresponden, a los tres ejes cartesianos (x, y, z), cada uno tiene su situación fija con respecto al dispositivo móvil [18].

#### Sistema de Audio

Los dispositivos móviles ofrecen la posibilidad de reproducir archivos de audio, por lo general el sistema de audio cuenta con dos características hardware para la reproducción, la primera es la salida de altavoz y la segunda la salida para auriculares (audífonos).

#### Teclado

Los teclados son periféricos comunes en los dispositivos móviles, aunque no todos los poseen de la misma forma. Existen los teclados físicos tradicionales, en el caso de los smartphones se encuentran el tipo

QWERTY, además de estos teclados, con la aparición de las pantallas táctiles (de buen tamaño) surgió la posibilidad de incorporar el teclado en la misma pantalla. Este trabajo de grado se centrara en teclado hardware (QWERTY) y teclados táctiles (en pantalla).

### 2.2.2. Android

Android es un sistema operativo para dispositivos móviles, inicialmente fue desarrollado por Android Inc., una firma adquirida en 2005 por Google. Google siguió con el desarrollo de la plataforma android y realizó su presentación el 5 de noviembre de 2007. Este sistema operativo está basado en Linux y es una plataforma completa y totalmente abierta [19].
Al ser una plataforma abierta, Android ofrece a los desarrolladores infinitas posibilidades de desarrollo en los dispositivos, ya que a estos se les ofrece de forma gratuita un SDK y las opciones de un plug-in para el entorno de desarrollo de Eclipse, con lo cual cuentan con todas las APIS necesarias para la creación de aplicaciones.
Las aplicaciones para Android se programan en lenguaje Java o en C++ (en el caso del Kit de Desarrollo Nativo - NDK) son ejecutadas en una máquina virtual llamada Dalvik, la cual fue especialmente diseñada para esta plataforma. El núcleo de Android está basado en Linux 2.6 [19].

#### Arquitectura de Android

La arquitectura de Android contiene una pila de software donde se incluye un sistema operativo, middleware y aplicaciones básicas para el usuario [19] . Su diseño cuenta con las siguientes características:

- Busca el desarrollo rápido de aplicaciones, que sean reutilizables y verdaderamente portables entre diferentes dispositivos.
- Los componentes básicos de las aplicaciones se pueden sustituir fácilmente por otros.
- Cuenta con su propia máquina virtual, Dalvik, que interpreta y ejecuta código escrito en Java.
- Permite la representación de gráficos 2D y 3D.
- Posibilita el uso de bases de datos.
- Soporta un elevado número de formatos multimedia.
- Servicio de localización GSM.
- Controla los diferentes elementos hardware: Bluetooth, Wi-Fi, cámara fotográfica o de vídeo, GPS, acelerómetro, infrarrojos, en fin, siempre y cuando el dispositivo móvil lo contemple.
- Cuenta con un entorno de desarrollo muy cuidado mediante un SDK disponible de forma gratuita.
- Ofrece un plug-in para uno de los entornos de desarrollo más populares, Eclipse, y un emulador integrado para ejecutar las aplicaciones.

En las siguientes líneas se dará una visión global por capas de la arquitectura empleada en Android [19][20]. Cada una de estas capas utiliza servicios ofrecidos por las anteriores, y permite a su vez interactuar con las capas de niveles superiores.
A continuación la figura 2.1 muestra la arquitectura de Android.

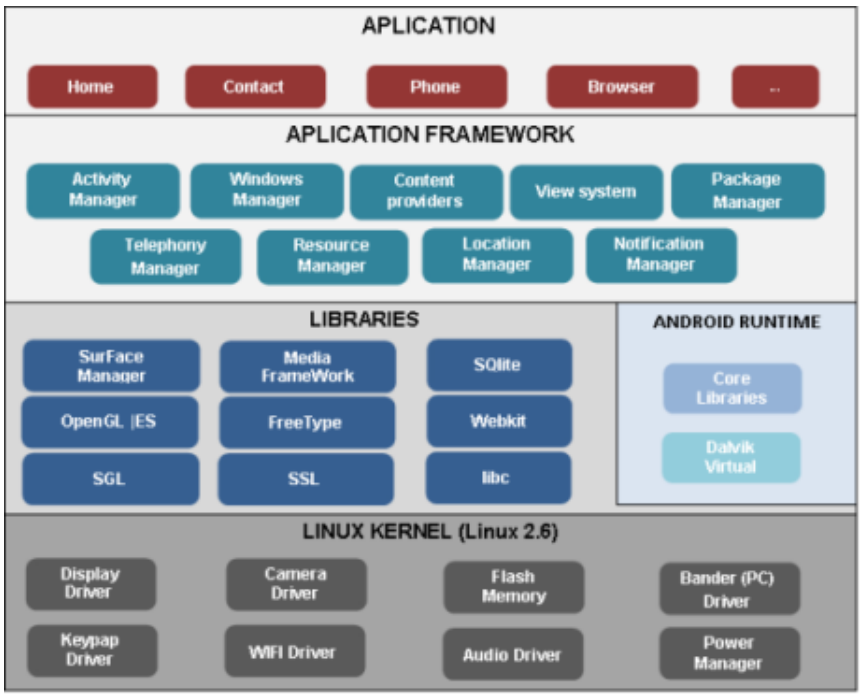


Figura 2.1: Arquitectura de Android. Adaptado de [20].

- La capa más baja corresponde al núcleo de Android, utiliza el núcleo de Linux 2.6 como una capa de abstracción para el hardware disponible en los dispositivos móviles. Esta capa contiene los drivers necesarios para que cualquier componente hardware pueda ser utilizado mediante las llamadas correspondientes.
  Siempre que un fabricante incluya un nuevo elemento de hardware, lo primero que se debe realizar para poder ser utilizado desde Android, es crear las librerías de control o drivers necesarias dentro de este kernel de Linux embebido en el propio Android.
  La elección de Linux 2.6 es debida principalmente a dos razones: la primera: su naturaleza de código abierto y libre se ajusta al tipo de distribución que se buscaba para Android (cualquier otra opción comercial disponible hoy en día, hubiera comprometido la licencia de Apache); la segunda: este kernel de Linux incluye de por sí numerosos drivers, además de contemplar la gestión de memoria, gestión de procesos, módulos de seguridad, comunicación en red y otras muchas responsabilidades propias de un sistemas operativo.

- La siguiente capa corresponde a las librerías utilizadas por Android. Éstas han sido escritas bajo el lenguaje C/C++ y proporcionan a Android la mayor parte de sus capacidades más características. Junto al núcleo basado en Linux, estas librerías constituyen el corazón de Android.

- Al mismo nivel que se encentra la capa de las librerías de Android, se sitúa el entorno de ejecución. Éste lo constituyen las Core Libraries, que son librerías con multitud de clases Java, y la máquina virtual Dalvik.

- Continuando ascendentemente, se encuentra la capa del framework de aplicaciones, que representa fundamentalmente el conjunto de herramientas de desarrollo de cualquier aplicación. Toda aplicación que se desarrolle para Android, ya sean las propias del dispositivo, las desarrolladas por Google o terceras compañías, e incluso las que el propio usuario cree, utilizan el mismo conjunto de API y el

mismo framework representado por este nivel. Estas aplicaciones se escriben sobre lenguaje Java. La tabla 2.2 muestra las API más importantes de esta capa [19][20].

| Nombre | Descripción |
|---|---|
| Activity Manager | Gestiona el ciclo de vida de la aplicación. |
| Window Manager | Gestiona las ventanas de las aplicaciones. |
| Telephone Manager | Gestiona las funcionalidades propias del teléfono (llamadas, mensajes, entre otras). |
| Content Provider | Permite compartir a una aplicación sus datos con las demás aplicaciones. |
| View System | Proporciona elementos para poder realizar interfaces de usuario (GUI) |
| Location Manager | Permite a las aplicaciones obtener información de localización y posicionamiento, y funcionar según ésta. |
| Notification Manager | Permite comunicar al usuario eventos que ocurran durante la ejecución de una aplicación. |
| Package Manager | Permite la recuperación de diversos tipos de información relacionados con los paquetes de aplicaciones que están actualmente instalados en el dispositivo |
| Resource Manager | Permite el acceso a recursos sin código como cadenas localizadas, gráficos y archivos de diseño |

Tabla 2.2: APIs nivel aplicación del framework de Android. Adaptado de Android Developers [19].

- El último nivel del diseño arquitectónico de Android son las aplicaciones. Éste nivel contiene tanto las incluidas por defecto de Android como aquellas que el usuario vaya añadiendo posteriormente, ya sean de terceras empresas o de su propio desarrollo.
  Todas estas aplicaciones utilizan los servicios, las API y librerías de los niveles anteriores, y son escritas en el lenguaje Java.

### 2.2.3. Librerías Android

Para la realización de la investigación se hará uso de librerías específicas para los periféricos ya mencionados, las cuales permitirán la implementación de pruebas. La tabla 2.3 describe algunas de ellas.

| Librerías | Periférico | Descripción |
|---|---|---|
| **android.location** | GPS | Permite implementación servicios de localización geográfica. |
| **android.hardware** | Acelerómetro y brújula | Proporciona soporte para las características de hardware, tales como el acelerómetro, la brújula, entre otros sensores. |
| **android.media** | Sistema de audio | Proporciona clases que manejan las interfaces de diversos medios de comunicación en audio y video. |

Tabla 2.3: Librerías de Android para periféricos específicos.Adaptado de Android Developers [22] [23] [24] [25].

Además de estas librerías, se hará uso de la librería android.os y en específico la clase batteryManager [24], la cual proporciona características y valores actuales de la batería en el dispositivo móvil

#### Librerías especificas para los periféricos

Android es un sistema operativo que cuenta con un gran número de librerías para el desarrollo de aplicaciones. Para las pruebas pertinentes a esta investigación es necesario la utilización de periféricos específicos y obtención de datos de la batería.

## 2.3. Trabajos Relacionados

Se ha seleccionado un grupo de trabajos relacionados con gestión de consumo de energía en dispositivos móviles que utilizan la plataforma Android y pueden aportar al desarrollo de este trabajo de grado. Para su descripción se hará uso de un patrón que incluye:

- *Nombre:* nombre completo de la publicación.
- *Año:* año de la publicación.
- *Tecnologías:* tecnologías y dispositivos de soporte.
- *Aportes:* aportes significativos relacionados con el objetivo de este trabajo de grado.

Después de revisar los trabajos relacionados, se concluirá acerca de las de brechas encontradas, para establecer el alance esta investigación.

### 2.3.1. Decomposing power measurements for mobile devices [26]

***Año:*** 2010.
***Tecnologías:*** 2G, 3G, WIFI.
***Aportes:*** Analiza de consumo de batería en envió de mensajes en WIFI modificando el tamaño del buffer. Analiza las diferencias de consumo de energía de reposo (sin estar ejecutando aplicaciones) entre 2G, 3G y WIFI por motivos de sincronización con las redes. Implementa un HW externo al del dispositivo para la medición de energía consumida.

### 2.3.2. GAC: Energy-Efficient Hybrid GPS-Accelerometer-Compass GSM Localization [3]

***Año:*** 2010.
***Tecnologías:*** EnLoc, GPS, Acelerómetro, Brújula.
***Aportes:*** Analiza consumo de energía de un sistema hibrido GAC (GPS, acelerómetro y brújula). Compara el sistema GAC con los sistemas EnLoc y GPS. Implementa el sistema GAC para disminuir el consumo de batería sustituyendo el GPS convencional del celular.

### 2.3.3. Power Saving in Mobile Devices Using Context-Aware Resource Control [4]

***Año:*** 2010.
***Tecnologías:*** GPS, Cámara(Gamaray y ARToolKit), Procesador.
***Aportes:*** Investiga sobre como el contexto en el que se encuentre el móvil, puede ayudar a ahorra batería. Utiliza un controlador de recursos y un algoritmo para crear curvas de QoE dependiendo del contexto (Interior/Exterior, Caminar/Estático) y así asignar los recursos necesarios a cada aplicación. Presenta una disminución de 45 % en el consumo de batería al implementar una técnica basada en contextos.

### 2.3.4. Android on Mobile Devices: An Energy Perspective [27]

***Año:*** 2010.
***Tecnologías:*** Dalvik VM, VM Sun JAVA(sin JIT), VM Sun JAVA(con JIT).
***Aportes:*** Compara la VM dalvik de Android con la VM Sun Java respecto al consumo de batería de las aplicaciones. Utiliza 3 procedimientos para comparar Dalvik VM (sin JIT) y Sun JAVA VM (con y sin JIT).

### 2.3.5. Improving Android Performance and Energy Efficiency [9]

***Año:*** 2011.
***Tecnologías:*** ARM, DSP, Dsplink, VM Dalvik.
***Aportes:*** Utiliza un marco paralelo entre el ARM y el DSP en las aplicaciones para android para mejorar el consumo de energía. Utiliza un algoritmo para dividir tareas entre los procesadores (ARM y DSP) utilizando hilos y compartiendo memoria. Implementa para la comunicación entre los procesadores a dsplink.

### 2.3.6. Batch Scheduling of Recurrent Applications for Energy Savings on Mobile Phones [28]

***Año:*** 2010.
***Tecnologías:*** Utilizaron varias aplicaciones al tiempo.
***Aportes:*** Establece un lote de programación (batchscheduling) sobre aplicaciones recurrentes para poder realizar un ahorro de energía. Implementa un programa para reutilizar los recursos del teléfono y así, si varias aplicaciones requieren de un periférico del celular al mismo tiempo lo hagan de forma compartida.

### 2.3.7. Improving Energy Efficiency of Wi-Fi Sensing on Smartphones [29]

***Año:*** 2011.
***Tecnologías:*** WI-FI, Acelerómetro.
***Aportes:*** Desarrolla un algoritmo de detección a partir de un modelo probabilístico utilizando la distribución de Poisson sobre WI-FI llamado WIFIsense, con el cual disminuyen el consumo de energía de exploración hasta un 79 % y el falso disparo hasta un 4.3 %. Utiliza el acelerómetro para inferir el movimiento del usuario y así calcular la frecuencia de detección. Introduce algoritmos de detección de disparo para optimizar la conexión WI-FI y reducir la detección accidental.

### 2.3.8. Android smartphone: Battery saving service [30]

***Año:*** 2011.
***Tecnologías:*** WI-FI, GPS, Bluetooth.
***Aportes:*** Utiliza una aplicación Android llamada PowerTutortool, la cual entrega el uso de energía aproximado y acumula las estadísticas de consumo de energía al ser ejecutada en el dispositivo. Opera aplicaciones de auto-sincronización sobre WI-FI y Bluetooth para encontrar una solución para ahorrar energía.

### 2.3.9. Aportes de trabajos relacionados:

En los trabajos anteriores se encontraron los siguientes aportes que pueden ser significativos para esta investigación.

- El modelo hardware de medición de consumo de energía y las técnicas de análisis utilizadas en el trabajo [26] servirán como referencia para la construcción e implementación de un sistema de medición acorde a los requerimientos de la investigación.

- La utilización de diferentes contextos en que puede utilizarse el dispositivo móvil estudiado en el trabajo [4] nos servirá para la definición de algunos de los escenarios a utilizar en las pruebas.
- La utilización de lotes de programación (Batch-Scheduling) sobre aplicaciones recurrentes en el trabajo [28] es un método interesante que se podrá implementar en el desarrollo de aplicaciones de prueba.

### 2.3.10. Brechas existentes

En los trabajos anteriores se plantean diversos estudios sobre el consumo de energía en diferentes dispositivos móviles soportados en la plataforma android, estos estudios involucran el impacto sobre la energía causado por el uso de redes (WIFI, GSM), sistema GPS y periféricos como acelerómetro y brújula. Además se plantean alternativas para disminuir el consumo en algunos casos puntuales. Tras el análisis de los trabajos se encontraron las siguientes brechas:

- En los trabajos [26] al [29] no se utilizó un administrador de aplicaciones que permitiera desactivar las aplicaciones en segundo plano, lo que no permitía saber el consumo real de la aplicación que se ejecuta en primer plano.
- En el trabajo [3] se hace un estudio sobre el consumo de batería del sistema GAC, pero al utilizar hardware externo para el suministro y medición de energía, no se utiliza la batería real del dispositivo móvil, lo que no permite medir el rendimiento de la batería si no solo el consumo de las aplicaciones.
- En el trabajo [4] se estudió el uso de contextos para asignación de recursos de las aplicaciones pero solo se definieron tres de estos, indicando la necesidad de encontrar muchos más ámbitos de uso para ampliar este estudio.

# Capítulo 3

# Modelo para la evaluación de consumo de energía sobre dispositivos móviles

En este capítulo se propone un modelo para la evaluación del consumo de energía en dispositivos móviles, el cual pretende marcar las pautas, procedimientos y parámetros que son necesarios para realizar una medición adecuada del consumo en los dispositivos móviles y así obtener prácticas de gestión de energía a través de experimentación y análisis.

## 3.1. Descripción General del modelo

El modelo para la evaluación del consumo es una herramienta de referencia para la medición de energía, planificación de pruebas y análisis de consumo de energía. Esta herramienta ha sido diseñada a partir de la observación de las necesidades de medición y evaluación de energía de este trabajo, no corresponde a instanciación alguna de otro referente similar. A continuación se describen los aspectos generales del modelo de la evaluación de energía.

### 3.1.1. Componentes esenciales del modelo

El modelo para la evaluación de energía en un nivel superficial de abstracción consta de 4 componentes esenciales, los cuales dan una perspectiva a las necesidades, procesos y métodos de ejecución y análisis necesarios para la evaluación del consumo de energía en dispositivos móviles. Los componentes son:

- Criterios para la adquisición de medidas (C.A.M.).
- Proceso para la planificación de pruebas (P.P.P.).
- Proceso para la ejecución de pruebas (P.E.P.).
- Proceso para el análisis de mediciones (P.A.M.).

El primer componente describe los parámetros necesarios para la recolección y procesamiento de medidas siendo una base para el desarrollo de herramientas que permitirán medir el consumo de energía. El segundo componente presenta los pasos necesarios para realizar una detallada y eficiente planificación de pruebas. El tercero contiene los procesos que son indispensables para ejecución de las pruebas que han sido previamente planteadas en el componente anterior. El cuarto y último componente contiene el proceso de análisis que se realiza a los resultados de las mediciones, con base en el segundo componente.

La figura 3.1 muestra una visión general del Modelo de evaluación de energía, donde los C.A.M., contenidos en un embudo, dan las pautas que deben cumplir los 3 procesos (P.P.P., P.E.P. y P.A.M.) para garantizar una medición adecuada. Los 3 procesos están representados en 3 engranajes, indicando que existe una dependencia entre los procesos. Partiendo que el P.P.P. es el primer proceso, siendo este necesario para que el P.E.P. pueda ser realizado, y a su vez este segundo proceso determina la realización del P.A.M.

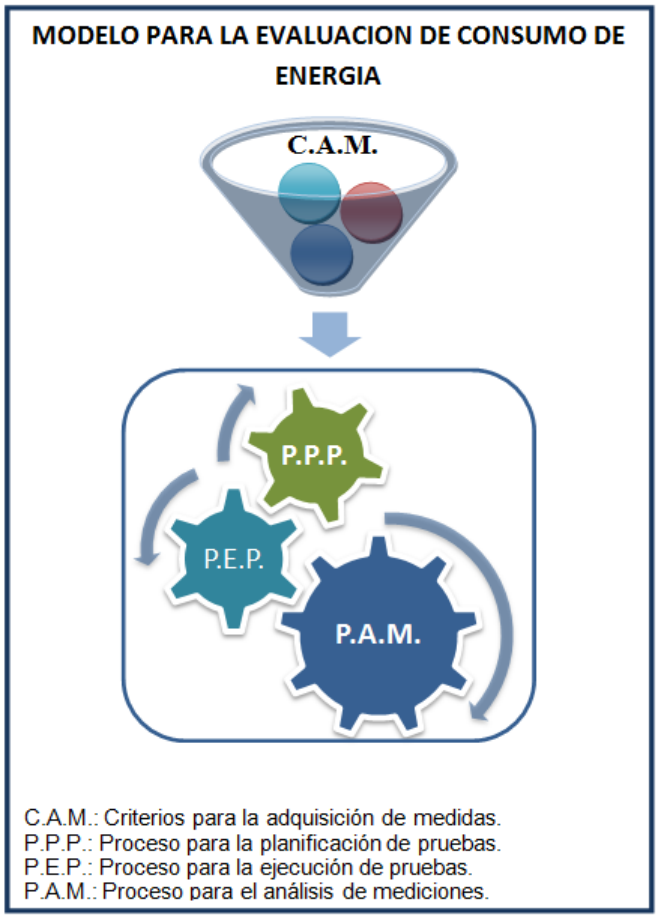


Figura 3.1: Vista general del Modelo Para la Evaluación de Consumo de Energía.

## 3.2. Criterios para la adquisición de medidas (C.A.M.)

Este componente presenta los criterios necesarios para realizar la medición del consumo de energía en dispositivos móviles de cualquier tipo, siendo un referente para la implementación o clasificación de una o más herramientas necesarias para poder realizar la medición.

### Criterios para una medición eficiente

Para realizar una medición eficiente del consumo de energía se debe cumplir con los siguientes criterios:

1. **Precisión:** Las mediciones de voltaje y corriente deben ser precisas para garantizar la confiabilidad de los datos del consumo de energía.


Figura 3.2: Criterios para una medición eficiente.

2. **Frecuencia de medición alta:** El sistema hardware debe permitir realizar mediciones en periodos pequeños para poder analizar mínimos cambios en el consumo de energía.

3. **Sincronización de eventos:** El método de medición debe permitir una sincronización entre los datos entregados y los eventos presentes en la aplicación que está siendo ejecutada en el dispositivo.

4. **Graficación en tiempo real:** Se debe graficar en tiempo real para poder analizar de una forma inmediata cual es consumo que está presentando el dispositivo.

5. **Persistencia de la información:** Los datos que se monitoreen deben ser guardados para su debido procesamiento y posterior análisis.

### Medidas principales

Para la medición del consumo de energía de un dispositivo móvil es necesario medir dos parámetros esenciales, los cuales son:

**La corriente:** Dado que la capacidad de energía de los dispositivos móviles está medida en Ah, la corriente nos permite saber el consumo instantáneo de energía que genera el dispositivo.

**El voltaje:** El nivel de voltaje permite obtener dos medidas, la primera es el porcentaje actual de la batería, y la segunda es la velocidad de descarga que se da de acuerdo al porcentaje de batería.

### Parámetros relacionados con el consumo de energía

Se pueden calcular diferentes parámetros que ayudarán a analizar el consumo de energía a partir de las medidas realizadas al dispositivo en estudio, estos parámetros son el porcentaje de batería, promedio de corriente, promedio de potencia y tiempo de duración de la batería.

**Porcentaje de batería:** Para calcular el porcentaje de energía restante en una batería es necesario disponer de su curva de descarga, la cual relaciona el nivel de voltaje con el porcentaje de energía actual. La curva de descarga se basa en un fenómeno presente en las baterías, este aparece cuando el voltaje disminuye proporcionalmente a la cantidad de energía almacenada. Existe una restricción para la medición del porcentaje de batería, esta ocurre cuando la batería es conectada a un cargador durante un tiempo cualquiera, esta eleva su nivel de voltaje casi al máximo y parece estar cargada a su totalidad, pero después de unos pocos minutos de descarga, el nivel de voltaje vuelve a bajar, revelando el verdadero porcentaje de energía. Para obtener la curva de descarga se puede documentar el porcentaje de energía mostrado en el dispositivo móvil y el voltaje de la batería durante un ciclo de descarga.

**Promedio de corriente** El promedio de corriente es una medida muy practica para analizar el consumo de un dispositivo móvil, ya que al tener una corriente promediada se puede calcular el tiempo de descarga del dispositivo y el impacto real si se mantiene este consumo. Para calcular el promedio es necesario obtener suficientes muestras de corriente en un intervalo dependiendo del objetivo de la medición y así obtener un valor lo mas aproximado posible al consumo de corriente real. El cálculo del promedio se puede realizar con la ecuación 3.1 donde *Ipro* es el promedio de corriente, *n* es el numero de muestras tomadas e *i* es el vector con las muestras de la corriente.

$$Ipro = \frac{1}{n}\sum_{j=0}^{n-1} i[j][mA] \tag{3.1}$$

**Promedio de potencia** El promedio de potencia se hace con base en las muestras de corriente y voltaje tomadas en cada instante, se puede calcular con la ecuación 3.2, donde *Ppro* es el promedio de potencia, *n* es el numero de muestras tomadas, *i* es el vector con las muestras de la corriente y *v* es el vector con las muestras del voltaje.

$$Ppro = \frac{1}{n}\sum_{j=0}^{n-1} i[j] * v[j][W] \tag{3.2}$$

**Tiempo de duración de la batería** Para establecer el tiempo de duración de la batería se hace uso de la capacidad (carga eléctrica almacenada) y el promedio de corriente que se ha encontrado. En la ecuación 3.3 se puede observar el tiempo, donde *Tbat* es el tiempo de duración de la batería.

$$Tbat = \frac{capacidad}{Ipro}[h] \tag{3.3}$$

### Métodos para la sincronización de eventos

La sincronización de eventos solo se puede realizar mediante la aplicación que va a ser ejecutada en el dispositivo. A continuación se presentan dos métodos posibles de sincronización.

- Picos de consumo: Los picos de consumo se utilizaran como método para identificar eventos a través del consumo en una aplicación. Hay dos formas de generar picos de consumo: la primera es llamada picos de consumo de bajada, esta se presenta cuando hay un nivel alto de consumo y es disminuido a un nivel bajo; la segunda es llamada picos de consumo de subida, esta se presenta cuando hay un nivel bajo de consumo y es elevado a un nivel alto.

- Marcas con retardos: Se colocan retardos en diferentes partes de la aplicación lo que va a producir una disminución en el consumo de energía.

Al graficar el consumo se puede sincronizar los eventos observando los picos o las marcas de sincronización. Los métodos de sincronización pueden afectar las mediciones en el caso de medir el consumo promedio en mAh de una aplicación, debido a que pueden generar un consumo adicional.

## 3.3. Proceso para la planificación de pruebas (P.P.P.)

Este proceso presenta los diferentes pasos para la creación, ejecución y toma de decisiones en pruebas de consumo de energía, garantizando la recolección de información para un análisis adecuado de los resultados. El proceso consta de 2 subprocesos: Caracterización del consumo actual y Diseño de pruebas.

### Caracterización del consumo inicial

Como primer paso para el diseño de pruebas, es necesario tener una idea general y actual del comportamiento de las aplicaciones respecto al consumo de energía sobre el dispositivo en estudio. Los pasos que se seguirán para cada prueba inicial se pueden observar en la figura 3.3, estos se describen a continuación.

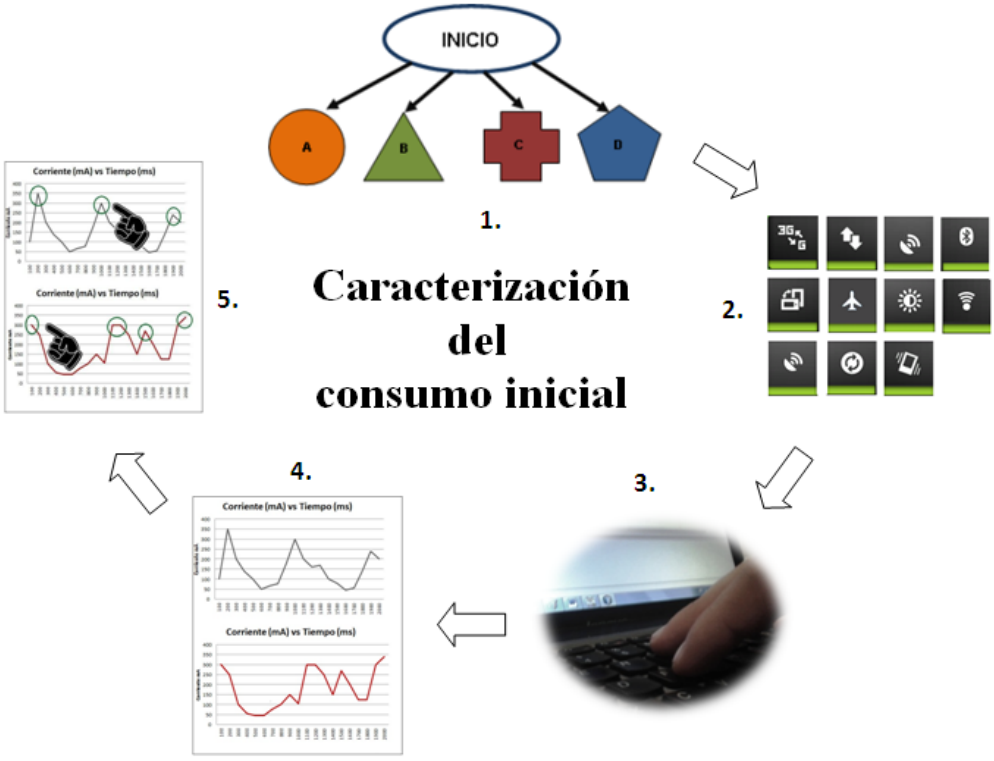


Figura 3.3: Caracterización del consumo inicial.

1. **Selección de algoritmo:** Para empezar el estudio se seleccionan los algoritmos de programación más utilizados por los desarrolladores sobre una función del dispositivo, para posteriormente evaluar cual de estos tiene un menor consumo de energía.

2. **Selección de escenario inicial:** Como segundo paso es escoge el escenario inicial en el que se va a medir el consumo de la aplicación con el algoritmo actual.

3. **Implementación inicial**: Se implementa una aplicación que utilice el/los algoritmos de programación existentes, la cual se ejecuta en el escenario anteriormente descrito.

4. **Medición del consumo inicial:** Se debe realizar la medición del consumo de energía de la aplicación anterior, obteniendo todos los posibles datos del consumo actual.

5. **Análisis del consumo inicial**:Se observan las gráficas de consumo y se analizan los picos de consumo que presentan un impacto significativo con respecto al promedio de consumo.

**Diseño de pruebas**


Figura 3.4: Diseño de pruebas.

Con los resultados que arroja la caracterización del consumo inicial, se obtiene la información necesaria para realizar el respectivo diseño de las pruebas. El objetivo de estas pruebas es plantear hipótesis de acuerdo a un periférico, una practica de programación y un escenario específico que se escoja.
Los pasos para el diseño de pruebas se describen a continuación y se pueden observar en la figura 3.4.

1. **Definición de la práctica de programación a evaluar:** Se debe definir las prácticas de programación que se implementarán con el fin de disminuir el consumo de energía, y las posibles variaciones para la ejecución de las pruebas.

2. **Especificación del escenario:** Especificar las condiciones en las cuales se va a realizar la prueba, como lo son: lugar, periféricos activados, estado de conexión a redes, entre otras.

3. **Planteamiento de hipótesis:** Especificar el posible resultado que se podría obtener al realizar la prueba. Este paso no es obligatorio pero puede ayudar a un mejor análisis de los resultados.

4. **Descripción software o aplicación:** Descripción del software o la aplicación que se va a utilizar al diseñar la prueba, basado en las prácticas de programación definidas en el numeral 1.

5. **Implementación del software:** Se implementa una aplicación siguiendo la descripción hecha y tomando como base la aplicación implementada en la caracterización del consumo inicial.

6. **Sincronización de eventos:** Se agregan los mismos puntos y/o marcas de sincronización utilizados en la caracterización del consumo actual.

## 3.4. Proceso para la ejecución de pruebas (P.E.P.)

Este proceso para la ejecución de pruebas se basa en el diseño de pruebas del P.P.P. descrito en el literal 3.3. En este proceso se ejecutarán las pruebas con un mínimo de repeticiones de 5 veces para cada una, para así verificar el comportamiento del consumo obtenido de cada prueba.

Los pasos para la ejecución de las pruebas de medición se describen a continuación y se pueden observar en la figura 3.5.

1. **Caracterización del consumo inicial:** Se realiza una prueba de medición de energía basándose en los escenarios planteados en el diseño de pruebas, y utilizando la aplicación de control para obtener el patrón de consumo actual, que servirá como punto comparativo para evaluar el consumo de la práctica de programación diseñada.
2. **Ejecución inicial:** Se realiza la medición del consumo de energía bajo el escenario descrito y corriendo la aplicación de prueba.
3. **Ejecución secundaria:** Se repiten las mediciones con la aplicación de prueba variando los parámetros especificados en el proceso de planteamiento de pruebas, con el fin de disminuir el consumo de energía.

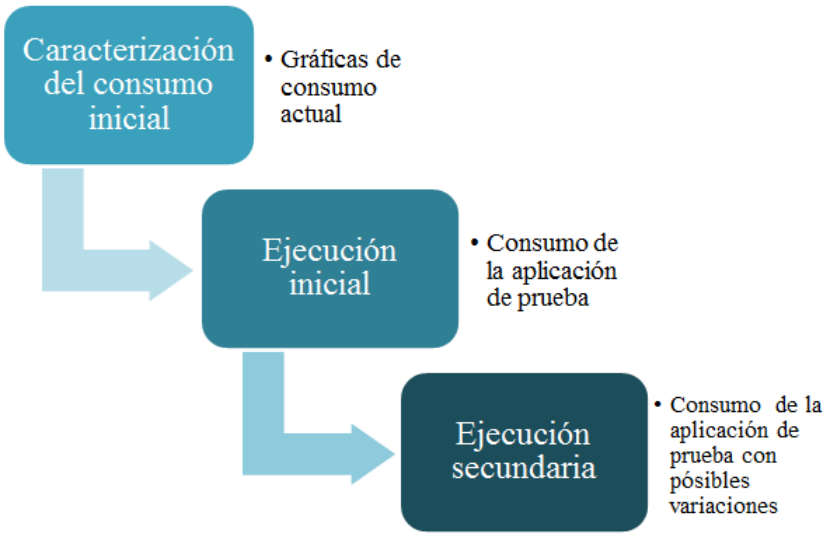


Figura 3.5: Proceso para la ejecución de pruebas (P.E.P.).

## 3.5. Proceso para el análisis de mediciones (P.A.M.)

El proceso para el análisis de mediciones contiene las pautas necesarias para el análisis y toma de decisiones respecto al planteamiento de nuevas pruebas de medición. Para el análisis se utilizarán las mediciones realizadas representadas en gráficas, con el fin de facilitar la visualización de resultados y así poder realizar una comparación entre las medidas tomadas a la aplicación de prueba y sus variaciones, y las medidas tomadas a la aplicación de control. Las gráficas que se utilizarán principalmente serán:

- Corriente vs Tiempo.

- Potencia vs Tiempo.


Figura 3.6: Proceso para análisis de mediciones (P.A.M.).

1. **Análisis comparativo:** Se analizarán los puntos de mayor consumo en la aplicación de control, comparándolos con el consumo en las aplicaciones de prueba. Adicionalmente, se comparará el consumo de energía promedio en mAh para verificar la efectividad de la práctica.

2. **Verificación de hipótesis:** Se analizará si se cumplió o no con la hipótesis planteada en el proceso de planificación, si fue planteada.

3. **Conclusiones:** Se concluirá si la práctica probada es una práctica de gestión de energía valida, sino se tomarán decisiones para la posible creación de una nueva prueba para obtener una práctica adecuada.

4. **Creación de alternativas:** Se planteará si existen más alternativas para mejorar la gestión de energía, con el objetivo de planificar nuevas pruebas.

# Capítulo 4

# Herramienta para la medición de consumo de energía en dispositivos móviles

En el presente capitulo se analizan las alternativas de medición de consumo de energía planteadas en trabajos relacionados con el fin de diseñar una herramienta que se adapte a las necesidades de medición del consumo de los periféricos y al modelo de evaluación plateado en el capítulo 3.

## 4.1. Herramientas existentes

Se describen a continuación las herramientas y métodos de medición de consumo utilizados en los trabajos relacionados mencionados en el capítulo 2 con el fin de identificar las características que podrían replicarse en la construcción de una herramienta.

### 4.1.1. Medidor Implementado en Tarjeta PCI-MIO-16E-4

En [26] Rice et al. implementan una herramienta para la medición sobre una tarjeta PCI-MIO-16E-4, la cual consta de dos etapas:

#### Etapa 1: Circuito medidor de voltaje y corriente

Permite medir el consumo de corriente del dispositivo con alta precisión mediante la inserción de resistencia de medición $0.02\Omega$, entre la batería y el conector del dispositivo móvil. Además utiliza un amplificador diferencial para amplificar el valor de voltaje que cae en la resistencia a un nivel que pueda ser detectado por un conversor analógico digital con el objetivo de calcular la corriente. La inserción de la resistencia de medición aumenta la impedancia del circuito, por lo tanto su consumo de energía en un porcentaje aproximado es del 1 %, lo cual no afecta significativamente la medición. Para medir el voltaje se conecta el ADC (Conversor Analógico Digital) en paralelo a la batería, como se muestra en la figura 4.1.

#### Etapa 2: Procesamiento de datos

La información del consumo del dispositivo se obtiene mediante la programación de una tarjeta PCI-MIO-16E-4 4.2 de National Instruments conectada a un computador. La tarjeta contiene un conversor analógico digital de 12 bits, con el cual se obtiene los datos de voltaje y corriente entregados por la etapa 1 para posteriormente ser guardados.

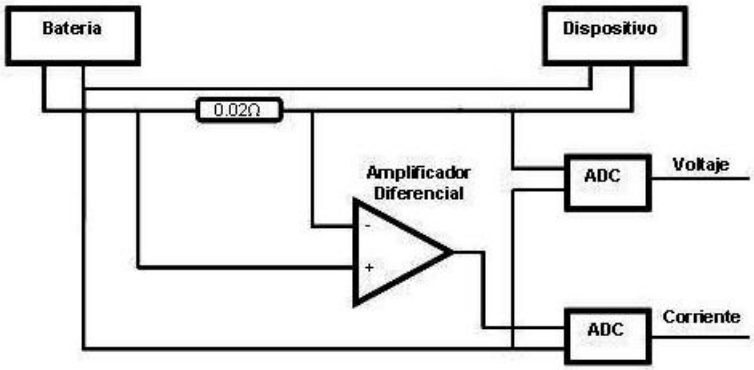


Figura 4.1: Circuito para la medición de voltaje y corriente. Adaptado de [26].

Figura 4.2: Tarjeta PCI-MIO-16E-1 [26].

### 4.1.2. Power monitor

En [3] Youssef et al. utilizan el *Power monitor*, el cual es un dispositivo de alimentación y medición compuesto por hardware y software distribuido por *Monsson Solution.inc*. Este proporciona una solución de medición de potencia en dispositivos móviles. Para medir la potencia consumida por el dispositivo se reemplaza la batería por el *power monitor*, el cual suministra la potencia necesaria, es decir, funciona como fuente de alimentación y mide la corriente instantánea y voltaje utilizado por el dispositivo móvil.Dado que la tensión es constante, la corriente puede ser utilizada como un indicador del consumo de energía. El *power monitor* está conectado a un ordenador por medio de una conexión USB, controlada por una aplicación cliente la cual recibe los valores y los almacena. La conexión del *power monitor* se puede observar en la figura 4.3.

### 4.1.3. PowerTutortool

En [30] Zahid et al. realizan una medición utilizando una aplicación Android corriendo en el dispositivo llamada *PowerTutortool*, la cual entrega el uso de energía aproximado y acumula las estadísticas de consumo de energía. La aplicación se ejecuta en la mayoría de los teléfonos Android, aunque está diseñado para dar cifras exactas sobre el *G1*, *G2*, y los teléfonos *Nexus One*. Esta aplicación registra los antecedentes sobre la utilización de energía para cada aplicación, información se resume en una interfaz de usuario intuitiva como se puede ver en la figura 4.4.

Figura 4.3: Power monitor [3].

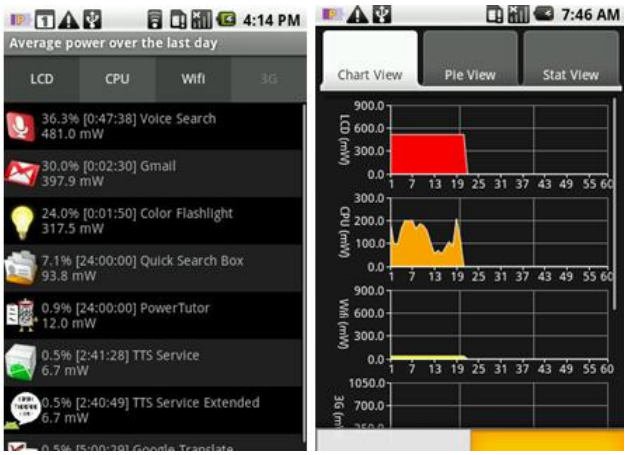


Figura 4.4: PowerTutortool [30].

### 4.1.4. Análisis de las Herramientas existentes

Las herramientas utilizadas en investigaciones previas sobre el consumo de energía presentan las siguientes características:

- Información de mediciones en tiempo real.
- Información desplegada en gráficas
- Medición del consumo en mAh.
- Conectividad con un computador.
- Persistencia de la información.
- La medición se puede hacer con hardware externo o software corriendo en el dispositivo.

Estas características fueron tenidas en cuenta para el diseño e implementación de la herramienta utilizada para este proyecto, siendo así una gran ayuda para el desarrollo del mismo. Adicional mente se tienen las siguientes observaciones sobre las herramientas:

- La herramienta 1 presenta una etapa que permite la medición de corriente con alta precisión y bajo consumo de energía.

- El *power monitor* y la tarjeta PCI-MIO-16E-4 presentan costo elevado para la investigación.
- El *power monitor* funciona como fuente de alimentación permitiendo remplazar la batería, bajo el supuesto de que el voltaje debe ser constante, lo que es erróneo, ya en las baterías disminuyen su nivel de tensión a medida que se descarga. Por tanto los datos de medición pueden variar para un escenario con la batería real.
- La herramienta 3 es un software corriendo sobre el dispositivo, lo que provoca un consumo de energía propio y conlleva problemas para identificar el verdadero consumo de una aplicación específica.

## 4.2. Diseño de herramienta para la medición de energía

Con base en los Criterios para la adquisición de medidas descritos en el apartado 3.2 y algunas características de las herramientas existentes, se diseñó e implementó una herramienta para la medición de energía .

### 4.2.1. Características de la herramienta

- Medición físicas de corriente y voltaje que la batería entrega al dispositivo móvil.
- Control de inicio y terminación de las mediciones.
- Persistencia de la información en archivos planos.
- Desplegar gráficas de las mediciones en tiempo real.

### 4.2.2. Componentes del sistema

La estructura funcional de la herramienta se divide en tres módulos como se muestra en el diagrama de componentes de la figura 4.5.

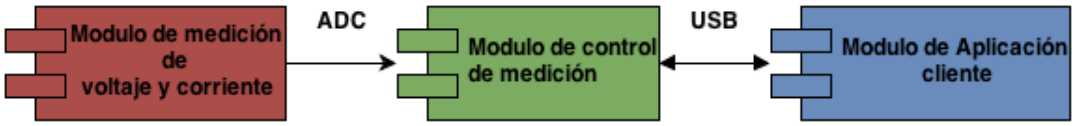


Figura 4.5: Diagrama de componentes.

**Modulo de medición de voltaje y corriente:** Compuesto por hardware, se encarga de medir corriente y voltaje, y adaptar el nivel para ser convertido por el ADC.

**Modulo de control de medición:** Este modulo hardware/software se encarga de obtener los datos de medición de forma periódica por medio del ADC y procesarlos para evitar errores de medición. Además, recibe peticiones vía USB de la aplicación cliente localizada en el computador para iniciar la medición y retornar esta información por este mismo medio.

**Modulo de Aplicación cliente:** Contiene la interfaz de usuario para el manejo de las mediciones. Realiza peticiones al control de medición para obtener los valores medidos, agregándolos a una tabla y graficándolos dinámicamente. También permite guardar la información de las tablas en archivos Excel y las gráficas en archivos con formato de imagen (.jpg).

### 4.2.3. Despliegue del sistema

En la figura 4.6 se muestra el diagrama de despliegue del sistema donde se especifica la interacción del hardware, donde el "equipo en uso" corresponde a un computador y el modulo de aplicación cliente corresponde al archivo Medidor_bateria.jar.

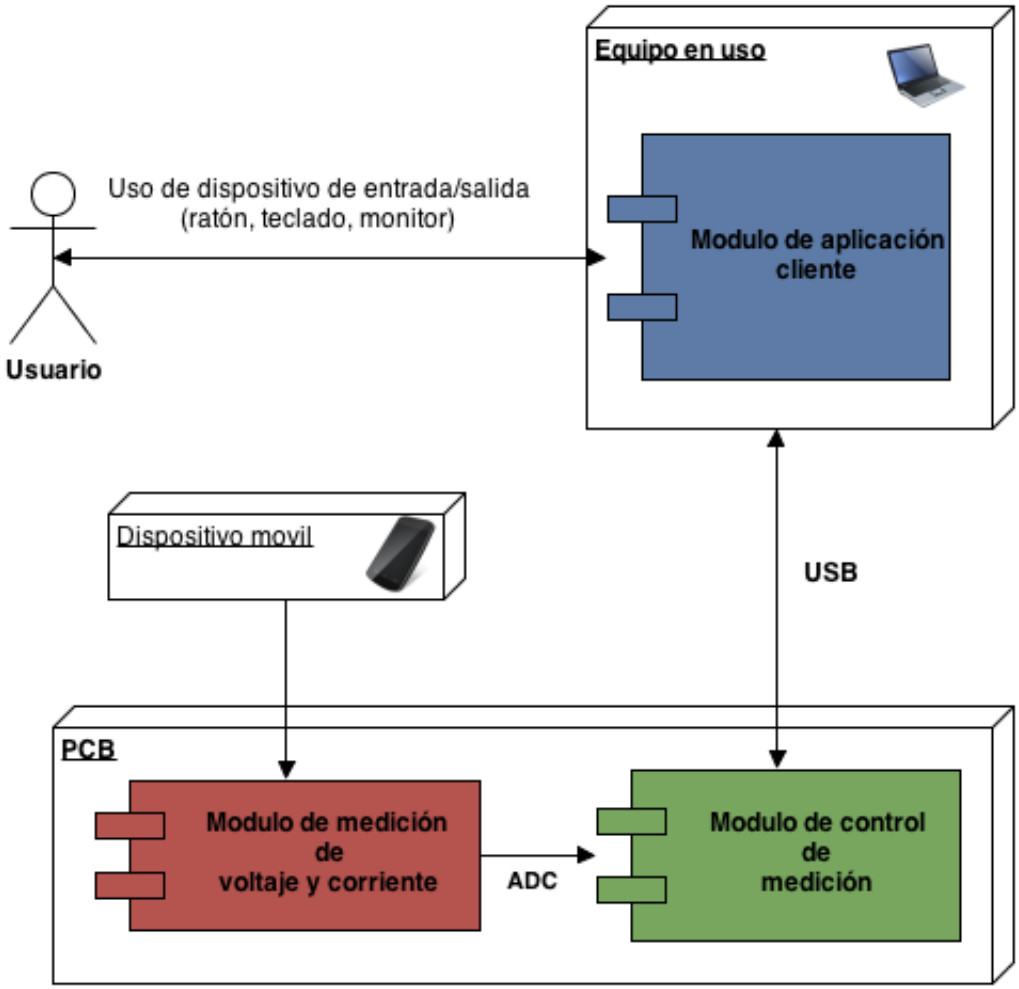


Figura 4.6: Diagrama de despliegue.

## 4.3. Diseño hardware

### 4.3.1. Modulo de medición de voltaje y corriente

Para este modulo se utilizó la etapa de medición de voltaje y corriente del dispositivo medidor implementado en la tarjeta PCI-MIO-16E-4 descrito en el numeral 4.1, agregando dos resistencia de 100kΩ en serie paralelas a la batería, con el fin de disminuir el consumo de corriente que puede generar el ADC, así como se muestra en la figura 4.7.
En la configuración de la figura 4.7 se utiliza un amplificador operacional para brindar una ganancia al voltaje de la resistencia *Rm* (resistencia de muestreo).
Los voltajes entregados a los canales ADC son *VBAT/2* y *VRI*, donde *VBAT/2* es el voltaje entregado por el divisor de voltaje, y *VRI* es el voltaje que entrega el amplificador operacional que corresponde al voltaje *VR*,

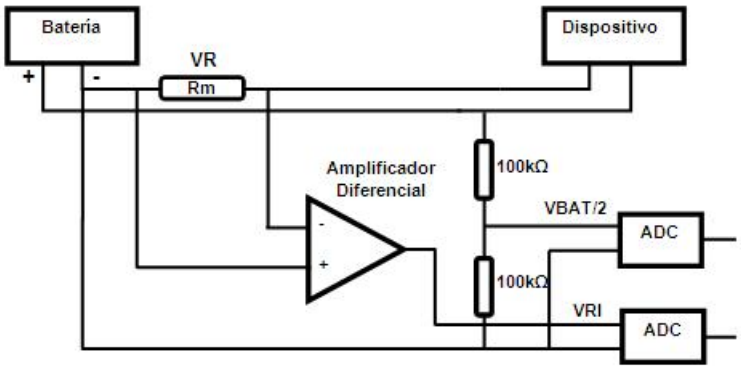


Figura 4.7: Configuración hardware modulo medición voltaje y corriente

que es multiplicado por la ganancia *Av* así como se observa en la siguiente expresión.

$$VRI = VR * Av \tag{4.1}$$

*VR* es el voltaje medido en la resistencia de muestreo con el cual se puede calcular la corriente que entrega la batería al dispositivo. Utilizando la ley de OHM:

$$Ic = \frac{VR}{Rm} \tag{4.2}$$

Y al despejar *VR* de la 4.1 y remplazándolo en la ecuación 4.2, se obtiene la corriente entrante *Ic* en función de *Av* y *VRI*:

$$Ic = \frac{VRI}{Rm * Av} \tag{4.3}$$

**Selección y cálculo de componentes**

Los componentes a seleccionar en el modulo de medición de voltaje y corriente son Rm que corresponde a la resistencia de muestreo y el Amplificador operacional (A.O.).

**Resistencia de muestreo *Rm*** La resistencia de muestreo es una resistencia que se encuentra en serie al dispositivo móvil a medir, esta resistencia debe afectar lo menos posible las mediciones generando un consumo muy pequeño de potencia, por tanto debe tener un valor muy pequeño en relación a la impedancia del dispositivo. La impedancia de un dispositivo móvil es variable debido a que su consumo varia, la tabla4.1 se puede observar algunos consumos aproximados sobre un dispositivo en diferentes condiciones de uso, las medidas están expresadas en mAh. En base a estos valores se considera: como mínimo de consumo 2mAh y

| Uso | Consumo |
|---|---|
| Modo avión | 2mAh |
| Lcd normal | 100mAh |
| Cpu 50-100% | 170mAh |
| Sensores en juego | 240 mAh |
| GPS | 250 mAh |
| WIFI trasferencia máxima | 450mAh |

Tabla 4.1: Consumo dispositivo móvil. (Adaptado de [11]).

como máximo 800mAh (tomando la suma de los mayores consumos, los cuales son WIFI en transferencia máxima, el GPS y la pantalla LCD encendida). Teniendo en cuenta que la batería de Li-ion entrega un voltaje entre 3.5V y 4.2V, y se tiene por ley de ohm que la impedancia del dispositivo puede variar entre 4,375Ω y 2100 Ω. Dado el intervalo para el valor de impedancia se buscaron resistencias menores a 1Ω, donde se encontraron en el mercado nacional resistencias de 0.11Ω como las de menor valor, por tanto se realiza un arreglo de 4 resistencias iguales de 0.11Ω a medio vatio configuradas en paralelo, logrando una resistencia de 0.0275Ω, y así permitiendo a su vez evitar el calentamiento y disminuir las variaciones por causa de la temperatura al dividir la corriente en 4 partes. El potencia disipada por esta resistencia con respecto a la corriente máxima y mínima se encuentra entre 110nW y 17,6mW.

**Amplificador Operacional:** Se selecciona el AD620 el cual es un amplificador de instrumentación que ofrece una ganancia entre 0 y 1000 dependiendo de una resistencia externa. La ganancia se puede calcular por medio de la ecuación 4.4:

$$Av = \frac{49,4K\Omega}{RG} + 1 \tag{4.4}$$

Donde *AV* es la ganancia y *RG* es la resistencia externa que la controla. Para utilizar este amplificador se necesita una fuente dual de +5 a -5v.

**Ganancia A.O.** Con una corriente de 2mA el voltaje en la resistencia *Rm* corresponde 55$\mu$V y con una corriente 800mA corresponde a 22mV que corresponde a un voltaje de entrada máximo. Dada la resolución del ADC de 4,88mv se debe amplificar el valor mínimo de voltaje a un valor mayor, teniendo en cuenta que el ultimo bit del ADC suele ser inestable se amplifica hasta 2 o 3 veces el de la resolución, es decir a 9,77mV o 14,64mv respectivamente. Con estos valores se calcula el intervalo en el cual puede estar la ganancia:

$$Av1 = \frac{Vo}{vi} = \frac{9,77mV}{55\mu V} = 177,63 \tag{4.5}$$

$$Av2 = \frac{Vo}{vi} = \frac{14,64mV}{55\mu V} = 266,18 \tag{4.6}$$

El intervalo de ganancia se encuentra entre 177,63 y 266,18. Dada esta ganancia se debe calcular el voltaje máximo que entregaría el A.O. al ADC, el cual debe ser menor a 5V ya que es el voltaje de referencia. Por lo tanto se calcula el valor máximo de la siguiente manera:

$$Vmaximo1 = Av1 * vi = (177,63) * (22mV) = 3,90V$$
$$Vmaximo1 = Av1 * vi = (266,63) * (22mV) = 5,85V$$

Al analizar el anterior intervalo se puede observar que con *Av2* el voltaje supera al de referencia, por lo que se selecciona una ganancia entre las 2 anteriores y que genere un *Vmaximo* menor a 5V. Dadas estas condiciones se puede calcular la ganancia tomando como voltaje de salida máximo de 5v y voltaje de entrada máximo de 22mv.

$$Avmaxima = \frac{Vmaximo}{vi} = \frac{5V}{22mV} = 227,27$$

Remplazando esta ganancia en la ecuación 4.4 se obtiene que *RG* es igual a 218,76$\omega$ que se puede aproximar a un valor comercial de 220$\omega$ y obtener una ganancia es 225 y un voltaje máximo a medir de 4.95V.

$$RG = \frac{49,5K\Omega}{(Av - 1)} \tag{4.7}$$

### 4.3.2. Módulo de control de medición

Para el control de la medición es necesario un microcontrolador que permita enviar información de una forma rápida y cuente con un ADC (Conversor Analógico Digital) de buena precisión, este se puede encontrar en la familia de microcontroladores 18Fxx5x de Microchip, que es la única que contiene una interfaz serie USB full-speed 2.0 y 1.0, que permite la comunicación rápida entre cualquier dispositivo USB y el microcontrolador. En la tabla 4.2 se muestran las características a analizar de la familia 18fxx5x.

| CARACTERÍSTICAS | PIC18F2455 | PIC18F2450 | PIC18F4455 | PIC18F4550 |
|---|---|---|---|---|
| Frecuencia de Operación | DC - 48MHZ | DC - 48MHZ | DC - 48MHZ | DC - 48MHZ |
| Memoria de programa | 24576 | 32768 | 24576 | 32768 |
| Memoria RAM de Datos(bytes) | 2048 | 2048 | 2048 | 2048 |
| Memoria EEPROM Datos(bytes) | 256 | 256 | 256 | 256 |
| Interrupciones | 19 | 19 | 20 | 20 |
| Líneas E/S | 24 | 24 | 35 | 35 |
| Temporizadores | 4 | 4 | 4 | 4 |
| Canales de conversión A/D de 10 bits | 10 | 10 | 13 | 13 |
| Canales USB | 1 | 1 | 1 | 1 |
| Comparadores analógicos | 2 | 2 | 2 | 2 |

Tabla 4.2: Características 18fxx5x basado en data sheet 18F2455/2550/4455/4550.

Con base en las características de los microcontroladores de esta familia se escogió el 18F4550 por presentar una mayor cantidad de memoria de programa, mayor cantidad de líneas de E/S y poseer canales analógicos de 10 bits de resolución. Adicionalmente es el más utilizado para la comunicación USB, encontrándose mayor cantidad de documentación. La figura 4.8 muestra el diagrama de conexión del microcontrolador para el funcionamiento del puerto USB.

Dadas las características de medición del microcontrolador es necesario utilizar un voltaje de referencia externo que permita intercambiar el voltaje de referencia en caso de necesitar una mayor precisión en el ADC a la que ofrece los 5V de la referencia interna del microcontrolador, para esto se eligio el integrado LM336-2.5, el cual brinda un voltaje preciso de 2,5V. La figura 4.9 muestra la conexión del integrado LM336-2.5 compensado en temperatura para evitar variaciones.

La resolución del dispositivo se puede calcular con la ecuación 4.8

$$Resolucion = \frac{Vref}{2^n - 1}[V] \tag{4.8}$$

Donde *Vref* el voltaje de referencia y *n* es el número de bits del ADC y se Obtenido como resultado un resolución de 4.88mv con un de referencia de 5 voltios y de 2.44mv con un voltaje de referencia de 2.5v.

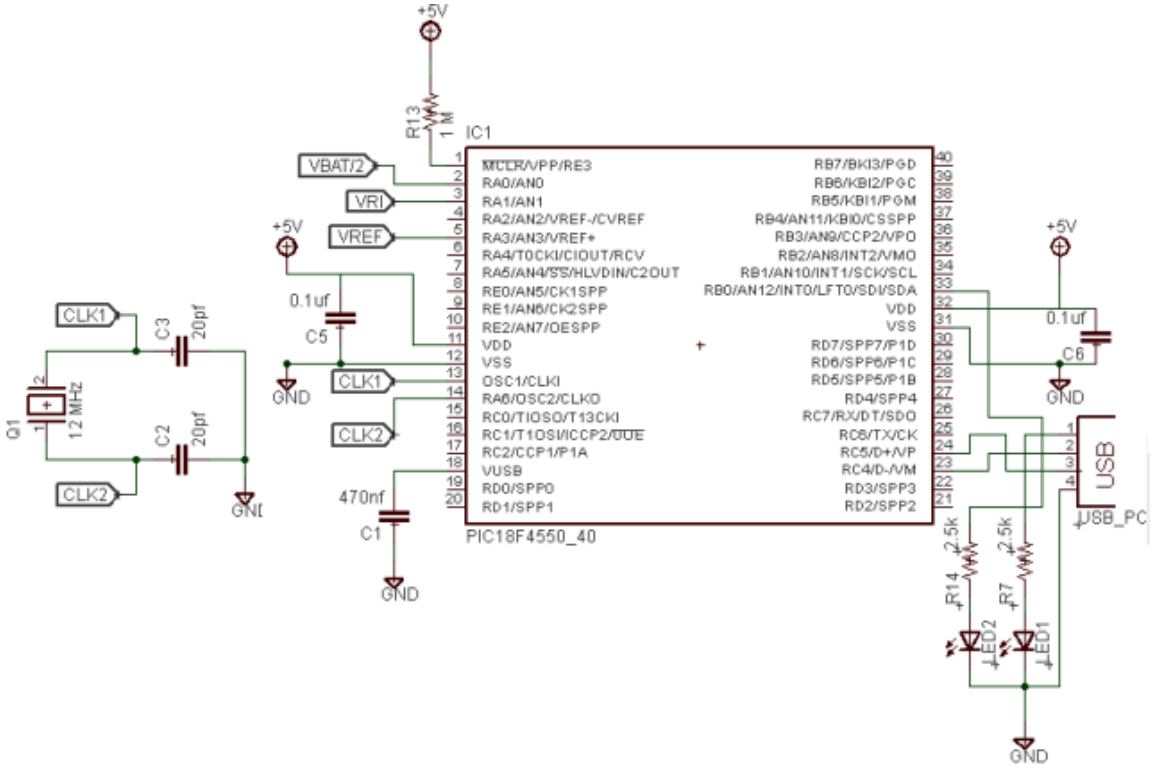


Figura 4.8: Circuito microcontrolador - USB.

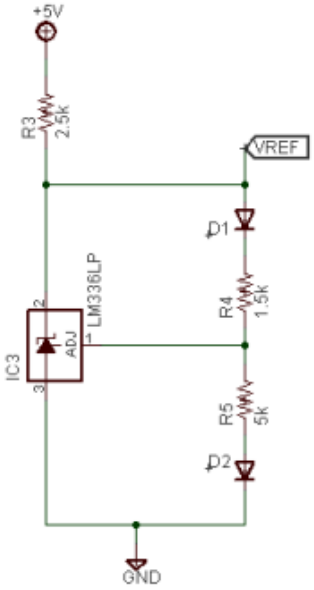


Figura 4.9: LM336-2.5 compensado en temperatura.(Adaptado de [31]).

### 4.3.3. Diseño de tarjeta de adquisición de datos (placa PCB)

Para la implementación del hardware de la herramienta se diseñó una PCB (Printed Circuit Board) en el software Eagle PCB 6.1.0. Eagle es un potente y flexible software de diseño de PCB con una funcionalidad de alto nivel, el cual permite la realización de diseños Hardware de forma eficiente, práctica y rápida [32]. El diseño de la PCB se realizó con el diagrama total de la herramienta de la figura A.1 y la lista de componentes de la tabla A.1, obtiendo una PCB de 8cm x 5,6cm de doble faz. La figura 4.10 muestra el diseño obtenido de la PCB. La PCB cuenta con componentes de la tecnología *through hole (Thru Hole)*, los cuales atraviesan

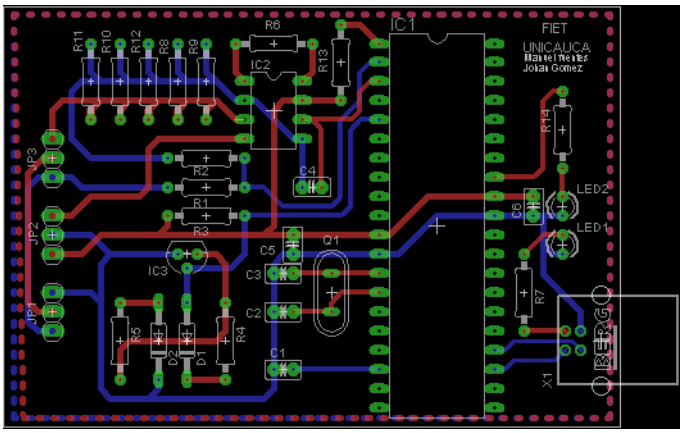


Figura 4.10: Placa PCB diseñada en software Eagle. Ruteo de la capa superior en color rojo y ruteo de la capa inferior en color azul.

la placa de un lado al otro. En la PCB se pueden observar las siguientes partes de la herramienta:

- Voltaje de referencia.
- Módulo de medición de voltaje y corriente.
- Módulo de control de medición.
- Conectores entrada batería dispositivo (JP1).
- Conectores entrada fuente dual (JP2)
- Conectores dispositivo (JP3)

La figura 4.11 muestra la PCB impresa por la capa superior y la figura 4.12 muestra la PCB impresa por la capa inferior. La figura 4.13 presenta la PCB impresa y con los elementos soldados.

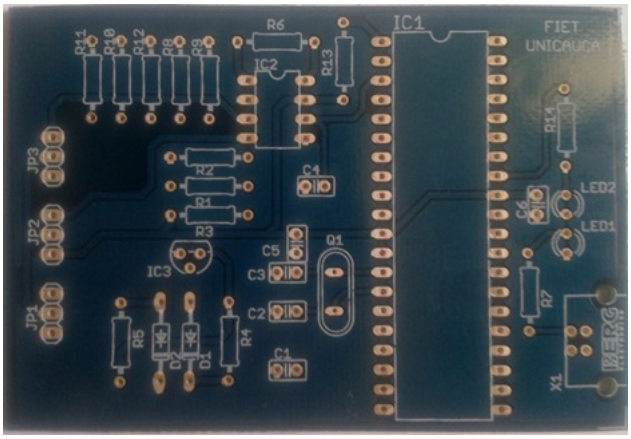


Figura 4.11: Placa PCB impresa - Capa superior.

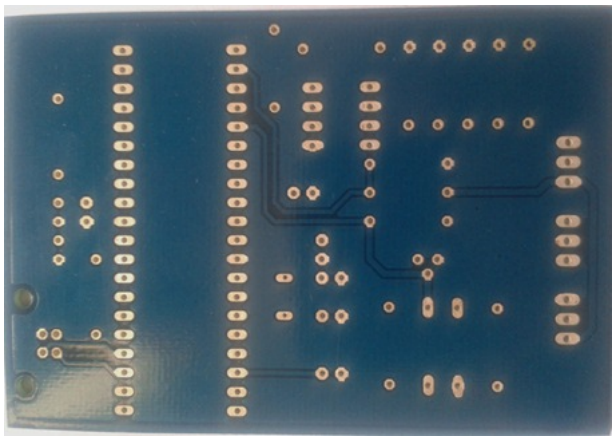

Figura 4.12: Placa PCB impresa - Capa inferior.Placa PCB impresa - Capa superior.

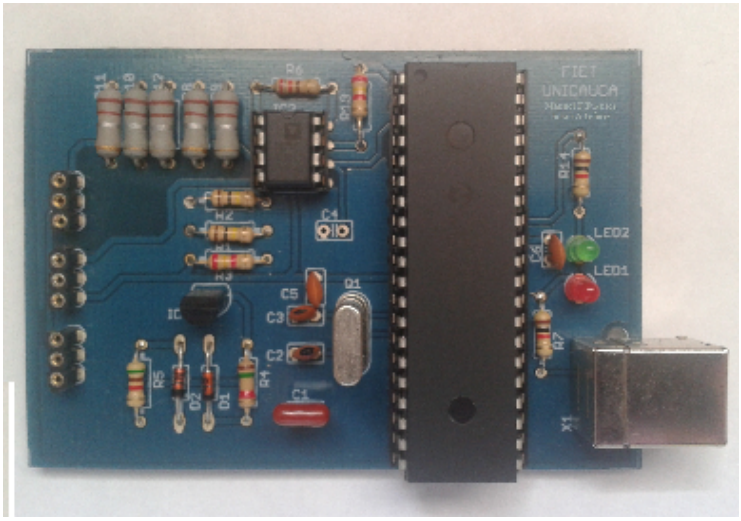


Figura 4.13: Placa PCB impresa con componentes.

### 4.3.4. Dispositivo para la conexión de baterías

se diseña un dispositivo que permite interconectar la batería con la PCB y con el dispositivo móvil, el cual consta de dos partes, la primera es una bateria falsa que remplaza la batería en el dispositivo móvil proveyendo los puertos necesarios para conectar alimentación externa al dispositivo, la segunda es un conector permite conectar la batería del dispositivo móvil de forma externa para realizar la conexión este con la tarjeta PCB, este consta de 2 partes y de 3 conectores. El dispositivo para la conexión de la batería se puede observar en la figura 4.14, la conexión detallada de este dispositivo se especifica en el anexo B.

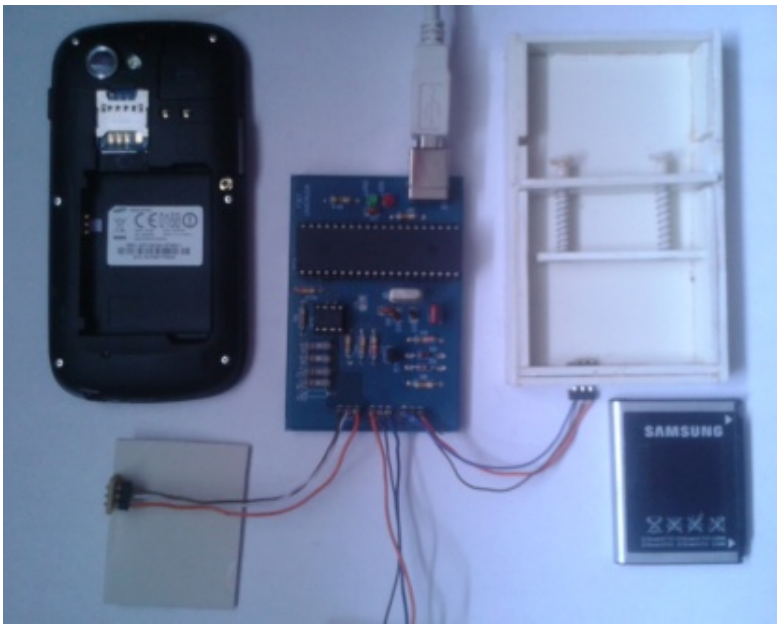


Figura 4.14: Dispositivos para la conexión de baterías.

## 4.4. Desarrollo software

Para los módulos de control de medición y aplicación cliente se implementaron dos aplicaciones. La primera corresponde a la aplicación firmware que se ejecuta en el microcontrolador y la segunda es una aplicación de escritorio que se ejecuta en un PC denominada aplicación cliente.

### 4.4.1. Herramientas de desarrollo

Para la construcción de las aplicaciones se seleccionó el lenguaje C para la construcción de la aplicación del microcontrolador y el lenguaje Java para la aplicación cliente. De acuerdo a esto se seleccionaron los siguientes entornos de desarrollo.

#### PIC-C

Para el desarrollo del programa que corre internamente en el microcontrolador "Firmware"se utilizó el entorno de desarrollo PIC-C basado en CCS. CCS ofrece una suite completa de herramientas integradas para desarrollar y depurar aplicaciones embebidas que se ejecutan en Microchip PIC y dsPIC MCU DSC. La suite de herramientas de código inteligente CCS y la optimización de compilador de C, permite a los

desarrolladores concentrarse en la funcionalidad de diseño en lugar de tener que convertirse en un experto en arquitectura MCU [33].

#### NetBeans IDE 7.1

Es una herramienta de codigo abierto para programadores que permite escribir, compilar, depurar y ejecutar programas en Java. Posee todas las herramientas necesarias para crear aplicaciones profesionales de escritorio, empresariales, web y aplicaciones móviles con la plataforma Java, así como con C / C + +, PHP, JavaScript y Groovy [34].

### 4.4.2. Firmware del microcontrolador

La firmware del microcontrolador se encarga de obtener los valores entregados por el modulo de medición de voltaje y corriente, utilizando un método llamado filtro promedio que procesa los datos del ADC. También se encarga de recibir peticiones del módulo de aplicación cliente por medio de una conexión USB y responder con la información de las medidas.

#### Algoritmo implementado

El algoritmo implementado en el microcontrolador se puede observar en la figura A.2 del anexo A, sus pasos son los siguientes:

1. Creación de variables.
2. Configuración del PIC18F4550.
3. Inicialización del USB.
4. Ciclo infinito.
5. Usbtask habilita el periférico USB e interrupciones.
6. Usbnumerate verifica si el USB del PIC se encuentra configurado.
7. Se realiza la medición por medio del filtro promedio.
8. Se verifica si hay datos entrantes vía USB.
9. Se recepcionan los datos entrantes del ADC en un arreglo de 2 bytes.
10. Se verifica la primera posición del arreglo que es de control, si es 1 se inicializa el filtro con la función iniciofiltro() que llama 10 veces al filtroprom1() y filtroprom2().
11. Si el control es 2 se envían los promedios de los filtro vía USB en un arreglo de 5 bytes.

#### Descripción de los métodos:

**Filtro promedio:** Debido a que las mediciones que hace el ADC presentan inestabilidad causada por la velocidad en que se realizan, es necesario mejorar la precisión de estas. El filtro promedio consiste en utilizar un registro de 10 posiciones, este se llena con el corrimiento en las posiciones, por lo que al ingresar un nuevo valor se elimina el valor más antiguo. Adicionalmente se garantiza la estabilidad de la medición comprobando que las medidas tengan un comportamiento parecido, es decir, se realizan mediciones valorando si una medición es menor que la última guardada en el registro, entonces las siguientes 2 mediciones deben

ser menores, al cumplirse esto se agrega el valor a la nueva posición; lo mismo en el caso que sean iguales o mayores. En la figura A.3 del anexo A se muestra el diagrama de flujo que describe el algoritmo del filtro promedio.

**Cálculo de promedio:** A partir del registro utilizado en el filtro promedio, se calcula el promedio de sus 10 valores. Posteriormente se escogen los valores que se encuentren en un rango comprendido entre 10 unidades por encima y 10 por debajo del valor del promedio, lo que en unidades de voltaje representa 24,4379mV por encima y 24,4379mV por debajo del valor del promedio en mV (valor encontrado de forma experimental). Finalmente con los valores escogidos se calcula un nuevo promedio, siendo este el promedio que se enviará vía USB, se descartan los decimales ya que son valores despreciables. La figura A.4 del anexo A muestra el diagrama de flujo que describe el algoritmo del promedio.

### 4.4.3. Aplicación cliente

La aplicación cliente hace la peticiones al microcontrolador y asi mismo recepciona la información solicitada por medio de la comunicación USB. La figura 4.15 muestra los principales casos de usos de la aplicación.

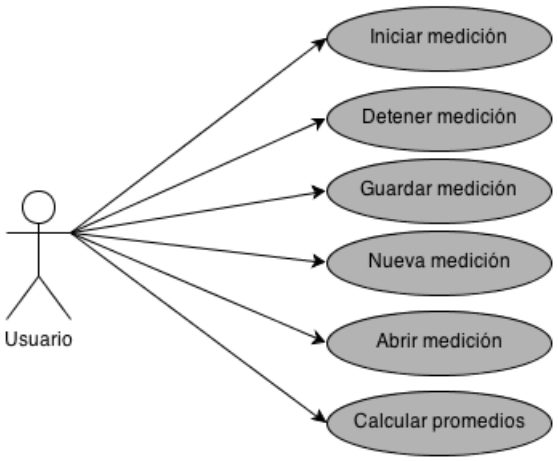


Figura 4.15: Diagrama de casos de uso aplicación cliente.

#### Descripción de los casos de usos esenciales

En la tabla 4.3 se muestra la descripción de los casos de uso esenciales de la aplicación.

#### Diagramas de estados

Los estados principales de la aplicación son en espera, en medición , medición detenida, guardando archivo y medición deshabilitada, estos se muestran el diagrama de la figura 4.16.

#### Descripción de estados

1. **En espera:** El sistema al iniciar la aplicación espera que el usuario inicie una medición, abra un archivo o calcular promedios.

2. **En medición:** El sistema realiza mediciones continuas hasta que se le de la orden de detener la medición.

| | |
|---|---|
| **Caso de uso no. 1** | Iniciar Medición |
| **Iniciador** | Usuario. |
| **Propósito** | Comenzar la obtención de mediciones. |
| **Pre condiciones** | El sistema no debe haber empezado una medición previa. |
| **Resumen** | Al iniciar este caso de uso la aplicación empieza a hacer solicitudes periódicas de medidas al microcontrolador, graficándolas y desplegándolas en una tabla. |
| **Caso de uso no. 2** | Detener Medición |
| **Iniciador** | Usuario. |
| **Propósito** | Parar la adquisición de mediciones. |
| **Pre condiciones** | Se debe haber iniciado una medición. |
| **Resumen** | Este caso de uso detiene las peticiones de medidas. |
| **Caso de uso no. 3** | Guardar Medición. |
| **Iniciador** | Usuario. |
| **Propósito** | Guardar las mediciones obtenidas hasta ese momento |
| **Pre condiciones** | Se debe haber detenido la medición. |
| **Resumen** | Este caso de uso permite guardar la información, solicita al usuario ingresar el nombre con el que se guardarán las mediciones. Al guardar se crea una carpeta con el nombre digitado y en su interior un archivo .csv (hoja de excel) con las mediciones y dos archivos .png con las imágenes de las graficas de corriente y voltaje.En el caso que no se haya detenido la medición al iniciar a este caso de uso, el sistema mostrará una alerta indicando si se desea detener la medición actual. |
| **Caso de uso no. 4** | Nueva medición |
| **Iniciador** | Usuario |
| **Propósito** | Para realizar una nueva medición. |
| **Pre condiciones** | Se debe haber detenido la medición. |
| **Resumen** | Este caso de uso reinicia el programa permitiendo realizar una nueva medición o abrir una medición. |
| **Caso de uso no. 5** | Abrir medición |
| **Iniciador** | Usuario. |
| **Propósito** | Abrir mediciones realizadas anteriormente |
| **Pre condiciones** | El programa debe estar recién iniciado. |
| **Resumen** | Este caso de uso permite abrir archivos .csv de Excel obteniendo la tabla con los valores tiempo, corriente, voltaje y potencia, y permitiendo la visualizar de las graficas y el cálculo de promedios. |
| **Caso de uso no. 6** | Calcular promedios |
| **Iniciador** | Usuario. |
| **Propósito** | Calcular promedios |
| **Pre condiciones** | Se debe haber realizado o abierto una medición. |
| **Resumen** | Este caso de uso permite calcular los promedios de voltaje corriente y potencia en un intervalo de tiempo que ingresa el usuario. |

Tabla 4.3: Descripción casos de uso aplicación cliente.

3. **Medición detenida:** El sistema se detiene por la orden de detener medición permitiendo guardar la medición o eliminarla reiniciando la aplicación con la orden nueva medición.

4. **Guardando archivo:** El sistema se paraliza mientras se guardan los archivos en formato .csv.

5. **Medición deshabilitada:** El sistema deshabilita las mediciones cuando la herramienta hardware esta desconectada.

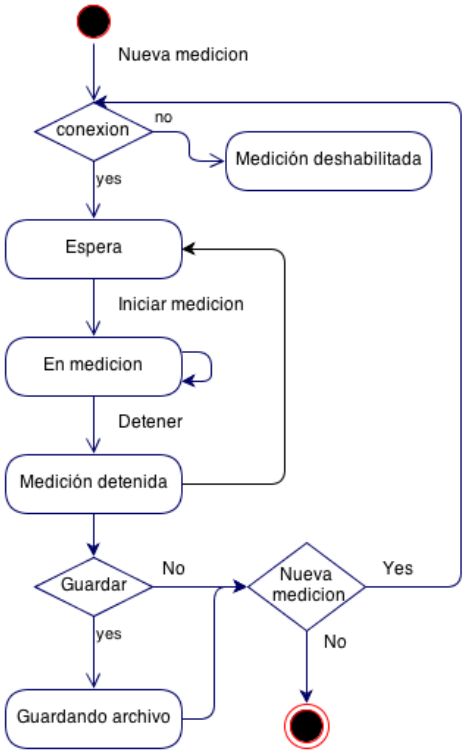


Figura 4.16: Diagrama de casos de uso aplicación cliente.

### Diagrama de paquetes de Análisis esenciales

La aplicación cliente consta de tres paquetes: *Modelo, Medición_vista y Conexión*.En la figura 4.17 se pueden observar los paquetes y su relación.

**Medición_vista:** Este paquete contiene las interfaces gráficas, y la clase *Control_medicion*, la cual implementa la lógica de la aplicación e instancia todas las clases.

**Modelo:** Este paquete contiene las clases auxiliares que permiten graficar las mediciones y manejar los datos en forma de tabla, para posteriormente ser guardados.

**Conexión:** Este último paquete contiene la clase que permite realizar la conexión USB entre la aplicación y el microcontrolador, permitiendo el envio y recepcion de datos. Además, procesa los mensajes con las mediciones, entregándolas al paquete de *Medición_vista*, para que sean mostradas al usuario.

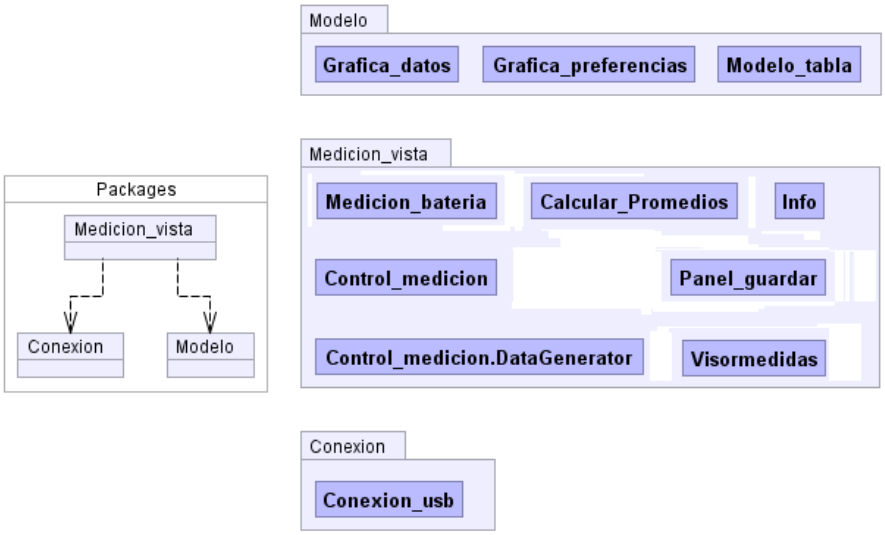


Figura 4.17: Diagrama de paquetes aplicación cliente.

### Interfaces gráficas aplicación

La aplicación cuenta con cinco interfaces gráficas contenidas en el paquete *Medicion_vista* las cuales se describen en la tabla 4.4. Las interfaces principales se pueden apreciar en las figuras 4.18, 4.19, 4.20 y las demás en las figuras A.5 y A.6 del anexo A.

| **Interfaz** | Medicion_bateria |
|---|---|
| **Responsabilidades** | Interfaz principal de la aplicación que detecta los eventos generados por el usuario. |
| **Interfaz** | Panel_guardar |
| **Responsabilidades** | Se encarga de obtiener el nombre de la carpeta y archivos a guardar. |
| **Interfaz** | Calcular_promedios |
| **Responsabilidades** | Con esta interface se realiza el calculo de promedios permitiendo ingresar un intervalo de tiempo y visualizar los resultados. |
| **Interfaz** | Visormedidas |
| **Responsabilidades** | Despliega las medidas de corriente y voltaje que recibe la aplicación. |
| **Interfaz** | Info |
| **Responsabilidades** | Muestra la información acerca de la aplicación. |

Tabla 4.4: Descripción de las interfaces de la aplicación cliente.

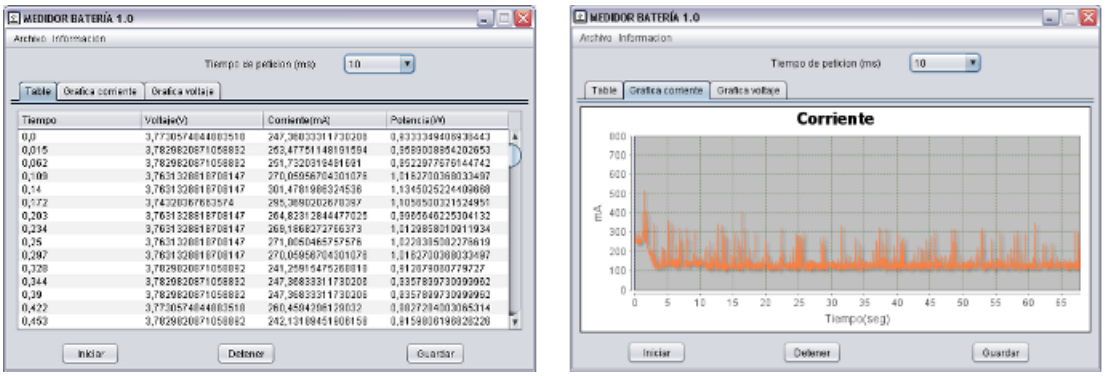


Figura 4.18: Interfaz gráfica *Medicion_bateria*.

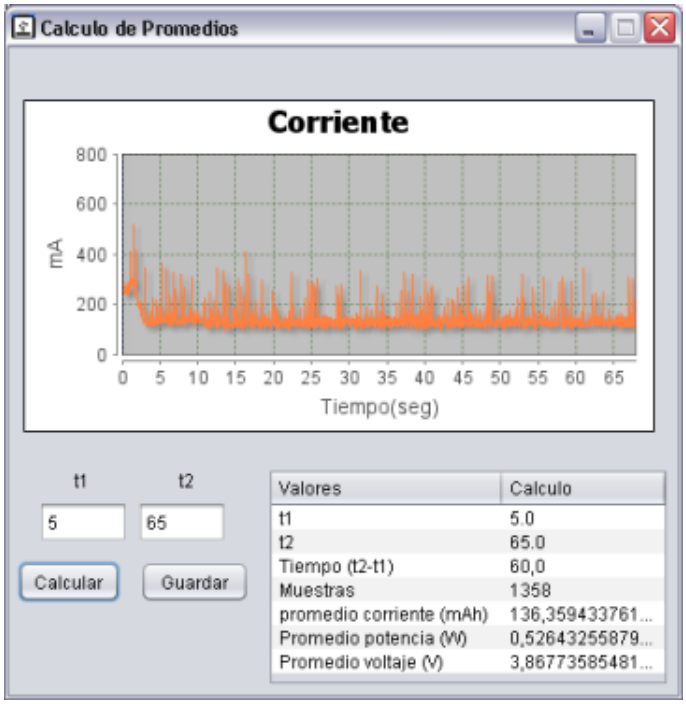


Figura 4.19: Interfaz gráfica *Calcular_promedios*.


Figura 4.20: Interfaz gráfica *VisorMedidas*.

### Diagramas de clases

A continuación se muestran las relaciones entre las clases contenidas en los paquetes de la aplicación, siendo Control_medicion la clase principal de la aplicación la cual contiene toda la lógica de la aplicación, haciendo una instanciación de Conexión_usb, Medicion_bateria, Control_medicion.DataGenerator, Calcular_Promedios, Visormedidas, Panel_guardar, Info, Grafica_datos y Modelo_tabla.La figura A.8 muestra la clase Control_medicion.

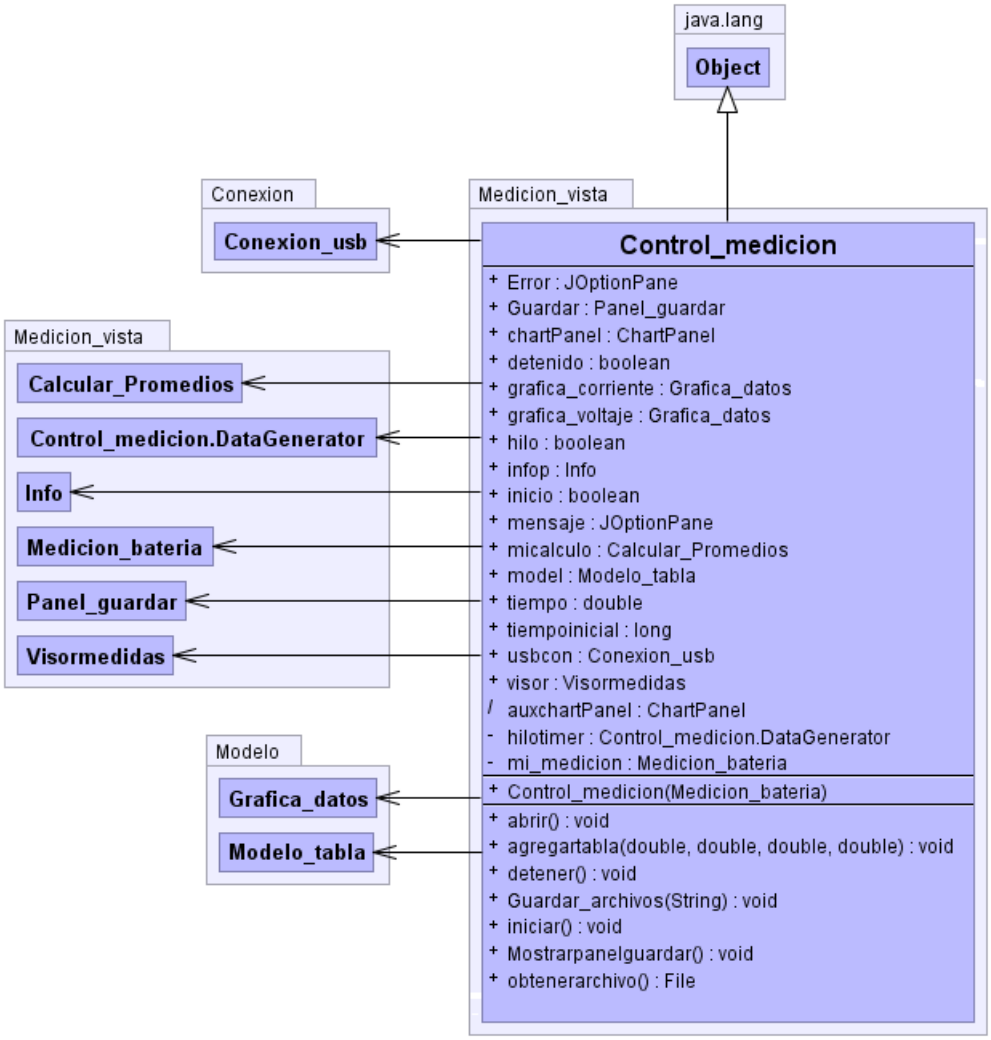


Figura 4.21: Diagrama de clases *Control_medición*.

**Control_medicion:** A continuación se describen las clases contenidas en los paquetes de la aplicación en la tabla 4.5, los gráficos se pueden observar en el numeral A.3.2 del anexo A.

### Diagramas de secuencia

Evaluando los casos de usos anteriores, se tienen los siguientes diagramas de secuencia, los cuales describen las actividades que se generan entre los objetos de la aplicación, a través de mensajes.

| Clase | Control_medicion |
|---|---|
| Responsabilidades | Clase principal que contiene la lógica de la aplicación. |
| Operaciones | |
| agregartabla | Agrega valores a la tabla de la interfaz. |
| iniciar | Inicia un hilo de tipo DataGenerator, con el cual se realiza la obtención periódica de valores del dispositivo hardware, a través de peticiones vía USB. |
| deterner | Detiene el hilo, deteniendo el envío de peticiones y la recepción de valores. |
| Mostrarpanelguardar | Activa la interfaz Panel_guardar. |
| Guardar_archivos | Ejecuta los metodos GuardarExcel y Guarda_imagen |

| Clase | Medicion_bateria.DataGenerator |
|---|---|
| Responsabilidades | Es un generador de datos implementado en la clase Control_medicion, funciona como un hilo que repite el evento ActionEvent |
| Operaciones | |
| actionPerformed | Detecta el evento ActionEvent. Este realiza las peticiones al hardware y recibe las medidas, actualizando los datos de las gráficas y la tabla. |
| Clase | Grafica_datos |
| Responsabilidades | Permite crear gráficas xy con todas sus características, y agregar datos de 1 o 2 funciones.Esta clase utiliza la librería jfreechart. |
| Operaciones | |
| addseries1 | Agrega los datos a la gráfica de la funcion 1. |
| addseries1 | Agrega los datos a la gráfica de la funcion 2. |
| Guarda_imagen | Guarda la imagen de la gráfica como un archivo de extensión .png. |
| Clase | Modelo_tabla |
| Responsabilidades | Hereda de DefaultTableModel sus características, permitiendole manejar los datos de la tabla creada en la interfaz, y así guardarlos. |
| Operaciones | |
| GuardarExcel | Guarda los valores de la tabla en un archivo de extension .csv. |
| abrir_cvs(File archivo) | Procesa un archivo .csv obteniendo los valores de la tabla y desplegandolos en la interface principal. |
| Clase | Conexion_usb |
| Responsabilidades | Se encarga de comunicar la aplicación via USB con el PIC18F4550, utilizando la librería jPicUsb. |
| Operaciones | |
| enviar | Envia 2 bytes al microcontrolador. |
| recibe | Recibe 5 bytes del microcontrolador. |
| Conversion_voltaje | Convierte valores binarios, recibidos del microcontrolador, a valores decimales de |

Tabla 4.5: Descripción de las clases esenciales de la aplicación cliente

**Diagrama de secuencia Iniciar:** El primer diagrama de secuencia es iniciar, el cual es iniciado por el usuario a través de la interfaz mi_medicion al activar el evento en el botón iniciar. A partir de esto, los objetos *control*, *usbcon* y el PIC18f4550 establecen un flujo de mensajes con los cuales se inicializa una medición periódica por medio del hilo DataGenerator, enviando solicitudes y recibiendo las mediciones para finalmente desplegarlas en la interfaz a través de los objetos model, grafica_corriente y grafica_voltaje. Este diagrama se encuentra en la figura A.15 del anexo A.

**Diagrama de secuencia Detener:** El usuario a través de la interfaz mi_medicion activa el evento del botón detener, sí la aplicación esta midiendo se envía el mensaje detener al objeto control, el cual detiene el hilo y así las solicitudes al microcontrolador. Este diagrama se encuentra en la figura A.14 del anexo A.

**Diagrama de secuencia Guardar:** En este diagrama el usuario a través de la interfaz mi_medicion activa el evento del botón guardar o el menú guardar. El objeto mi_medicion envía el mensaje Guardar al objeto control, el cual verifica el estado del hilo, sí el hilo esta activado envía un mensaje de alerta al usuario con la opción de detener el hilo, ya que no es posible guardar sin detener previamente la medición; si el hilo esta desactivado se despliega la interfaz Guardar, en la cual el usuario ingresa el nombre de la carpeta que se creará para guardar los archivos, posteriormente si el usuario presiona el botón Guardar, se dará la orden de guardar a los objetos model, grafica_corriente y grafica_voltaje. Este diagrama se encuentra en la figura A.16 del anexo A.

## 4.5. Calibración de la herramienta para la medición del consumo de energía

La figura 4.22 muestra los parámetros que corresponden a las medidas y valores que deben ser verificados en la herramienta hardware y en la aplicación cliente para una correcta calibración.

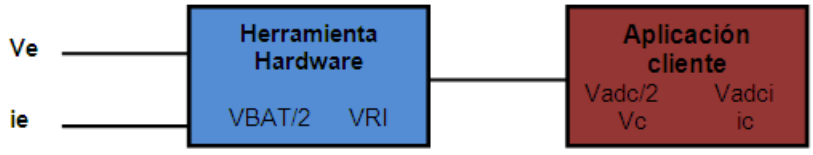


Figura 4.22: Medidas y valores a calibrar en la herramienta.

**Medidas a calibrar:**

- ***Ve:*** voltaje de la batería del dispositivo
- ***ie:*** corriente que entrega la batería al dispositivo
- ***VBAT/2:*** corresponde al voltaje entregado por el Modulo de medición de voltaje y corriente, donde *VBAT/2 = Ve/2* aproximadamente.
- ***VRI:*** corresponde a la medida de corriente expresada en voltaje que es entregada por el Modulo de medición de voltaje y corriente, donde *VRI = ie * Rm*.
- ***Vadc/2:*** es la medida del *VBAT/2* entregada al Módulo Aplicación Cliente.
- ***Vadci:*** es la medida de *VRI* entregada al Módulo Aplicación Cliente.
- ***Vc:*** es el voltaje calculado en la Aplicación Cliente correspondiente a Ve.

- ***ic:*** es la corriente calculado en la Aplicación Cliente correspondiente a *ie*.

### 4.5.1. Calibración de las medidas

Para realizar la calibración de las medidas anteriores se utilizaron tres herramientas que permiten la obtención de datos precisos, estas son:

- Multmetro: BEST BT60A
- Multimetro: Tech TM 135
- Osciloscopio: Rigol DS1102E

Adicionalmente, para la medición de los voltajes se utilizan resistencias fijas remplazando el dispositivo móvil en la conexión de la tarjeta PCB (la conexión del dispositivo se puede ver en el anexo B en la sección B.1).

### 4.5.2. Calibración del voltaje *Vc*

Utilizando las herramientas anteriormente nombradas se calibro el voltaje de la siguiente manera.

1. Se mide el voltaje en la entrada *Ve* y se obtiene el voltaje *Vadc/2* en la aplicación cliente, variando el voltaje *Ve* con una fuente variable.
2. Se multiplica por 2 el voltaje *Vadc/2* para calcular el voltaje *Vc*.
3. Se calcula el coeficiente de error con la ecuación 4.9 , el cual es provocado por la tolerancia de las resistencias del divisor de voltaje.
$$Ce = \frac{Ve - Vc}{Ve} \tag{4.9}$$
4. Se calcula el promedio del coeficiente de error y se realiza la corrección en la aplicación cliente ingresando la ecuación 4.10.
$$VC = (Vadc * 2) + (Vadc * 2 * Ce) \tag{4.10}$$

En la tabla 4.6 se muestran los valores medidos en los pasos anteriores.

**Observación:** En el momento que se tomaron los valores con la herramienta se observó que estos permanecían constantes o en su defecto presentaban variaciones muy pequeñas, es por esto que se concluye que la herramienta tiene una estabilidad muy buena.

### 4.5.3. Calibración de corriente *ic*

Utilizando las herramientas anteriormente nombradas se calibro corriente de la siguiente manera.

1. Se obtiene el voltaje *Vadci* y la corriente *Ie*, colocando una resistencia en lugar del dispositivo y variándola con el fin de obtener diferentes valores de corriente. El voltaje Ve permanece constante.
2. Se calcula la corriente *ic* a partir del voltaje *Vadci* con la ecuación 4.11.
$$ic = \frac{VRI}{Rm * Av} \tag{4.11}$$
donde *VRI* = *Vadci*, *Rm*=0.0275ohm y *Av*=225.

| Fuente de voltaje (V) | Voltaje Multímetro Ve (V) | Vadc/2 (V) | Vc (V) | Coeficiente de Error Ce |
|---|---|---|---|---|
| 4,2 | 4,190 | 2,082653079 | 4,165306159 | 0,005893518 |
| 4,15 | 4,150 | 2,06279938 | 4,12559876 | 0,005879817 |
| 4,1 | 4,100 | 2,038478598 | 4,076957196 | 0,005620196 |
| 4,05 | 4,050 | 2,014157816 | 4,028315632 | 0,005354165 |
| 4 | 4,000 | 1,984873609 | 3,969747219 | 0,007563195 |
| 3,95 | 3,950 | 1,960552827 | 3,921105655 | 0,007315024 |
| 3,9 | 3,900 | 1,935735703 | 3,871471406 | 0,007315024 |
| 3,85 | 3,850 | 1,911414921 | 3,822829842 | 0,007057184 |
| 3,8 | 3,800 | 1,887094139 | 3,774188278 | 0,006792558 |
| 375 | 3,750 | 1,857809932 | 3,715619865 | 0,009168036 |
| 37 | 3,700 | 1,83348915 | 3,666978301 | 0,008924784 |
| 3,65 | 3,658 | 1,813635451 | 3,627270902 | 0,008400519 |
| 3,6 | 3,599 | 1,789314669 | 3,578629338 | 0,005660089 |
| 3,55 | 3,551 | 1,760030462 | 3,520060924 | 0,008712778 |
| 3,5 | 3,500 | 1,737198708 | 3,474397416 | 0,007315024 |
| | | | Coeficiente de Error promedio | 0,007131461 |

Tabla 4.6: Calibración voltaje *Vc*.

3. Se corrigen los errores en los datos encontrados.

En la tabla 4.7 se muestran los valores medidos en los pasos anteriores.

**Observación:** Dado los valores de la tabla 4.7 se observa una diferencia entre los valores de ic e ie la cual se atribuye a utilizar valores teóricos por tanto se procede a realizar una curva de calibración utilizando los valores de ie y Vadci, que son valores reales. La figura 4.23 muestra la curva de calibración.

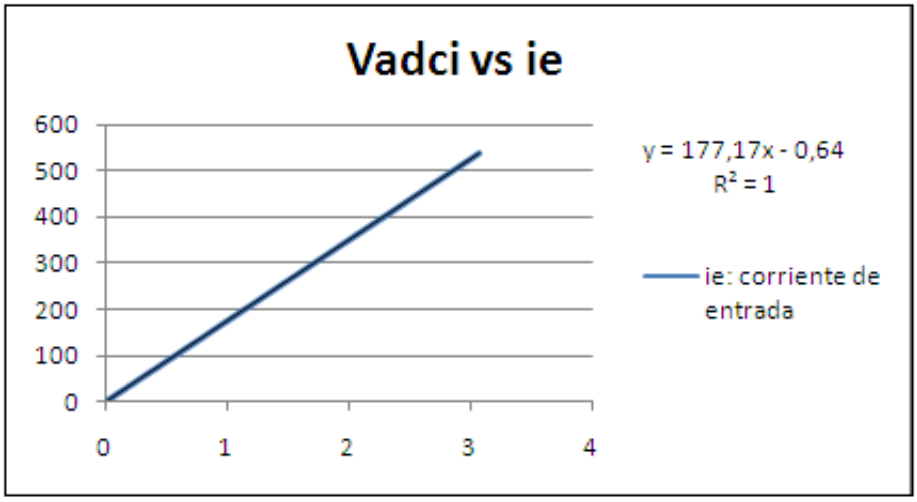


Figura 4.23: Curva Vadci vs ie.

A partir de la curva de la figura 4.23 se obtiene una aproximación lineal, la cual genera un valor cuadrado $R^2$ igual a 1, que significa que se ha encontrado la mayor precisión de la ecuación lineal. La ecuación lineal resultante es de la forma Y = aX + b, donde X es el voltaje Vadci y Y es la corriente ic en mA. La ecuación

| ie (mA) | Vadci (V) | ic (mA) |
|---|---|---|
| 1,6 | 0,007419355 | 1,199087644 |
| 1,96 | 0,009892473 | 1,598783531 |
| 2,6 | 0,01483871 | 2,398175305 |
| 3,2 | 0,017311828 | 2,797871192 |
| 4,92 | 0,027204301 | 4,396654723 |
| 10,11 | 0,05688172 | 9,193005317 |
| 14,3 | 0,081612903 | 13,18996415 |
| 15,48 | 0,08655914 | 13,98935593 |
| 20,5 | 0,116236559 | 18,78570652 |
| 25,9 | 0,148387097 | 23,98175302 |
| 30,8 | 0,175591398 | 28,37840773 |
| 34,7 | 0,197849462 | 31,97567069 |
| 39,8 | 0,227526882 | 36,77202128 |
| 44,2 | 0,252258065 | 40,76898012 |
| 49,8 | 0,284408602 | 45,96502662 |
| 55 | 0,314086022 | 50,76137721 |
| 58,9 | 0,336344086 | 54,35864016 |
| 64,2 | 0,366021505 | 59,15499077 |
| 68,4 | 0,390752688 | 63,15194961 |
| 83,6 | 0,477311828 | 77,14130554 |
| 99,6 | 0,568817204 | 91,93005322 |
| 110,9 | 0,63311828 | 102,3221462 |
| 117,7 | 0,672688172 | 108,7172803 |
| 135,6 | 0,77655914 | 125,5045074 |
| 145,3 | 0,830967742 | 134,2978169 |
| 310 | 1,770861528 | 286,199843 |
| 540 | 3,082833681 | 498,2357465 |

Tabla 4.7: Calibración corriente $ic$.

4.12 es la forma aproximada que nos entrega Excel.

$$ic = 177,17 * Vadci - 0,64 \tag{4.12}$$

### 4.5.4. Características finales de la herramienta

A continuación se en listan las características finales de la herramienta que se encontraron por medio de la "Guía - Prueba de Comprobación de la Herramienta para la Medición del Consumo de Energía" ubicada en el anexo C.

- **Precisión:** variación de 4,88mV entre medidas.
- **Frecuencia de medición:** 66,667Hz.
- **Graficación en tiempo real:** gráficas de voltaje y corriente.
- **Persistencia de la información:** archivos .csv y .jpg.

# Capítulo 5

# Planificación de pruebas

Este capítulo aborda la planificación de pruebas sobre los periféricos GPS, sistema de audio (auricular y altavoz), acelerómetro, brújula y teclado en busca de posibles prácticas de programación sobre la gestión de energía aplicando el Proceso para la planificación de pruebas (P.P.P.) del Modelo de evaluación de energía para dispositivos Móviles. La planificación de pruebas consta de dos procesos, la caracterización del consumo inicial y el diseño de pruebas.

## 5.1. Dispositivos móviles utilizados

Para realizar la planificación se hace uso de los dispositivos móviles Samsung Nexus S y Samsung galaxy ACE, sus características de describen en la tabla 5.1. Inicialmente se utiliza el Samsung Nexus S por tener

| Características | Samsung Nexus S | Samsung Galaxy ACE |
|---|---|---|
| Sistema operativo | Android 4.1.2 | Android 2.3.6 |
| Procesador | ARMv7 1000Mhz | ARMv7 832Mhz |
| Tamaño pantalla | 4" 480x800 | 3,5" 320x480 |
| Acelerómetro | KR3DM 3-axis, de STMicroelectronics | BMA222, de Bosch Sensortec |
| Magnetómetro | AK8973 3-axis, de Asahi Kasei Microdevices. | MS-3E (YAS530), de Yamaha Corporation |
| Sensor de Orientación | Por Google Inc. | MS-X, deYamaha Corporation |

Tabla 5.1: Caracteristicas Nexus S y Samsung galaxy ACE.(Adaptado de [35]).

un sistema operativo más reciente de android, el Samsung Galaxy ACE se utilizó como segunda opción.

## 5.2. Caracterización del consumo inicial

Para la caracterización del consumo inicial se utilizan los método de sincronización de eventos denominado "Picos de Consumo" de la sección 3.1.2, con el fin de detectar los puntos entre los cuales se realizará la medición y así facilitar el análisis de los datos. Los picos son implementados en la clase Marcas.java, la cual se describe en el anexo D. Como medidas principales para el análisis del consumo se utilizan el promedio de corriente, tiempo de ejecución de las pruebas y el tiempo de duración de la batería. A continuación se describe la caracterización del consumo inicial sobre los periféricos especificados.

### 5.2.1. GPS

**Selección de algoritmo:**

Para la utilización de este periférico se utiliza el paquete *android.location* el cual contiene 4 clases principales (*LocationManager, Location, Provider, Location, Criteria*) y una interfaz (*LocationListener*), las cuales permiten encontrar la localización del dispositivo accesando a diferentes proveedores los cuales son *GPSlocation, Network location, Passivelocation*. El algoritmo utilizado con los métodos anteriormente nombrados se puede observar en la figura 5.1. Al utilizar un proveedor fijo como en el caso del GPS se hace innecesario

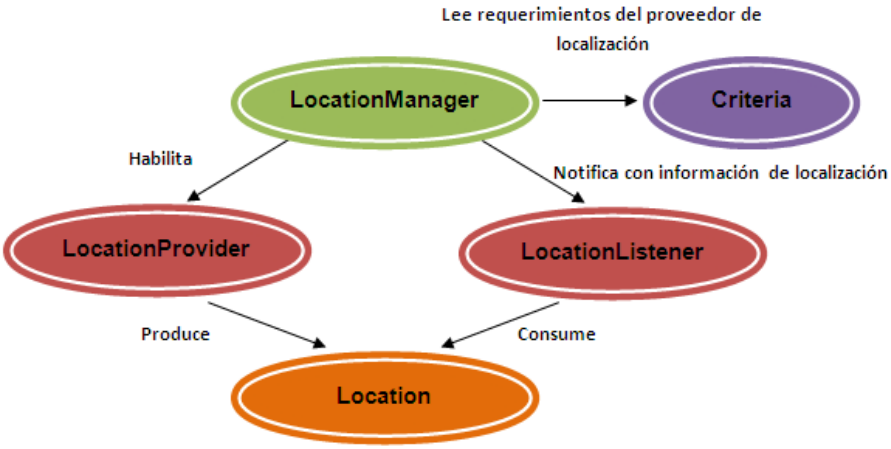


Figura 5.1: Algoritmo *android.location* con proveedor variable. (Adaptado de [36]).

utilizar las clases *LocationProvider* y *Criteria* por tanto el algoritmo a utilizar se simplifica como se muestra en la figura 5.2. En la tabla 5.2 se especifican las funciones de las clases y los métodos a utilizar.

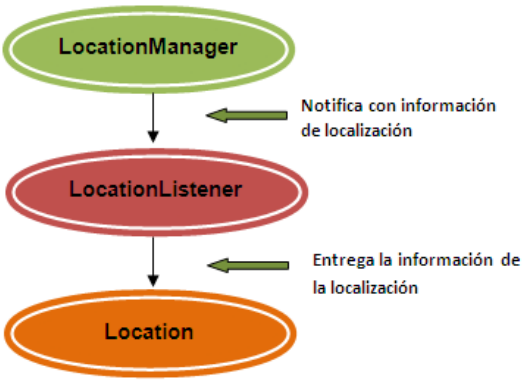


Figura 5.2: Algoritmo *android.location* con proveedor GPS.

**Descripción del algoritmo** El algoritmo para la utilización del GPS consta de los siguientes pasos:

1. Obtener referencia al *LocationManager*.

```
LocationManager lM= (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
```

| | Descripción | Métodos |
|---|---|---|
| **LocationManager** | Esta clase proporciona acceso a los servicios del sistema de localización. | **getLastKnownLocation(String provider):** Devuelve una ubicación que indica los datos de la última ubicación conocida obtenida del proveedor dado. |
| | | **requestLocationUpdates(String provider, long minTime, float minDistance, LocationListener listener):** Registra las actualizaciones de ubicación utilizando el nombre del proveedor y el listener. |
| | | **removeUpdates(LocationListener listener);**Elimina todas las actualizaciones de ubicación |
| **LocationListener** | Se utiliza para recibir notificaciones del LocationManager. | **onLocationChanged(Location location):** Se llama cuando la ubicación ha cambiado. |
| | | **onProviderDisabled(String provider):** Se llama cuando el proveedor es desactivado por el usuario. |
| | | **onProviderEnabled(String provider):** Se llama cuando el proveedor es habilitado por el usuario. |
| | | **onStatusChanged(String provider, int status, Bundle extras):** Se llama cuando el estado de un proveedor cambia. |
| **Location** | Una clase que representa los datos de una ubicación geográfica. | **getAccuracy():** Obtiene la precisión estimada de esta ubicación, en metros. |
| | | **getLatitude():** Obtiene la latitud en grados. |
| | | **getLongitude():** Obtener la longitud en grados. |
| | | **getProvider():** Retorna el nombre del proveedor que ha generado la revisión. |
| | | **getSpeed():** Obtener la velocidad si está disponible, en metros / segundo sobre la tierra |

Tabla 5.2: Clases y métodos *android.location* utilizados. (Adaptado de [23]).

2. Obtener la última posición conocida del proveedor GPS. Location loc = lM.getLastKnownLocation(LocationManager.GPS_PROVIDER);

3. Obtener valores de localización.

```
loc.getLatitude();    // obtiene la latitud
loc.getLongitude();   // obtiene la longitud
loc.getAccuracy();    // obtiene la precisión
loc.getSpeed();       // obtiene la velocidad
loc.getProvider();    // obtiene el proveedor
```

4. Registrar las actualizaciones del localizaciónListener teniendo en cuenta el proveedor, el tiempo en *ms*, distancia en *mt* y el *LocationListener*.

```
locManager.requestLocationUpdates(
LocationManager.GPS\_PROVIDER, 1000,0, locListener);
```

5. Mostrar la información mediante los eventos escuchados con el listener(los siguientes métodos son propios de *locationListener*).

```
public void onLocationChanged(Location location){
```

```
        mostrarPosicion(location);
    }

    public void onProviderDisabled(String provider) {
        lblEstado.setText("Proveedor Desactivado");
    }

    public void onProviderEnabled(String provider) {
        lblEstado.setText("Proveedor Activado");
    }

    public void onStatusChanged(String provider,
    int status, Bundle extras) {
        lblEstado.setText(status);
    }
```

6. Cancelar las actualizaciones de localización(este paso se realiza para detener la aplicación).

```
locManager.removeUpdates(locListener);
```

### Selección de escenario inicial

Para la medición del consumo del GPS en el dispositivo móvil presenta el siguiente escenario:

- Modo avión.
- GPS activado.
- Pantalla con brillo mínimo.
- WIFI, auto sincronización y Rotación de pantalla desactivados.
- Aplicaciones en segundo plano desactivadas.
- La aplicación se correrá hasta obtener 10*mt* de precisión.

### Implementación inicial

Se implementa una aplicación con el algoritmo anteriormente descrito y el proveedor GPS. La aplicación se llama “Localización” la cual se puede observar en la figura 5.3. Esta aplicación consta de tres botones. El botón “Activar” inicia la localización del dispositivo, el botón “Desactivar” permite detener la localización y el botón “Activar Servicio GPS” despliega el menú donde se pueden activar los proveedores de localización.

### Medición del consumo inicial

Para la medición del consumo se agrega un pico de consumo de bajada al activar la localización y un pico de consumo de subida al seleccionar la opción desactivar, permitiendo ubicar el activación y el desactivación del servicio de localización a través de las gráficas desplegadas en la herramienta para la medición de consumo energía.

1. Como primer paso se mide el consumo de la interfaz de la aplicación bajo el escenario descrito. Se realiza la prueba 5 veces obteniendo un valor de corriente promedio de 100,0748737mA y una potencia promedio de 397,946215mW. La figura 5.4 muestra la gráfica de corriente en una de las 5 repeticiones.


Figura 5.3: Aplicación Localización.

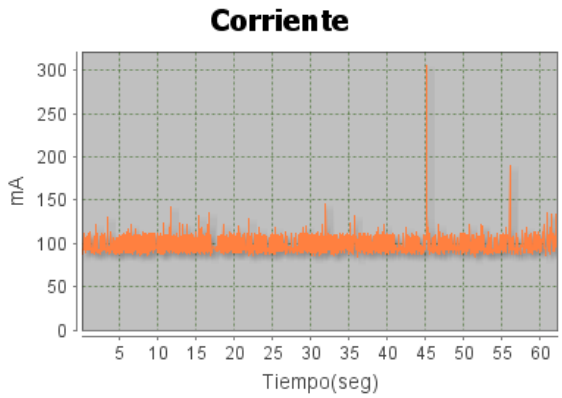


Figura 5.4: Medición interfaz aplicación “localización”.

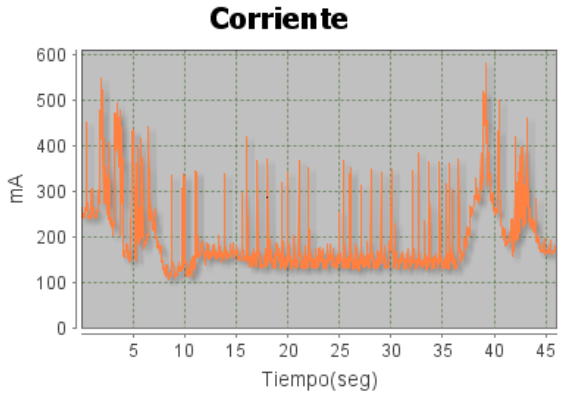


Figura 5.5: Medición GPS en exteriores.

2. Se mide el consumo del GPS activando la aplicación Localización en exteriores, sin obstáculos a la señal del satélite, e interiores, en la segunda planta de una casa, realizando 5 repeticiones para cada caso. La figura 5.5 muestra el consumo en exteriores en donde se obtuvo un promedio de 149,8615346mA

y 586,964736mW. La figura 5.6 muestra el consumo en interiores en donde se obtuvo un promedio de 155,8949518mA y 618,037204mW.

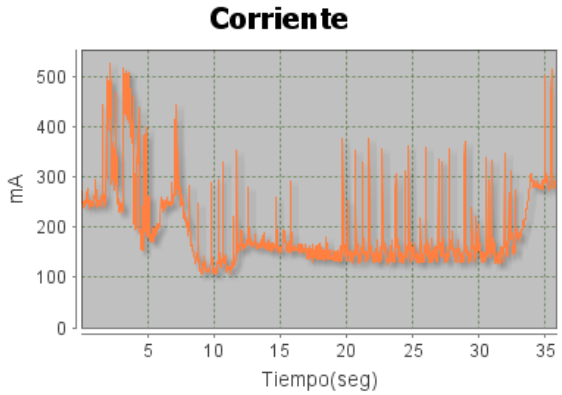


Figura 5.6: Medición GPS al interior de una casa.

3. Se calcula el consumo del GPS restando el consumo de la interfaz, obteniendo como promedio los resultados de la tabla 5.3.

| Ubicación | Promedio | | | |
|---|---|---|---|---|
| | Corriente (mAh) | Potencia (mW) | Corriente sin interfaz (mAh) | Potencia sin interfaz (mW) |
| Interiores | 155,895 | 618,037 | 55,820 | 220,091 |
| Exteriores | 149,862 | 586,965 | 49,787 | 189,019 |

Tabla 5.3: Promedio de consumo caracterización inicial GPS.

En la figura 5.7 se muestra el gráfico de barras correspondiente al promedio de cada prueba, incluyendo el promedio de la interfaz.

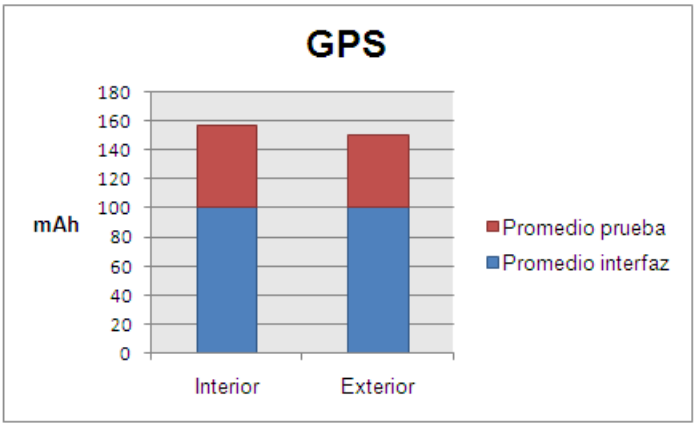


Figura 5.7: Promedios de corriente, caracterización inicial GPS.

### Análisis del consumo inicial

En la implementación del código se especifica que las actualizaciones se realizan cada 1000ms, pero al observar la aplicación se nota que las actualizaciones no presentan este periodo. Esto puede ser causado debido a que el GPS es un sistema que depende de variables externas al dispositivo, por lo cual las actualizaciones pueden generarse en tiempos diferentes al establecido. Como puede esperarse el consumo del GPS en exteriores presenta un nivel menor al del GPS en interiores. El análisis del consumo se realiza sobre el GPS en exteriores ya que los dos casos son similares. En la figura 5.8 se marcan los aspectos a analizar, la cual fue dividida en segmentos de diferentes colores. A continuación se analizan los segmentos:

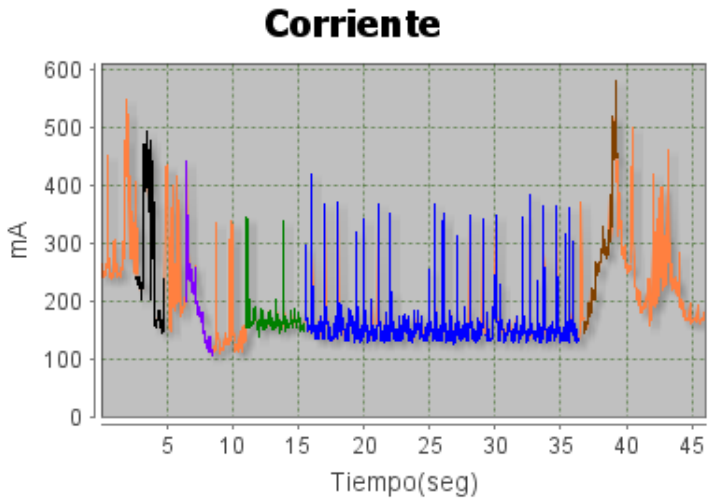


Figura 5.8: Consumo de corriente por segmentos caracterización inicial GPS.

- **Segmento negro:** este segmento se presenta cuando se activa en el dispositivo el GPS.
- **Segmento morado:** este corresponde al pico de consumo de bajada que antecede a la activación del servicio de localización.
- **Segmento verde:** en este segmento el nivel de corriente se eleva por unos cuantos segundos por la inicialización del GPS en donde este establece la conexión con el satélite. Este segmento es mayor en interiores ya que al encontrarse dentro de una edificación el dispositivo se demora más en encontrar y establecer la conexión con el satélite.
- **Segmento azul:** en este segmento se observa un nivel mínimo que se puede considerar constante en los siguientes segundos de la medición hasta desactivar el servicio de localización, además se generan picos de magnitud significativa que pueden ser atribuidos a las actualizaciones de ubicación o el cambio de valores en la interfaz.
- **Segmento café:** este corresponde al pico de consumo de subida que indica que se desactivó el servicio de localización.

**Preguntas:** Dados los resultados analizados anteriormente surgen las siguientes preguntas:

- *¿Los picos de actualizaciones presentes en el segmento azul pueden disminuirse al modificar las variables de método requestLocationUpdates()?*
- *¿Qué tanto afecta el cambio de valores en la interfaz al consumo en la aplicación?*

### 5.2.2. Audio

#### Selección de algoritmo

Para reproducir archivos de sonido se encontraron tres librerías de Android *SoundPool*, *MediaPlayer* y *AudioTrack* las cuales se describe en la tabla 5.4. Dadas las características de las librerías descritas en la tabla, se decide utilizar la librería MediaPlayer ya que es la más utilizada en las aplicaciones de reproducción.

| Características | Soundpool | Mediaplayer | Audiotrack |
|---|---|---|---|
| Tamaño del Archivo | 1mb<= | Cualquiera | Cualquiera |
| Uso | Juegos con sonidos cortos | Música y video | Música |
| Formato | mp3, wav | mp3,wav | 16 bit PCM (wav y mp3 deben convertirse) |
| Latencia | Baja | Alta | Baja |

Tabla 5.4: Tabla comparativa librerías audio.(Adaptado de Android Developers [37][38][39]).

**Algoritmo** El algoritmo utilizado la librería *MediaPlayer* se puede observar en la figura 5.9.

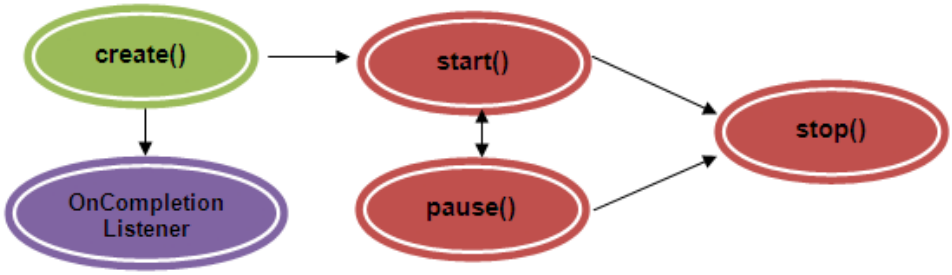


Figura 5.9: Algoritmo para reproducción de audio.
Algoritmo para reproducción de audio.

**Descripción del algoritmo** Los pasos para la reproducción de audio son los siguientes

1. Crear objeto MediaPlayer con la url del archivo (puede ser interno o externo a la aplicación).

```
data = Uri.parse(''file:///sdcard'' +
''/Tesis/Audio/angry.mp3'');
MediaPlayer mp = MediaPlayer.create(
AudioMPMain.this, data);
```

2. Establecer el OnCompletionListener al objeto MediaPlayer.

```
mp.setOnCompletionListener(this);
```

   Este listener permite detectar que el archivo de audio se termino de reproducir con el evento OnCompletion.

3. Definir las opciones de reproducción en la aplicación con los métodos mp.start(), mp.stop(), mp.pause().

### Selección de escenario inicial

Para la medición del consumo del audio el dispositivo móvil se presenta el siguiente escenario:

- Modo avión
- Pantalla con brillo mínimo
- GPS, WIFI, auto sincronización y Rotación de pantalla desactivados.
- Aplicaciones en segundo plano desactivadas.
- Nivel de sonido multimedia 75 %.
- Archivo de audio angry.mp3, se selecciona debido a que es el tono musical del juego más descargado en el 2012 en el Google Play.
- Archivo de audio mp3 estabilizado a *95db*.
- Reproducción con y sin audífonos.

### Implementación inicial

Se implementa una aplicación con el algoritmo que utiliza la clase MediaPlayer, esta se llama “Audiomp” y se puede observar en la figura 5.10. Esta aplicación consta de cinco botones. El botón “Cargar Archivo sd”

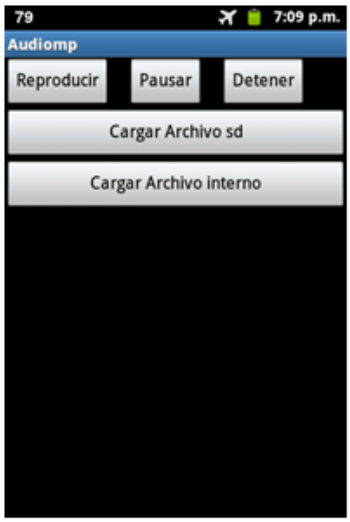


Figura 5.10: Aplicación “Audiomp”.

carga un archivo de audio .mp3 externo a la aplicación, el botón “Cargar Archivo interno” carga un archivo de audio .mp3 interno a la aplicación, el botón “Reproducir” inicia la reproducción del archivo de audio, el botón “Pausar” permite pausar la reproducción y el botón “Detener” finaliza la reproducción.

### Medición del consumo inicial

Para la medición del consumo se agrega una marca de alto consumo al inicio de la aplicación, una de bajo consumo al seleccionar el botón reproducir y otra de alto consumo cuando se detecta que el archivo de audio se termino de reproducir. Con lo anterior se podrá ubicar el inicio y fin de la reproducción.

1. Como primer paso se mide el consumo de la interfaz de la aplicación bajo el escenario descrito. Se realiza la prueba 5 veces obteniendo un valor de corriente promedio de 107,397799mA y una potencia promedio de 438,302mW. La figura 5.11 muestra la gráfica de corriente en una de las 5 repeticiones.

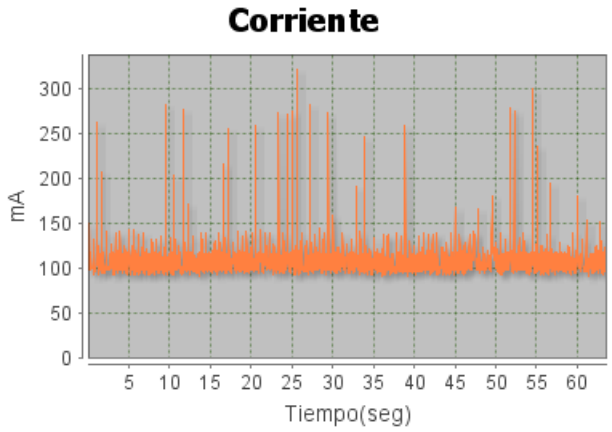


Figura 5.11: Medición interfaz aplicación “Audiomp”.

2. Se mide el consumo del Audio ejecutando la aplicación “Audiomp”, esta aplicación se ejecuta con o sin audífonos y con un archivo de audio interno y un externo. Se realizan 5 repeticiones para cada caso específico.

   *a*) **Sin audífonos:** La figura 5.12 a) muestra el consumo del archivo interno donde se obtuvo un promedio de 125,6246608mA y 508,307856mW. La figura 5.12 b) muestra el consumo del archivo externo donde se obtuvo un promedio de 120,4693303mA y 484,871634mW.

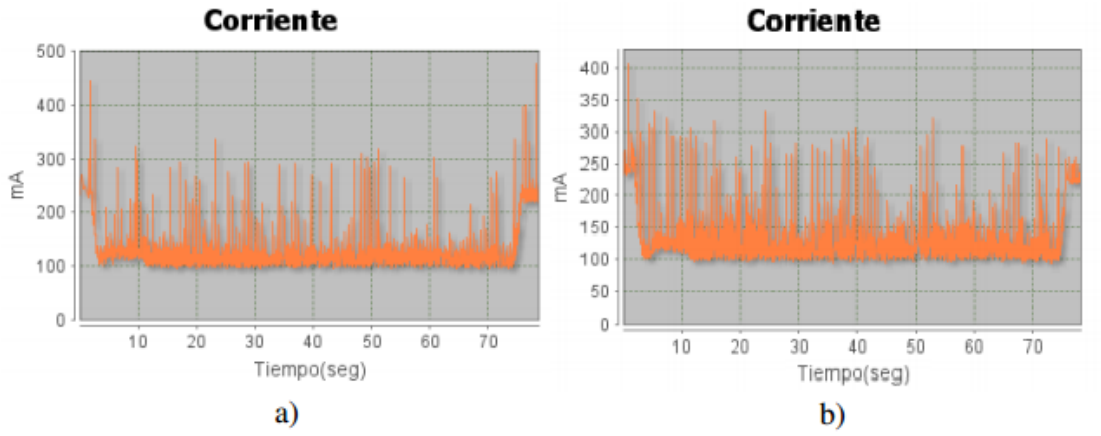


Figura 5.12: Medición audio sin audífonos. a) Archivo interno. b) Archivo externo.

   *b*) **Con audífonos:** La figura 5.13 a) muestra el consumo del archivo interno donde se obtuvo un promedio de 118,0593383mA y 470,587439mW. La figura 5.13 b) muestra el consumo del archivo externo donde se obtuvo un promedio de 118,5957637mA y 472,544444mW.

   *c*) Se calcula el consumo del Audio restando el consumo de la interfaz, obteniendo como promedio los resultados de la tabla 5.5. En la figura 5.14 se muestra el gráfico de barras correspondiente al promedio de cada prueba, incluyendo el promedio de la interfaz.

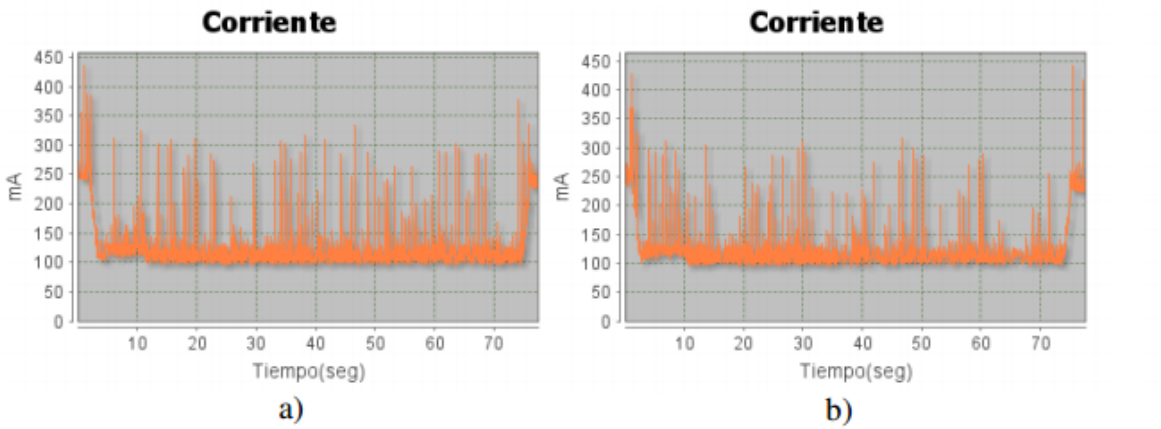


Figura 5.13: Medición audio con audífonos. a) Archivo interno. b) Archivo externo.

| Archivo | Promedio sin audífonos | | | | Promedio con audífonos | | | |
|---|---|---|---|---|---|---|---|---|
| | Corriente (mAh) | Potencia (mW) | Corriente sin interfaz (mAh) | Potencia sin interfaz (mW) | Corriente (mAh) | Potencia (mW) | Corriente sin interfaz (mAh) | Potencia sin interfaz (mW) |
| Interno | 125,625 | 508,308 | 18,227 | 70,006 | 118,059 | 470,587 | 10,662 | 32,285 |
| Externo | 120,469 | 484,872 | 13,072 | 46,569 | 118,596 | 472,544 | 11,198 | 34,242 |

Tabla 5.5: Consumo promedio Audio.

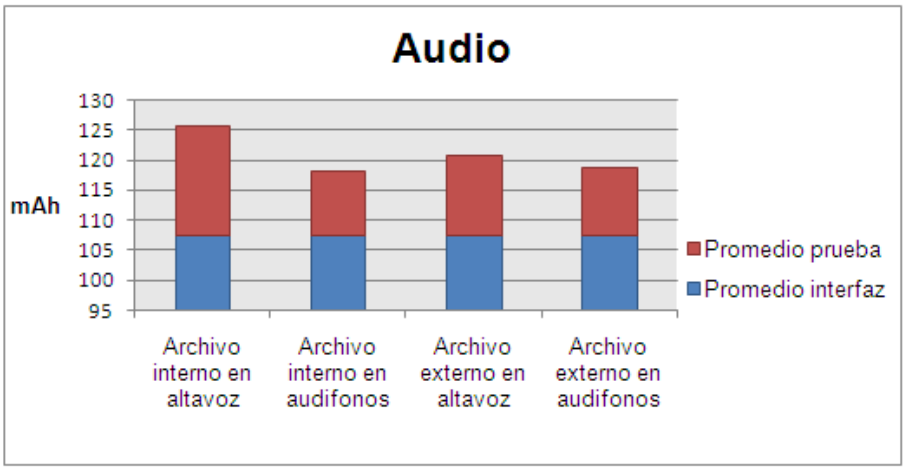


Figura 5.14: Promedios de corriente, caracterización inicial audio.

### Análisis del consumo inicial

En la figura 5.11 de la medición del consumo de la interfaz se puede observar que se presenta picos inesperados que se atribuyen al procesamiento de la aplicación, por esta razón es difícil distinguir los picos generados por la reproducción del archivo, además que el algoritmo de reproducción es muy sencillo y no requiere mayor análisis. Por las anteriores razones se toma el promedio de consumo como referencia para este caso de estudio. Analizando la tabla de promedios se puede concluir lo siguiente:

- Reproducir un archivo de música con audífonos: El consumo de un archivo interno o externo no produce una diferencia significativa en el consumo. Reproducir un archivo con los audífonos

produce menos consumo que sin ellos.

- Reproducir un archivo de música sin audífonos: El consumo de un archivo interno es mayor al consumo de un archivo externo a la aplicación.

**Preguntas:** Dado el escenario presente, el tipo de archivo utilizado y las pocas opciones de modificar el algoritmo surgen las siguientes preguntas:

- *¿Cuál es la diferencia entre los géneros de música más conocidos teniendo canciones normalizadas a un nivel de volumen definido?*
- *Al ser angry.mp3 un archivo de una canción creada digitalmente y además ha sido normalizado su volumen, ¿Cuál sería la diferencia al utilizar una melodía sin normalizar que haya sido grabada en vivo en un estudio discográfico?*
- *Sí se utiliza el control de volumen como practica de programación utilizando diferentes niveles entre audífonos y altavoz, ¿se puede disminuir el consumo?*

### 5.2.3. Acelerómetro

#### Selección de algoritmo

Para la utilización de los sensores del dispositivo existe una clase llamada SensorManager y una interfaz llamada *SensorEventListener*, ubicadas en el paquete *android.hardware*. En la tabla 5.6 se encuentra su descripción .

| | Descripción | Métodos |
|---|---|---|
| **SensorManager** | Permite tener acceso a los datos de cualquiera de los sensores físicos | **getDefaultSensor(inttype):** Obtiene el sensor predeterminado para el tipo especificado. |
| | | **registerListener(SensorEventListener listener, Sensor sensor, intrate):** Registra un SensorEventListener para un sensor en específico. |
| | | **unregisterListener(SensorListener listener):** Anula el registro del listener para todos los sensores. |
| **SensorEventListener** | Recibe notificaciones del sensor manager. | **onSensorChanged(SensorEventevent):** Se le llama cuando los valores del sensor han cambiado. |
| **SensorEvent** | Representa el evento de un Sensor y contiene los datos de este. | No se utiliza ningún método, pero si el variable values, el cual es de tipo Array: **Values:** Contiene un arreglo de los valores del sensor que generó el evento. |
| **SensorListener** | Recibe notificaciones del sensor manager. Obsoleta desde la API de nivel 3. | **onSensorChanged(int sensor, float[] values):** Entrega un entero (int) del sensor que se está utilizando y un arreglo de valores flotantes (float[]) los cuales corresponden a los datos medidos por el sensor. Este evento se genera cuando ocurre un cambio de estado en el sensor. |

Tabla 5.6: Clases del paquete android.hardware.

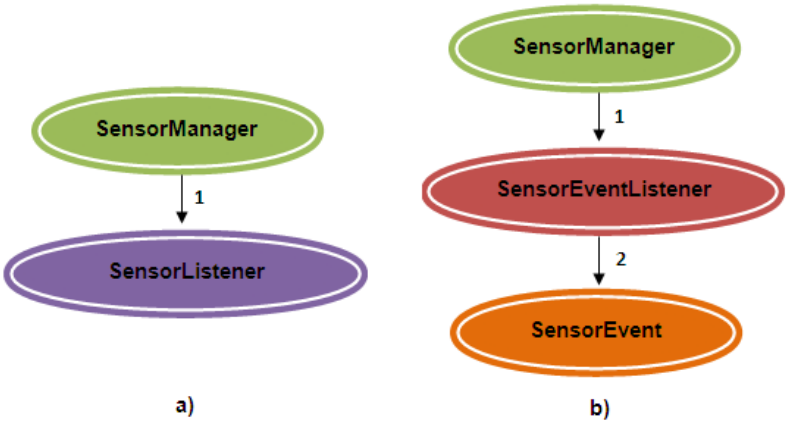


Figura 5.15: Algoritmos para utilización del acelerómetro. a) Algoritmo 1, 1) Notifica el evento de cambio de estado del sensor. b) Algoritmo 2, 1) Notifica el evento de cambio de estado del sensor, 2) Entrega la información del sensor al recibir el evento *onSensorChanged(SensorEventevent)*.

**Algoritmo** Para el estudio del Acelerómetro se encontraron 2 algoritmos principales, que corresponden a diferentes niveles de las API's, estos se pueden observar en la figura 5.15.

**Descripción de los algoritmos** Los algoritmos de la figura 5.15 se implementan siguiendo los pasos que se describen a continuación.

- **Algoritmo 1:**
  - *a*) Obtener referencia al *SensorManager*.

```
SensorManager sm =
(SensorManager) getSystemService(SENSOR_SERVICE);
```

  - *b*) Seleccionar el tipo de sensor.

```
static final int sensor = SensorManager.
SENSOR_ACCELEROMETER;
```

  - *c*) Registrar el listener.

```
sm.registerListener(this, sensor);
```

  - *d*) Cancelar el registro del listener.

```
sm.unregisterListener(this);
```

- **Algoritmo 2:**
  - *a*) Obtener referencia al *SensorManager*.

```
SensorManager sm =
(SensorManager)getSystemService(SENSOR_SERVICE);
```

  - *b*) Seleccionar el tipo de sensor.

```
Sensor mAccelerometer = sm.getDefaultSensor(
Sensor.TYPE\_ACCELEROMETER);
```

*c*) Registrar el listener con el tiempo de retardo entre mediciones.

```
int retardo = SensorManager.SENSOR_DELAY_GAME;
sm.registerListener(this, mAccelerometer, retardo);
```

**Nota:** el tiempo de retardo puede seleccionarse entre las constantes asignadas por android (SENSOR_DELAY_FASTEST= 0 ms, SENSOR_DELAY_GAME= 20 ms, SENSOR_DELAY_UI= 67ms, SENSOR_DELAY_NORMAL= 200 ms) o especificarse un valor entero en ms.

*d*) cancelar el registro del listener.

```
sm.unregisterListener(this);
```

### Selección de escenario inicial

Para la medición del consumo del acelerómetro en el dispositivo móvil se presenta el siguiente escenario:

- Modo avión.
- Pantalla con brillo mínimo.
- GPS, WIFI, auto sincronización y Rotación de pantalla desactivados.
- Aplicaciones en segundo plano desactivadas.
- Dispositivo sin movimiento y con movimiento.

### Implementación inicial

En base a la clase SensorManager y los dos algoritmos mencionados anteriormente, se implementa la aplicación “Acelerometro_1”, la cual se puede observa en la figura 5.16. Esta aplicación consta de tres botones. El botón “Usar SENSOR_ACCELEROMETER” inicia las mediciones del acelerómetro usando el algoritmo 1, el botón“Usar TYPE_ACCELEROMETER” inicia las mediciones del acelerómetro usando el algoritmo 2, el botón “Detener” detiene la medición de cualquiera de las dos opciones que se haya iniciado.

### Medición del consumo inicial

Para la medición del consumo se agrega una marca de alto consumo al inicio de la aplicación, una de bajo consumo al iniciar la medición del acelerómetro usando el algoritmo 1 o 2, y una de alto consumo cuando se detiene la medición.

*a*) Como primer paso se mide el consumo de la interfaz de la aplicación “Acelerometro_1” bajo el escenario descrito. Se realiza la prueba 5 veces obteniendo un valor de corriente promedio de 98,985157mA y una potencia promedio de 396,759mW. La figura 5.17 muestra la gráfica de corriente en una de las 5 repeticiones.

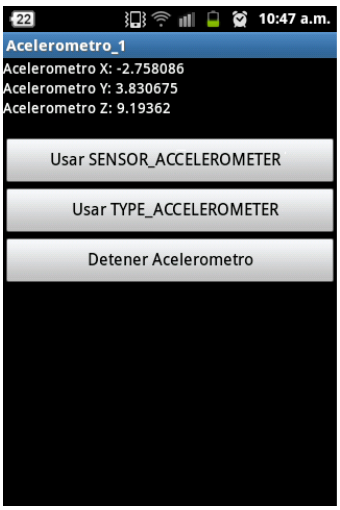


Figura 5.16: Aplicación “Acelerometro 1”.

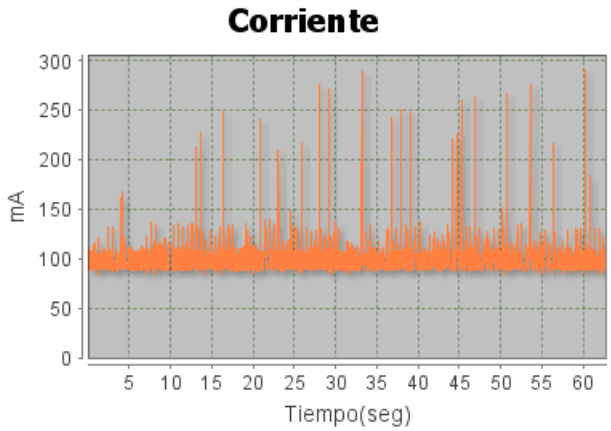


Figura 5.17: Medición interfaz aplicación “Acelerometro_1”.

*b*) Se mide el consumo del Acelerómetro ejecutando la aplicación “Acelerometro_1”, esta aplicación se ejecuta con o sin movimiento utilizando los 2 algoritmos a estudiar. Se realizan 5 repeticiones para cada caso específico.

1) **Sin movimiento:** La figura 5.18 a) muestra el consumo del acelerómetro utilizando el algoritmo 1 en donde se obtuvo un promedio de 216,1224337mA y 851,548304mW. La figura 5.18 b) muestra el consumo del acelerómetro utilizando el algoritmo 2 en donde se obtuvo un promedio de 234,6819849mA y 922,295215mW.
2) **Con movimiento:** La figura 5.19 a) muestra el consumo del acelerómetro utilizando el algoritmo 1 en donde se obtuvo un promedio de 236,6262358mA y 936,415797mW. La figura 5.19 b) muestra el consumo del acelerómetro utilizando el algoritmo 2 en donde se obtuvo un promedio de 234,8654626mA y 924,200622mW.

*c*) Se calcula el consumo del Acelerómetro restando el consumo de la interfaz, obteniendo como promedio los resultados de la tabla 5.7. En la figura 5.20 se muestra el gráfico de barras correspondiente al promedio de cada prueba, incluyendo el promedio de la interfaz.

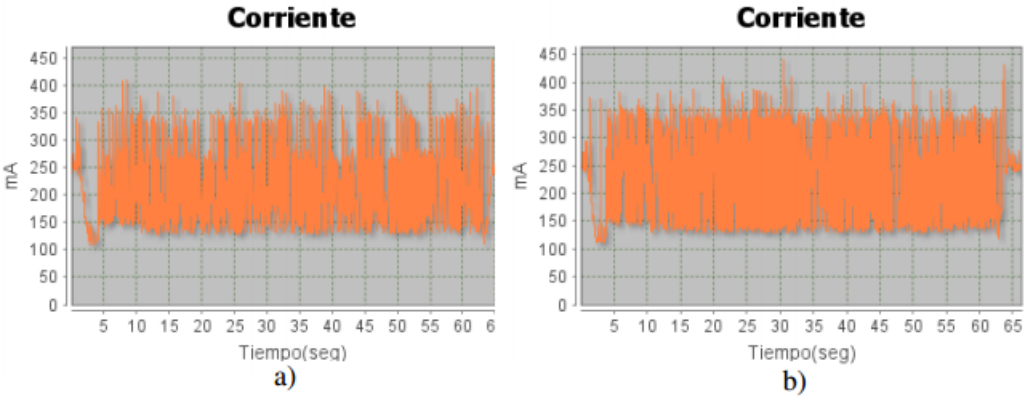


Figura 5.18: Medición Acelerómetro sin movimiento. a) Algoritmo 1. b) Algoritmo 2.

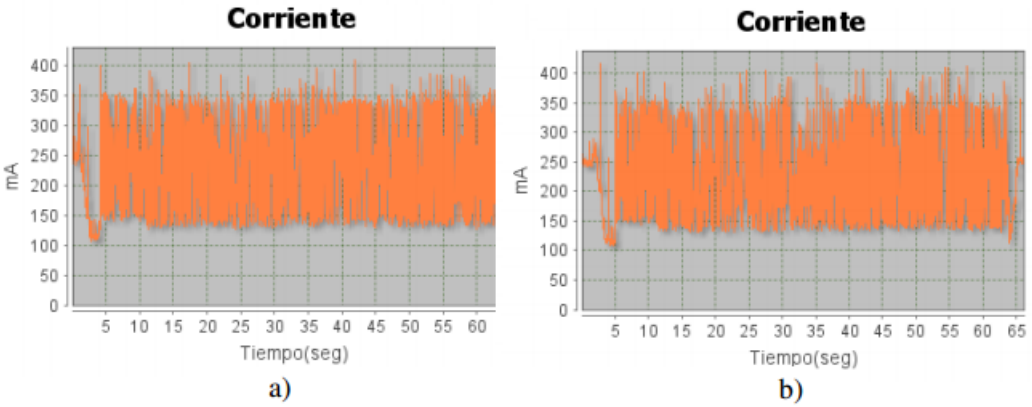


Figura 5.19: Medición Acelerómetro con movimiento. a) Algoritmo 1. b) Algoritmo 2.

| Algoritmo | Promedio sin movimiento | | | | Promedio con movimiento | | | |
|---|---|---|---|---|---|---|---|---|
| | Corriente (mAh) | Potencia (mW) | Corriente sin interfaz (mAh) | Potencia sin interfaz (mW) | Corriente (mAh) | Potencia (mW) | Corriente sin interfaz (mAh) | Potencia sin interfaz (mW) |
| 1 | 216,122 | 851,548 | 117,137 | 454,789 | 236,626 | 936,416 | 137,641 | 539,657 |
| 2 | 234,682 | 922,295 | 135,697 | 525,536 | 234,865 | 924,201 | 135,880 | 527,441 |

Tabla 5.7: Consumo promedio Acelerómetro.

**Análisis del consumo inicial**

En las figuras anteriores se puede observar que después del pico de consumo de bajada se presentan picos significativos dado la constante variación del sensor, aun estando el dispositivo en reposo (sin movimiento), esto se puede corroborar observando la interfaz de la aplicación en el dispositivo, la cual presenta una variación rápida y constante de valores. Analizando los resultados de la tabla se puede apreciar que el algoritmo 1 presenta un mayor consumo con movimiento que sin movimiento (la diferencia es de 20mA aproximadamente), que es lo esperado de este sensor; en cambio el algoritmo 2 no posee diferencia entre los 2 estados, por tanto genera más consumo que el algoritmo 1. Aunque el algoritmo 1 presenta un menor consumo este ha sido declarado obsoleto y remplazado por el algoritmo 2, por tanto se utilizará el algoritmo 2 en busca de disminuir su consumo.

**Preguntas:** Del anterior análisis resultan las siguientes preguntas:

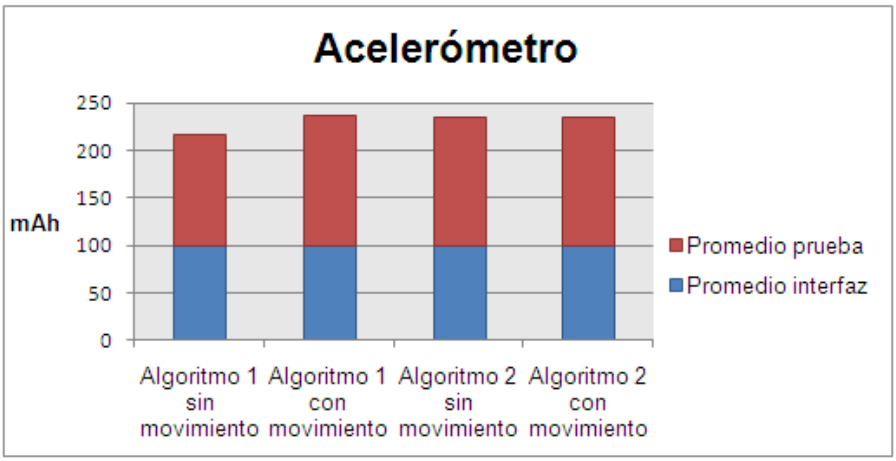


Figura 5.20: Promedios de corriente,caracterización inicial Acelerómetro.

- *¿Es posible generar un consumo menor si se modifica el tiempo que se establece en el registerListener()?, (en el registerListener() se indica cada cuanto el SensorEventlistener recibe valores).*
- *¿El cambio de valores en la interfaz de la aplicación genera un consumo considerable?, ¿Cada cuanto se deben desplegar los valores en la aplicación sin disminuir la apreciación de los cambios por el usuario?*

### 5.2.4. Brújula

#### Selección de algoritmo

La obtención de los datos necesarios para una brújula se puede realizar por medio de dos tipos de sensores que provee Android, estos son los sensores primarios y los sensores sintéticos. los primarios son sensores físicos del dispositivo de los cuales se obtiene los datos sin un previo procesamiento, en este caso son necesarios el magnetómetro y el acelerómetro. Los sintéticos son sensores creados a partir del procesamiento de información de uno o más sensores primarios, en este caso puntual existe el *SENSOR.TYPE_ORIENTATION* el cual se puede implementar de dos formas dependiendo del nivel API que se utilice. A partir de los sensores anteriormente nombrados se encontraron tres diferentes algoritmos para este estudio que utilizan la clase *SensorManager*, los algoritmos 1 y 2 siguen el mismo patrón de la parte a) de la figura 5.15 por estar utilizando la misma clase, y el algoritmo tres utiliza el mismo patrón de la parte b) de la figura 5.15, pero utilizando dos sensores al mismo tiempo.

**Descripción de los algoritmos** Los 3 algoritmos se implementan siguiendo los pasos que se describen a continuación.

- **Algoritmo 1:**

  *a*) Obtener referencia al *SensorManager*.

```
SensorManager sm =
(SensorManager) getSystemService(SENSOR_SERVICE);
```

  *b*) Seleccionar el tipo de sensor.

```
static final int sensor = SensorManager.
SENSOR_ORIENTATION;
```

*c*) Registrar el listener.

```
sm.registerListener(this, sensor);
```

*d*) Cancelar el registro del listener.

```
sm.unregisterListener(this);
```

- **Algoritmo 2:**

*a*) Obtener referencia al *SensorManager*.

```
SensorManager sm =
(SensorManager) getSystemService(SENSOR_SERVICE);
```

*b*) Seleccionar el tipo de sensor.

```
Sensor mOrientation =sm.getDefaultSensor(
Sensor.TYPE_ORIENTATION);
```

*c*) Registrar el listener con el tiempo de retardo entre mediciones.

```
int retardo = SensorManager.SENSOR_DELAY_GAME;
sm.registerListener(this, mOrientation, retardo);
```

*d*) cancelar el registro del listener.

```
sm.unregisterListener(this);
```

- **Algoritmo 3:**

*a*) Obtener referencia al SensorManager.

```
SensorManager sm =
(SensorManager) getSystemService(SENSOR_SERVICE);
```

*b*) Seleccionar el tipo de sensor.

```
Sensor mAccelerometer = sm.getDefaultSensor(
Sensor.TYPE_ACCELEROMETER);
Sensor mField = sm.getDefaultSensor(
Sensor.TYPE_MAGNETIC_FIELD);
```

*c*) Registrar el listener con el tiempo de retardo entre mediciones.

```
int retardo = SensorManager.SENSOR_DELAY_GAME;
sm.registerListener(this, mAccelerometer, retardo);
sm.registerListener(this, mField, retardo);
```

*d*) cancelar el registro del listener.

```
sm.unregisterListener(this);
```

### Selección de escenario inicial

Para la medición del consumo del acelerómetro el dispositivo móvil se presenta el siguiente escenario:

- Modo avión.
- Pantalla con brillo mínimo.
- GPS, WIFI, auto sincronización y rotación de pantalla desactivados.
- Aplicaciones en segundo plano desactivadas.
- Dispositivo sin movimiento y con movimiento.

### Implementación inicial

En base a la clase *SensorManager* y los tres algoritmos mencionados anteriormente, se implementa la aplicación "Brujula_inicial", la cual se puede observa en la figura 5.21. Esta aplicación consta de cuatro botones. El botón "Usar SensorManager.SENSOR_ORIENTATION" inicia las mediciones de la brújula usando el algoritmo 1, el botón "Usar Sensor.TYPE_ORIENTATION" inicia las mediciones de la brújula usando el algoritmo 2, el botón "Usar SensorManager.getOrientation()" inicia las mediciones de la brújula usando el algoritmo 3 y el botón "Detener" detiene la medición de cualquiera de las tres opciones que se haya iniciado.

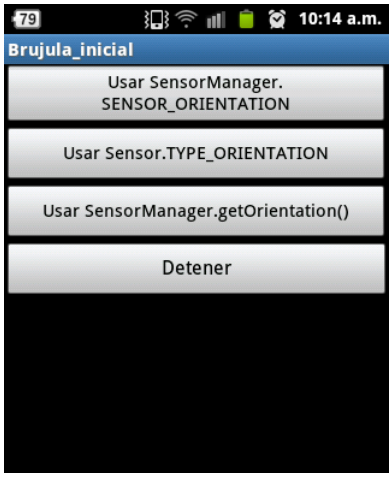


Figura 5.21: Aplicación "Brujula inicial".

### Medición del consumo inicial

Para la medición del consumo se agrega una marca de alto consumo al inicio de la aplicación, una de bajo consumo al iniciar la medición de la brújula usando el algoritmo 1, 2 o 3, y una de alto consumo cuando se detiene la medición.

*a*) Como primer paso se mide el consumo de la interfaz de la aplicación "Brujula_inicial" bajo el escenario descrito. Esta prueba se ejecutó la aplicación en el dispositivo Nexus S encontrando que solo funciona el algoritmo 3, los otros dos algoritmos no presenta error en la aplicación pero no obtienen valores lo cual se atribuyo después de una serie de pruebas a que el dispositivo necesita la conectada directamente. Por el anterior motivo se realiza la caracterización de la brújula

en el dispositivo Samsung Galaxy ACE que posee Android 2.3.6. Se realizaron las prueban 5 veces obteniendo un valor de corriente promedio de 95,4897423mA y una potencia promedio de 378,283326mW. La figura 5.22 muestra la gráfica de corriente en una de las 5 repeticiones.

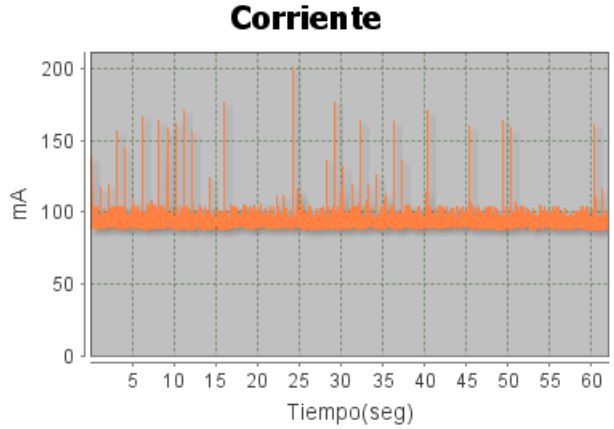


Figura 5.22: Medición interfaz aplicación “Brujula_inicial”.

*b*) Se mide el consumo de la brújula ejecutando la aplicación “Brujula_inicial”, esta aplicación se ejecuta con o sin movimiento utilizando los 3 algoritmos a estudiar. Se realizan 5 repeticiones para cada caso específico.

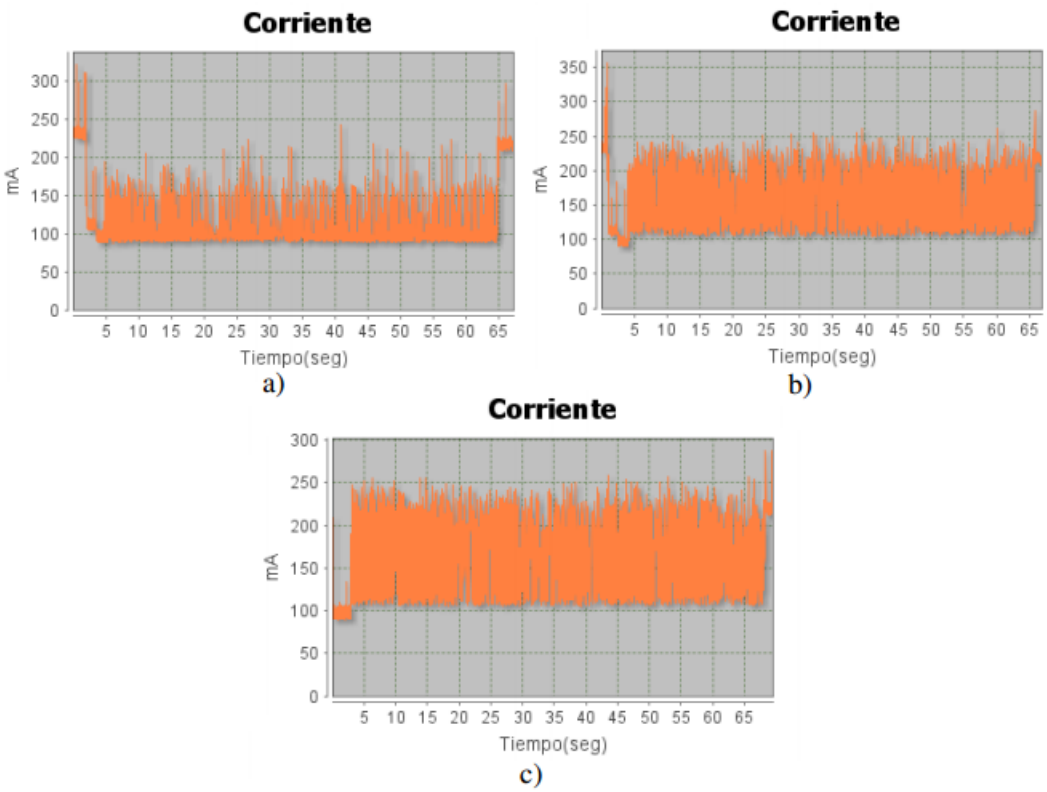


Figura 5.23: Medición brújula sin movimiento. a) Algoritmo 1. b) Algoritmo 2. c) Algoritmo 3.

1) **Sin movimiento:** La figura 5.23 a) muestra el consumo de la brújula utilizando el algoritmo 1 en donde se obtuvo un promedio de 105,8817269mAh y 417,647229mW. La figura

5.23 b) muestra el consumo de la brújula utilizando el algoritmo 2 en donde se obtuvo un promedio de 158,155819mA y 618,441124mW. La figura 5.23 c) muestra el consumo de la brújula utilizando el algoritmo 3 en donde se obtuvo un promedio de 164,3611041mA y 639,179737mW.

2) **Con movimiento:** La figura 5.24 a) muestra el consumo de la brújula utilizando el algoritmo 1 en donde se obtuvo un promedio de 109,2059756mA y 431,800124mW. La figura 5.24 b) muestra el consumo de la brújula utilizando el algoritmo 2 en donde se obtuvo un promedio de 158,6130972mA y 623,196554mW. La figura 5.24 c) muestra el consumo de la brújula utilizando el algoritmo 3 en donde se obtuvo un promedio de 164,2107287mA y 641,630399mW.

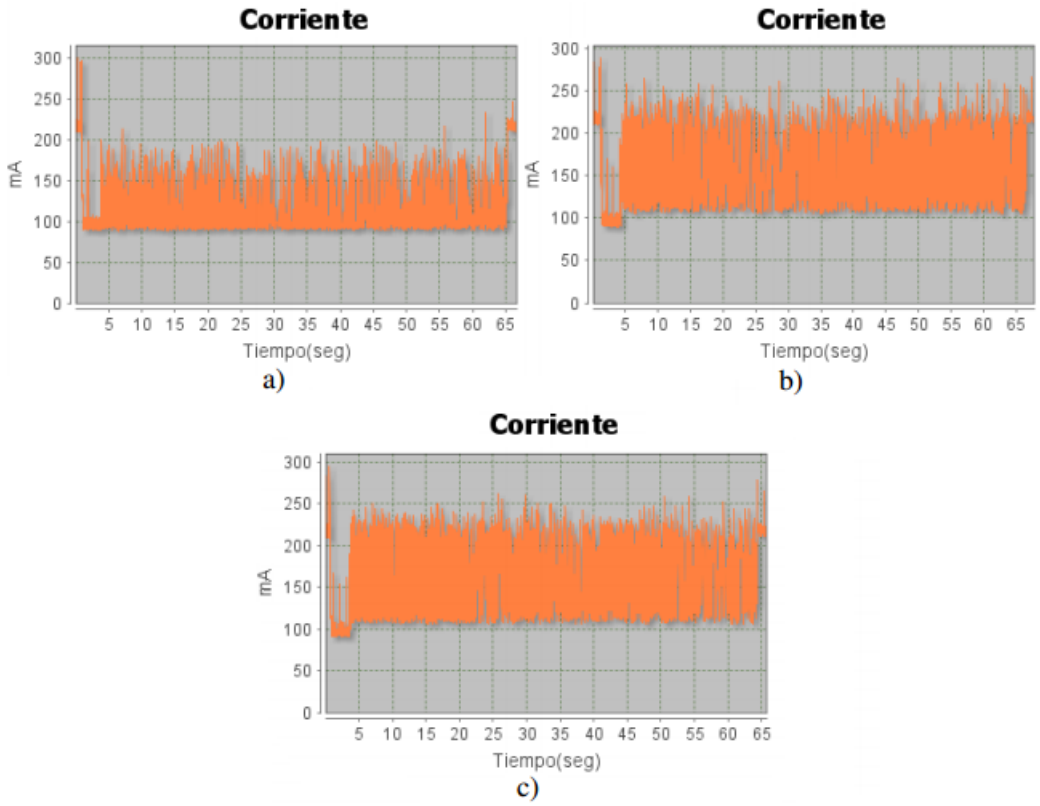


Figura 5.24: Medición brújula con movimiento. a) Algoritmo 1. b) Algoritmo 2. c) Algoritmo 3.

*c*) Se calcula el consumo de la Brújula restando el consumo de la interfaz, obteniendo como promedio los resultados de la tabla 5.8.

| Algoritmo | Promedio sin movimiento | | | | Promedio con movimiento | | | |
|---|---|---|---|---|---|---|---|---|
| | Corriente (mAh) | Potencia (mW) | Corriente sin interfaz (mAh) | Potencia sin interfaz (mW) | Corriente (mAh) | Potencia (mW) | Corriente sin interfaz (mAh) | Potencia sin interfaz (mW) |
| 1 | 105,881 | 417,647 | 10,391 | 39,3639 | 109,205 | 431,8 | 13,716 | 53,516 |
| 2 | 158,155 | 618,441 | 62,666 | 240,157 | 158,613 | 623,196 | 63,123 | 244,913 |
| 3 | 164,361 | 639,179 | 68,871 | 260,896 | 164,21 | 641,630 | 68,720 | 263,347 |

Tabla 5.8: Consumo promedio Brújula.

En la figura 5.25 se muestra el gráfico de barras correspondiente al promedio de cada prueba, incluyendo el promedio de la interfaz.

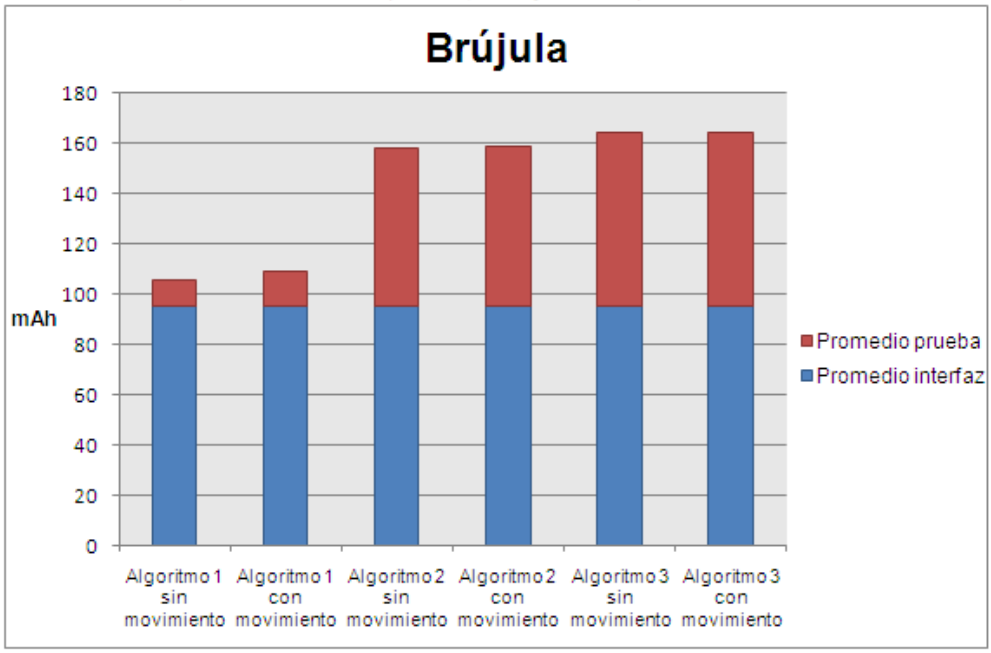


Figura 5.25: Promedios de corriente, caracterización inicial brújula.

### Análisis del consumo inicial

En las figuras anteriores se puede observar que después del pico de consumo de bajada se presentan picos significativos dado la constante variación del sensor, aun estando el dispositivo en reposo (sin movimiento), esto se puede corroborar observando la interfaz de la aplicación en el dispositivo, la cual presenta una variación rápida y constante de valores. Analizando los resultados de la tabla se puede apreciar que en los 3 algoritmos no se presentan diferencias significativas del consumo estando en movimiento y sin movimiento. Los algoritmos 1 y 2 presentan un menor consumo que el algoritmo 3, pero estos ha sido declarados obsoletos y remplazados por el algoritmo 3, por tanto se utilizará el algoritmo 3 en busca de disminuir su consumo.

**Preguntas:** Del anterior análisis resultan las siguientes preguntas:

- *¿Cómo la brújula utiliza 2 sensores en el algoritmo 3, es posible generar un consumo menor si se modifican los tiempos que se establecen en el registerListener() de forma diferente en cada sensor?, (el registerListener() cual es el método que indica cada cuanto el listener recibe valores).*
- *¿El cambio de valores en la interfaz de la aplicación genera un consumo considerable?, ¿Cada cuanto se deben desplegar los valores en la aplicación sin disminuir la apreciación de los cambios por el usuario?*

### 5.2.5. Teclado

#### Selección de algoritmo

En este caso se estudiara las entradas de teclado táctil existentes en el sistema operativo Android 4.1.2 por tanto no es necesario seleccionar un algoritmo para la medición.

#### Selección de escenario inicial

Para la medición del consumo del audio el dispositivo móvil se presenta el siguiente escenario:

- Modo avión
- Pantalla con brillo mínimo
- GPS, WIFI, auto sincronización y Rotación de pantalla desactivados.
- Aplicaciones en segundo plano desactivadas.
- Teclado QWERTY.

Adicionalmente se implementa una aplicación llamada “EscenarioTeclado” que complementa al escenario inicial, debido a que las aplicaciones propias del dispositivo pueden provocar consumos inesperados y no permiten hacer uso de las marcas. En la figura 5.26 se puede apreciar la interfaz de la aplicación.

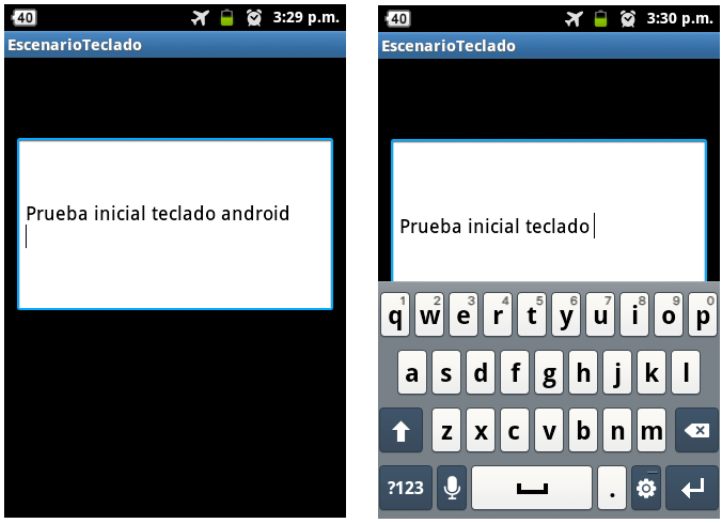


Figura 5.26: Aplicación “EscenarioTeclado”.

#### Implementación inicial

Debido a que no se hace una utilización de algoritmos de Android para controlar el teclado este paso es omitido.

### Medición del consumo inicial

Para la medición del consumo se agrega una marca de alto consumo al inicio de la aplicación, una de bajo consumo al seleccionar el cuadro de texto, y una de alto consumo cuando se cuando se presiona "ENTER" en el teclado.

*a*) Como primer paso se mide el consumo de la interfaz de la aplicación bajo el escenario descrito. Se realiza la prueba 5 veces en posición vertical y en posición horizontal, obteniendo un valor de corriente promedio de 116,407343mA y una potencia promedio de 463,486mW en posición vertical, y obteniendo un valor de corriente promedio de 125,9532968mAh y una potencia promedio de 497,088789mW en posición horizontal. La figura 5.27 muestra la gráfica de corriente en posición vertical y la figura 5.28 muestra la gráfica de corriente en posición horizontal.

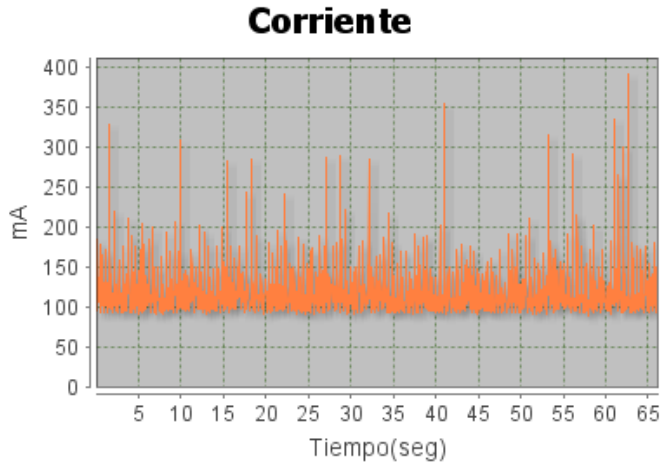


Figura 5.27: Medición interfaz de la aplicación "EscenarioTeclado" con el dispositivo en posición vertical.

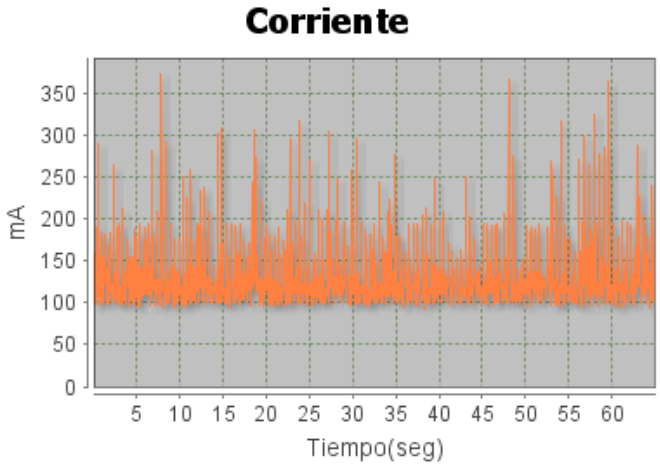


Figura 5.28: Medición interfaz de la aplicación "EscenarioTeclado" con el dispositivo en posición horizontal.

*b*) Para caracterizar el consumo del teclado se escribió la frase "hola android" escribiendo las letras en intervalos de un segundo con la ayuda de un metrónomo. La prueba se realizó en el teclado QWERTY configurado por defecto en el dispositivo, utilizando la posición vertical y horizontal de este. Se calculará el consumo al presionar cada letra por lo cual se hacen dos repeticiones de cada prueba para comparar.

1) **Teclado posición vertical:** La figura 5.29 se muestra las dos repeticiones de la medición del teclado en posición vertical y en la tabla 5.9 se encuentran los valores obtenidos en estas dos pruebas.

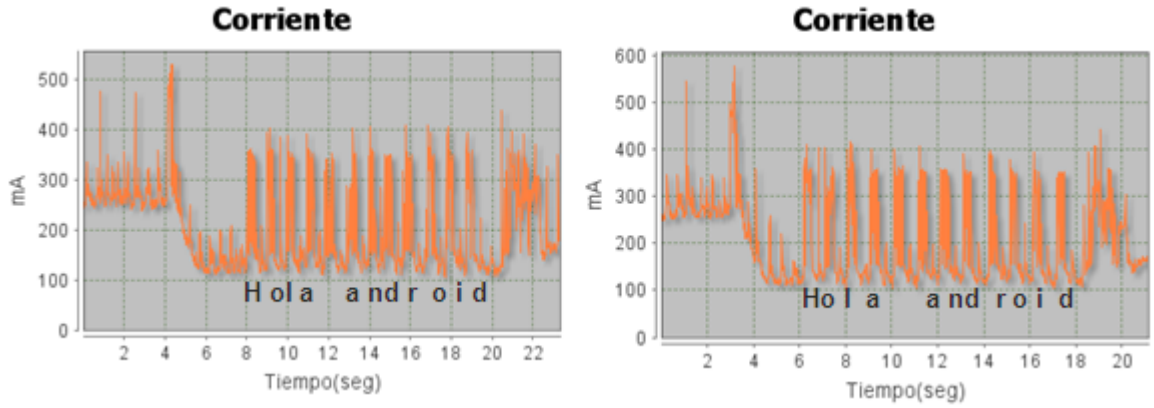


Figura 5.29: Medición teclado QWERTY posición vertical.

| Prueba | Valores | h | o | l | a | a | n | d | r | o | i | d |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Corriente (mAh) | 192,06 | 211,85 | 193,71 | 231,48 | 198,92 | 189,12 | 204,76 | 210,67 | 211,22 | 239,98 | 235,04 |
| | Tiempo (s) | 0,915 | 0,613 | 0,882 | 0,551 | 0,82 | 0,803 | 0,806 | 0,714 | 0,857 | 0,561 | 0,503 |
| 2 | Corriente (mAh) | 203,69 | 208,57 | 215,058 | 227,8 | 197,91 | 209,92 | 195,65 | 194,73 | 198,83 | 198,23 | 190,26 |
| | Tiempo (s) | 0,812 | 0,646 | 0,744 | 0,646 | 0,684 | 0,721 | 0,714 | 0,820 | 0,716 | 0,753 | 0,769 |

Tabla 5.9: Promedios de corriente por letras en teclado QWERTY posición vertical.

2) **Teclado posición horizontal:** La figura 5.30 se muestra las dos repeticiones de la medición del teclado en posición horizontal y en la tabla 5.10 se encuentran los valores obtenidos en estas dos pruebas.

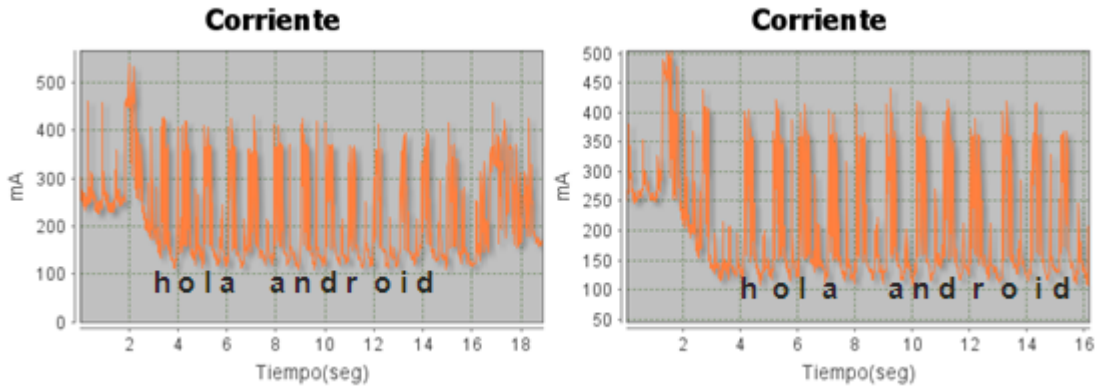


Figura 5.30: Medición teclado QWERTY posición horizontal.

| Prueba | Valores | h | o | l | a | a | n | d | r | o | i | d |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Corriente (mAh) | 192,06 | 211,85 | 193,71 | 231,48 | 198,92 | 189,12 | 204,76 | 210,67 | 211,22 | 239,98 | 235,04 |
| | Tiempo (s) | 0,915 | 0,613 | 0,882 | 0,551 | 0,82 | 0,803 | 0,806 | 0,714 | 0,857 | 0,561 | 0,503 |
| 2 | Corriente (mAh) | 203,69 | 208,57 | 215,058 | 227,8 | 197,91 | 209,92 | 195,65 | 194,73 | 198,83 | 198,23 | 190,26 |
| | Tiempo (s) | 0,812 | 0,646 | 0,744 | 0,646 | 0,684 | 0,721 | 0,714 | 0,820 | 0,716 | 0,753 | 0,769 |

Tabla 5.10: Promedios de corriente por letras en teclado QWERTY posición horizontal.

**Análisis del consumo inicial**

Al observar y analizar las gráficas se encuentra que los pulsos generados por presionar las teclas presentan valores de consumo que varían, aunque presentan patrones similares, se puede notar que el consumo depende de factores como el área y tiempo de presión en la pantalla. Para corroborar lo anterior se puede observar que en la pantalla horizontal se presentan pulsos mayores a los alcanzados en la pantalla vertical, esto debido a que el área de las letras es más grande en esta posición. En la tabla 5.11 se puede apreciar los promedios de corriente de presionar las letras en las dos posiciones del teclado. Teniendo en cuenta que el consumo generado depende del tiempo que dure el pulso de consumo, es decir, entre más rápido se escriba menor será el ancho de este pulso.

| | Teclado Vertical | | Teclado Horizontal | |
|---|---|---|---|---|
| Prueba | 1 | 2 | 1 | 2 |
| Promedio corriente (mAh) | 208,93 | 203,01 | 225,17 | 218,88 |
| Promedio corriente sin interfaz (mAh) | 92,52 | 86,6 | 99,22 | 92,93 |
| Tiempo promedio(s) | 0,7316 | 0,731 | 0,635 | 0,654 |

Tabla 5.11: Promedios de corriente y tiempo en pulsos de teclado QWERTY en las dos posiciones.

**Preguntas:** Dado el escenario presente y los resultados del teclado en las dos posiciones surgen las siguientes preguntas:

- *¿Utilizar teclado predictivo permite escribir mas palabras presionando menos veces la pantalla, esto genera un menor consumo?*
- *¿La interfaz del teclado contribuye significativamente al consumo?*
- *¿Otros teclados disponibles y mejor raqueados en el Google Play presentan un menor consumo que el teclado QWERTY del dispositivo?*

## 5.3. Diseño de Pruebas

Para el diseño de pruebas se crea una plantilla que será utilizada en los 5 periféricos y en las diversas pruebas que se puedan diseñar para cada uno. La plantilla se puede observar en la tabla 5.12. A continuación se describe un ejemplo del diseño de pruebas, los demás diseños se encuentran en el anexo E.

| Plantilla de diseño de pruebas | |
|---|---|
| Periférico | |
| Práctica de programación a evaluar | |
| Especificación del escenario | |
| Hipótesis | |
| Descripción software o aplicación | |

Tabla 5.12: Plantilla para el diseño de pruebas.

### 5.3.1. Ejemplo de un diseño de prueba sobre GPS

En la tabla 5.13 se observar un ejemplo de una prueba diseñada para el periférico GPS utilizando la plantilla.

| Prueba nº1 | |
|---|---|
| Periférico | GPS |
| Práctica de programación a evaluar | Variación de los parámetros minTime y minDistance en el registro de actualizaciones requestLocationUpdates(). |
| Especificación del escenario | • Modo avión.<br>• Pantalla con brillo mínimo.<br>• GPS, WIFI, auto sincronización y Rotación de pantalla desactivados.<br>• Aplicaciones en segundo plano desactivadas<br>• Lugar: exteriores<br>• Dispositivo móvil estático y en movimiento (a 60Km/h). |
| Hipótesis | • Aumentar los parámetros minTime y minDistance disminuye las actualizaciones y esto a su vez el consumo. |
| Descripción software o aplicación | Se modifica la aplicación "Localización" variando los parámetros minTime y minDistance. |

Tabla 5.13: Plantilla para el diseño de pruebas.

# Capítulo 6

# Ejecución y análisis de pruebas

Este capítulo contiene la ejecución y análisis de pruebas sobre los periféricos GPS, sistema de audio (auricular y altavoz), acelerómetro, brújula y teclado utilizando el proceso de ejecución de pruebas P.E.P. y el proceso de análisis de mediciones P.A.M del modelo para la evaluación de consumo de energía. Como medidas principales para el análisis del consumo se utilizan el promedio de corriente, tiempo de ejecución de las pruebas y el tiempo de duración de la batería.

## 6.1. GPS

### 6.1.1. Prueba 1: Variación de parámetros minTime y minDistance.

Variación de los parámetros minTime y minDistance en el registro de actualizaciones requestLocationUpdates(). La descripción completa de esta prueba se encuentra en el anexo E tabla E.1.

#### Ejecución

*a*) **Caracterización del consumo inicial**: se utiliza como base caracterización inicial del GPS realizada en el capítulo 5 en donde se obtuvieron los promedio de corriente y potencia de la interfaz de la aplicación "Localización" correspondientes a 100,074874mA y 0,397946W.

*b*) **Ejecución inicial**: Se realiza la medición del consumo de energía bajo el escenario descrito con el dispositivo estático, corriendo la aplicación y variando el parámetrominTime, como resultado se obtuvo los datos de la tabla 6.1.

| Tiempo minTime (ms) | Promedio corriente (mA) | Promedio corriente sin interfaz (mA) |
|---|---|---|
| 1000 | 157,442 | 57,367 |
| 2500 | 154,666 | 54,591 |
| 5000 | 154,394 | 54,319 |

Tabla 6.1: Promedios de corriente GPS prueba 1 - variación del parámetro minTime.

*c*) **Ejecución secundaria**: Se realiza una segunda ejecución variando el parámetro minDistance con el dispositivo en moviendo a una velocidad constante de 60Km/h (16.67m/s). La tabla 6.2 muestra los promedios obtenidos.

| Distancia minDistance (mt) | Promedio corriente (mA) | Promedio corriente sin interfaz (mA) |
|---|---|---|
| 5 | 164,43 | 64,355 |
| 10 | 167,834 | 67,759 |
| 15 | 161,696 | 61,621 |

Tabla 6.2: Promedios de corriente GPS prueba 1 - variación del parámetro minDistance.

### Análisis de mediciones

*a*) **Análisis comparativo:** Como primera medida se analizan y comparan las gráficas de los consumos en los 3 diferentes tiempos de actualizaciones tomando como muestra el segmento posterior a la activación del servicio, la figura 6.1 presenta las 3 gráficas.

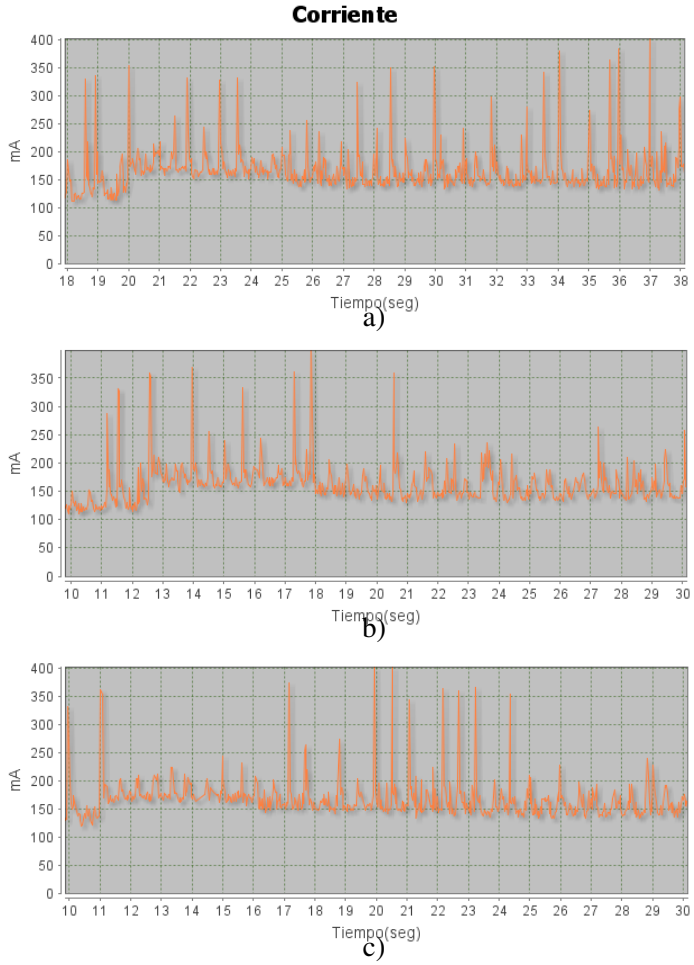


Figura 6.1: Medición GPS prueba 1 - Variación del parámetro minTime. a) 1000ms. b) 2500ms. c) 5000ms.

De la figura 6.1 se puede apreciar que:

- La gráfica a) presenta más picos que la b) y la c).
- Las gráficas b) y c) presentan un consumo muy similar en cuanto a número de picos.

- Se puede apreciar que en los momentos donde se establecía una buena precisión (entre 15 y 10mt) los picos se generaban con un periodo mayor al que se presentaban cuando se inició el servicio. Lo anterior puede ser atribuido a que las actualizaciones también dependen de la variación de la posición y en el momento inicial el dispositivo no es muy preciso en su localización, al realizar una nueva medición esta varía considerablemente comparándola con la primera, lo cual genera nuevas actualizaciones sucesivamente hasta que encuentra una localización con una precisión mejor.

De la tabla 6.1 se obtiene el gráfico de barras del consumo del GPS variando el tiempo de actualizaciones. (Figura 6.2).

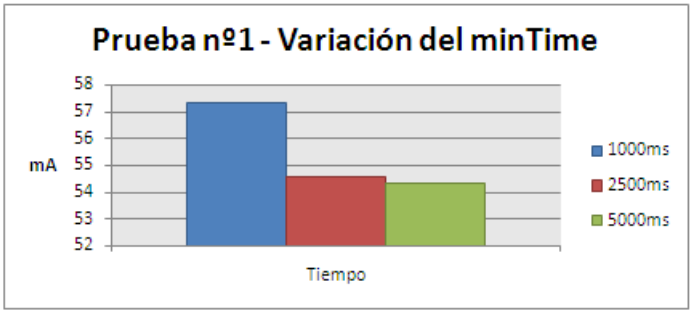


Figura 6.2: Promedio consumo GPS prueba 1 - Variación del parámetro minTime.

Del grafico presentado en la figura 6.2 se puede analizar lo siguiente:

- Aumentar el tiempo de las actualizaciones genera una reducción en el consumo del GPS, aunque este llega a un momento en el cual se estabiliza.

Continuando con la ejecución secundaria se analizan 3 diferentes distancias con el dispositivo moviéndose sobre un vehículo a una velocidad constante de 60Km/h, notando que el consumo aumenta debido al constante cambio de posición lo que hace que siempre haya actualizaciones al completarse el periodo de 1000ms. El recorrido utilizado para la prueba fue sobre la vía Variante de la ciudad de Popayán entre los puntos A y B (figura 6.3).


Figura 6.3: Recorrido Prueba 1 - Ejecución secundaria.

Observando la aplicación y comparando entre las 3 distancias se puede observar que las actualizaciones se generaban siempre cada 1000ms independiente que se tuviera una u otra distancia, para corroborar lo anterior se ejecutó la aplicación cambiando la variable minTime a 10000ms, la variable minDistance a 5mt y manteniendo la velocidad de 60Km/h dando como resultado que las actualizaciones de la localización se presentaban cada 10000ms, eran independientes del cambio de distancia para efectuarse. Por último se dejaron las variables en 10000ms y 5mt y se aumentó la velocidad del vehículo a 80Km/h presentando las mismas actualizaciones. Para comprobar que las actualizaciones sólo dependen del minTime se ejecuta la aplicación con minTime en 10000ms, minDistance en 5mt y el dispositivo sin movimiento, encontrando que las actualizaciones no se realizan cada 10000ms, hubo tiempos de 78000ms en donde no se presentaban cambios de la localización, es por esto que se concluye lo siguiente:

- El minTime es el tiempo que tomaría una actualización si se cumple una diferencia mínima de distancia equivalente al minDistance.

Al analizar la velocidad que entrega el GPS se pudo observar que al moverse en el vehículo a 60Km/h, que es 16.67m/s, el valor entregado por el servicio variaba entre 12m/s y 15m/s, lo cual se puede atribuir a la precisión del GPS.

De la tabla 6.2 se obtiene el gráfico de barras de la figura 6.4 que corresponde al consumo del GPS con un minTime de 1000ms y minDistance de 5, 10 y 15mt.

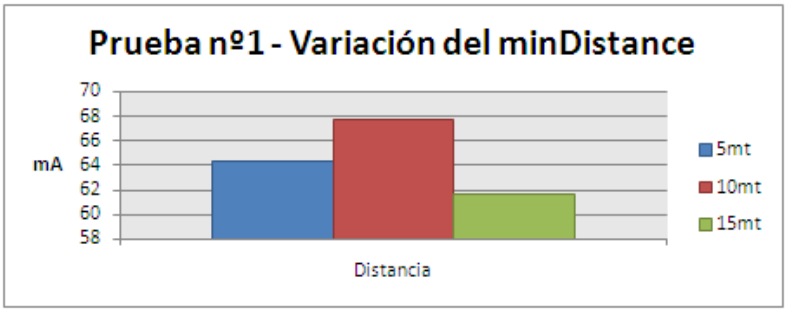


Figura 6.4: Promedio consumo GPS prueba 1 - variación del parámetro minDistance.

Del gráfico presentado en la figura 6.4 se puede analizar lo siguiente:

- No se presenta una tendencia del consumo debido a los diversos parámetros que no pueden controlarse en la prueba, como lo son la señal que establece el dispositivo con el satélite GPS, los cambios de dirección del recorrido y presencia de diversas nubes a lo largo del trayecto.
- En el caso de 15mt la velocidad que entrega el servicio de localización nunca fue mayor a 15m/s, por lo cual las actualizaciones no se presentaban cada 1000ms como con las anteriores distancias, es por esto que en 15mt se presentó un menor consumo.

*b*) **Comprobación de hipótesis** Se comprobó que aumentar los parámetros minTime y minDistance disminuye las actualizaciones y esto a su vez el consumo.

*c*) **Conclusiones**

- Aunque aumentar los parámetros minTime y minDistance disminuyen el consumo del GPS esta disminución no presenta un gran impacto sobre el consumo del dispositivo.
- El minTime es el tiempo que tomaría una actualización si se cumple una diferencia mínima de distancia equivalente al minDistance.

- Aun cuando el número de picos disminuye, el GPS presenta un valor de corriente mínima que se encuentra alrededor de 130mA con interfaz (30mA sin interfaz) aproximadamente, correspondiente al funcionamiento del periférico.
- Parámetros externos como los son las nubes, línea de vista con los satélites, constantes cambios en la geografía del trayecto, entre otros, son difíciles de controlar y afectan significativamente el consumo del GPS creando un esfuerzo mayor al dispositivo.
- Variar los parámetros minTime y minDistance con respecto a la velocidad del dispositivo puede generar un menor consumo en este.
- Determinar el contexto en el cual es utilizada la aplicación de localización puede ser una buena práctica para asignar los valores de minTime y minDistance.

## 6.2. AUDIO

### 6.2.1. Prueba 1: Géneros musicales.

Comparación entre géneros musicales utilizando el algoritmo inicial, los géneros son: salsa, rock, reggaeton y clásica. La descripción completa de esta prueba se encuentra en el anexo E tabla E.2.

#### Ejecución

*a*) **Caracterización del consumo inicial**: se utiliza la caracterización inicial del teclado realizada en el capítulo 5 en donde se encontró el promedio de la interfaz de la aplicación "Audiomp"(se utiliza la misma interfaz) obteniendo los valores de corriente y potencia de 107,397799mA y 0,438302W.

*b*) **Ejecución inicial**: Se realiza la medición del consumo de energía bajo el escenario descrito y corriendo la aplicación, como resultado se obtuvieron los datos de la tabla 6.9.

| Género | Promedio corriente (mA) | Promedio corriente sin interfaz (mA) |
|---|---|---|
| Salsa | 126,773 | 19,375 |
| Rock | 127,797 | 20,400 |
| Reggaeton | 128,802 | 21,404 |
| Clásica | 127,375 | 19,977 |

Tabla 6.3: Promedios de corriente audio prueba 1 - Géneros musicales.

*c*) **Ejecución secundaria**: no se realiza una ejecución secundaria ya que esta prueba no se definen parámetros variables.

#### Análisis de mediciones

*a*) **Análisis comparativo:** De la tabla 6.9 se obtiene el gráfico de barras del consumo de los géneros sin la interfaz (figura 6.5).

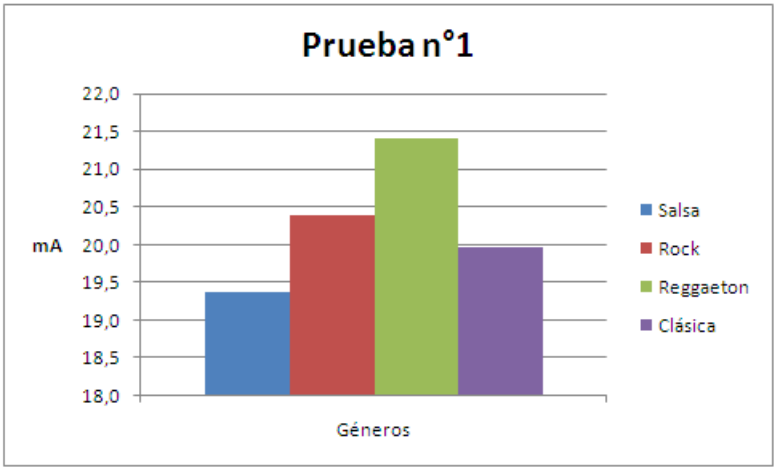


Figura 6.5: Promedio de consumo audio prueba 1 - Géneros musicales.

De la figura 6.5 se puede analizar lo siguiente:

- El género que tiene un consumo mayor es el reggaeton.
- Los géneros de rock y clásica poseen un consumo muy parecido.
- El género que tiene un menor consumo es la salsa.
- La diferencia entre el reggaeton y la salsa es de 2mA, lo cual no es muy significativo para el consumo general del dispositivo.

Al calcular la duración de la batería con respecto al consumo de corriente anterior y tomando la batería del Nexus S que es de 1500mAh, se obtienela tabla 6.4.

| Género | Promedio corriente (mA) | Tiempo de duración (min) | Promedio corriente sin interfaz (mA) | Tiempo de duración sin la interfaz (min) | Tiempo de duración sin la interfaz (días) |
|---|---|---|---|---|---|
| Salsa | 126,773 | 709,931 | 19,375 | 4645,152 | 3,226 |
| Rock | 127,797 | 704,24 | 20,4 | 4411,849 | 3,064 |
| Reggaeton | 128,802 | 698,748 | 21,404 | 4204,819 | 2,92 |
| Clásica | 127,375 | 706,574 | 19,977 | 4505,099 | 3,129 |

Tabla 6.4: Promedios de corriente y tiempo de duración de la batería - audio prueba 1.

*b*) **Comprobación de hipótesis** No se realiza ya que no se especificó en el diseño.

*c*) **Conclusiones**

- No se presenta una diferencia significativa entre las canciones utilizadas ya que fueron normalizadas.
- Reproducir el archivo de salsa Mosaico del Grupo niche produce un consumo sin interfaz de19.375mA, que corresponde a descargar la batería en 3 días, 5 horas y 25 minutos si se reproduce continuamente.
- Reproducir el archivo de salsa Mosaico del Grupo niche produce un consumo con interfaz de 126,773 mA, que corresponde a descargar la batería en 11 horas y 50 minutos si se reproduce continuamente.

- Reproducir el archivo de rock Back in black de AC/DC produce un consumo sin interfaz de 20.4mA, que corresponde a descargar la batería en 3 días, 1 horas y 32 minutos si se reproduce continuamente.
- Reproducir el archivo de rock Back in black de AC/DC produce un consumo con interfaz de 127,797mA, que corresponde a descargar la batería en 11 horas y 44 minutos si se reproduce continuamente.
- Reproducir el archivo de reggaeton Danza kuduro de Don Omar produce un consumo sin interfaz de 21.404mA, que corresponde a descargar la batería en 2 días, 22 horas y 5 minutos si se reproduce continuamente.
- Reproducir el archivo de reggaeton Danza kuduro de Don Omar produce un consumo con interfaz de 128,802mA, que corresponde a descargar la batería en 11 horas y 39 minuto si se reproduce continuamente s.
- Reproducir el archivo de clásica Sinfonía nº5 de Beethoven produce un consumo sin interfaz de 19,977mA, que corresponde a descargar la batería en 3 días, 3 horas y 6 minutos si se reproduce continuamente.
- Reproducir el archivo de clásica Sinfonía nº5 de Beethoven produce un consumo con interfaz de 127,375mA, que corresponde a descargar la batería en 11 horas y 46 minutos si se reproduce continuamente.

### 6.2.2. Prueba 2: Canción en estudio y en vivo.

Comparación entre una canción grabada en estudio y una en vivo. La descripción completa de esta prueba se encuentra en el anexo E tabla E.3.

#### Ejecución

*a*) **Caracterización del consumo inicial**: se utiliza la caracterización inicial del teclado realizada en el capítulo 5 en donde se encontró el promedio de la interfaz de la aplicación “Audiomp” (se utiliza la misma interfaz).

*b*) **Ejecución inicial**: Se realiza la medición del consumo de energía bajo el escenario descrito y corriendo la aplicación, como resultado se obtuvo los datos de la tabla 6.5.

| Grabación | Promedio corriente (mA) | Promedio corriente sin interfaz (mA) |
|---|---|---|
| Estudio | 124,574 | 17,176 |
| Vivo | 128,563 | 21,165 |

Tabla 6.5: Promedios de corriente audio prueba 2 - Canción en estudio y en vivo.

*c*) **Ejecución secundaria**: no se realiza una ejecución secundaria ya que esta prueba no se definen parámetros variables.

**Análisis de mediciones**

*a*) **Análisis comparativo:** De la tabla 6.5 se obtiene el gráfico de barras del consumo de los géneros sin la interfaz, este se puede observar en la figura.

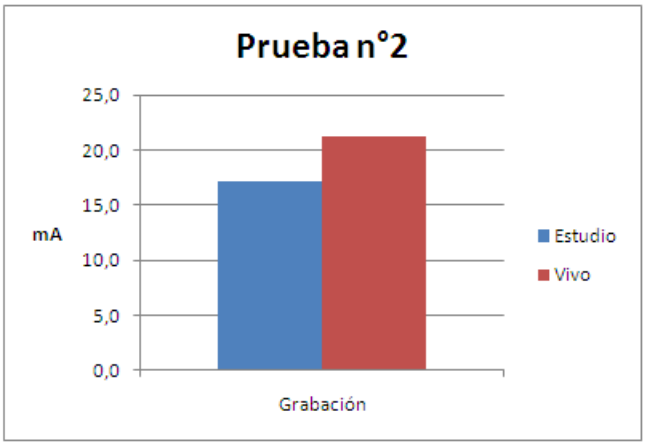


Figura 6.6: Promedios de corriente archivos audio prueba 2 - Canción en estudio y en vivo.

De la figura 6.6 se puede analizar lo siguiente:

- La grabación que más consume es la realizada en vivo.
- La diferencia entre la grabada en estudio y en vivo es de 4mA.

Al calcular la duración de la batería con respecto al consumo de corriente anterior y tomando la batería del Nexus S que es de 1500mAh, se obtiene la tabla 6.6.

| Grabación | Promedio corriente (mA) | Tiempo de duración (min) | Promedio corriente sin interfaz (mA) | Tiempo de duración sin la interfaz (min) | Tiempo de duración sin la interfaz (días) |
|---|---|---|---|---|---|
| Estudio | 124,574 | 722,462 | 17,176 | 5239,789 | 3,639 |
| Vivo | 128,563 | 700,046 | 21,165 | 4252,272 | 2,953 |

Tabla 6.6: Promedios y tiempo de duración audio prueba 2 - Canción en estudio y en vivo.

*b*) Comprobación de hipótesis: No se especificó hipótesis en el diseño.

*c*) **Conclusiones**

- La canción Back in Black de AC/DC grabada en vivo consume más que en estudio.
- La canción Back in Black de AC/DC grabada en vivo permite una duración de la batería 22 minutos mayor a la grabada en estudio si se reproducen continuamente.

### 6.2.3. Prueba 3: Reproducción con control de volumen.

Se evalúa si existe un menor consumo al utilizar el control de volumen como practica de programación usando diferentes niveles entre audífonos y altavoz. La descripción completa de esta prueba se encuentra en el anexo E tabla E.4.

### Ejecución

*a*) **Caracterización del consumo inicial**: se utiliza la caracterización inicial del teclado realizada en el capítulo 5 en donde se encontró el promedio de la interfaz de la aplicación "Audiomp" (se utiliza la misma interfaz).

*b*) **Ejecución inicial**: Se realiza la medición del consumo de energía bajo el escenario descrito y corriendo la aplicación, como resultado se obtuvo los datos de la tabla 6.7.

| Estado | Promedio corriente (mA) | Promedio corriente sin interfaz (mA) |
|---|---|---|
| Altavoz | 146,633 | 39,235 |
| Audífonos | 128,563 | 17,438 |

Tabla 6.7: Promedios de corriente audio prueba 3 - Reproducción con control de volumen.

*c*) **Ejecución secundaria**: no se realiza una ejecución secundaria ya que esta prueba no se definen parámetros variables.

### Análisis de mediciones

*a*) **Análisis comparativo:** De la tabla 6.7 se obtiene el gráfico de barras del consumo de los géneros sin la interfaz, este se puede observar en la figura 6.7.

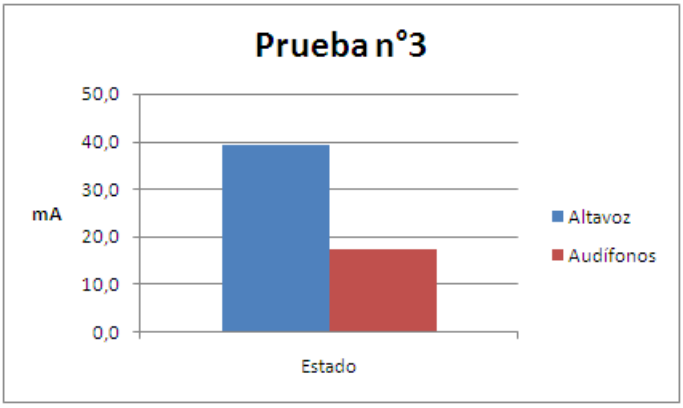


Figura 6.7: Promedios de corriente audio prueba 3- Reproducción con control de volumen.

| Estado | Promedio corriente (mA) | Tiempo de duración (min) | Promedio corriente sin interfaz (mA) | Tiempo de duración sin la interfaz (min) | Tiempo de duración sin la interfaz (días) |
|---|---|---|---|---|---|
| Altavoz | 146,633 | 613,778 | 39,235 | 2293,865 | 1,593 |
| Audífonos | 124,746 | 721,469 | 17,348 | 5187,998 | 3,603 |

Tabla 6.8: Promedios y tiempo de duración audio prueba 3 - Reproducción con control de volumen.

De la figura 6.7 se puede analizar lo siguiente:

- El consumo de la reproducción con alta voz es mayor que el consumo de de la reproducción con audífonos.

Al calcular la duración de la batería con respecto al consumo de corriente anterior y tomando la batería del Nexus S que es de 1500mAh, se obtiene la tabla 6.8.

*b*) **Comprobación de hipótesis** No se especificó hipótesis en el diseño.

*c*) **Conclusiones**

- La reproducción con altavoz en nivel de 100 % de volumen consume más del doble que la reproducción con audífonos en nivel de 70 % de volumen.
- La diferencia entre la canción de reggaetón Danza kuduro de Don Omar con volumen al 100 % y 75 % no normalizada es de 17,831mA.
- El control de volumen aplicado a la reproducción de la canción de reggaetón Danza kuduro de Don Omar disminuye utilizando los audífonos disminuye el consumo 21,887mA en comparación al alta voz.
- Controlar el volumen fijando nivele de volumen adecuados para el altavoz y los audífonos es una práctica que permite disminuir el consumo de energía.

## 6.3. ACELERÓMETRO

### 6.3.1. Prueba 1: Modificación del tiempo de retardo entre mediciones.

Modificación tiempo de retardo entre mediciones del sensor en el registerListener(). La descripción completa de esta prueba se encuentra en el anexo E tabla E.5.

#### Ejecución

*a*) **Caracterización del consumo inicial**: se utiliza la caracterización inicial del acelerómetro realizada en el capítulo 5 en donde se encontró el promedio de la interfaz de la aplicación "Acelerometo_1" (se utiliza la misma interfaz) obteniendo los valores de corriente y potencia de 98,985157mA y 0,396759W.

*b*) **Ejecución inicial**: Se realiza la medición del consumo de energía bajo el escenario descrito y corriendo la aplicación, como resultado se obtuvo los datos de la tabla 6.9.

| Tasa de eventos (ms) | Promedio corriente (mA) | Promedio corriente sin interfaz (mA) |
|---|---|---|
| 200 | 243,851 | 144,865 |
| 500 | 242,259 | 143,274 |
| 1000 | 237,952 | 138,967 |
| 5000 | 237,387 | 138,402 |

Tabla 6.9: Promedios de corriente acelerómetro prueba 1.

*c*) **Ejecución secundaria**: no se realiza una ejecución secundaria ya que esta prueba no se definen parámetros variables.

### Análisis de mediciones

*a*) **Análisis comparativo:** Al analizar los gráficos que generan con los diferentes tiempos de retardo se puede observar que los consumos son muy similares al que se presenta en la caracterización del consumo inicial, presentando una cantidad muy alta de picos. Los gráficos de las 4 tasas se presentan el la figura 6.8.

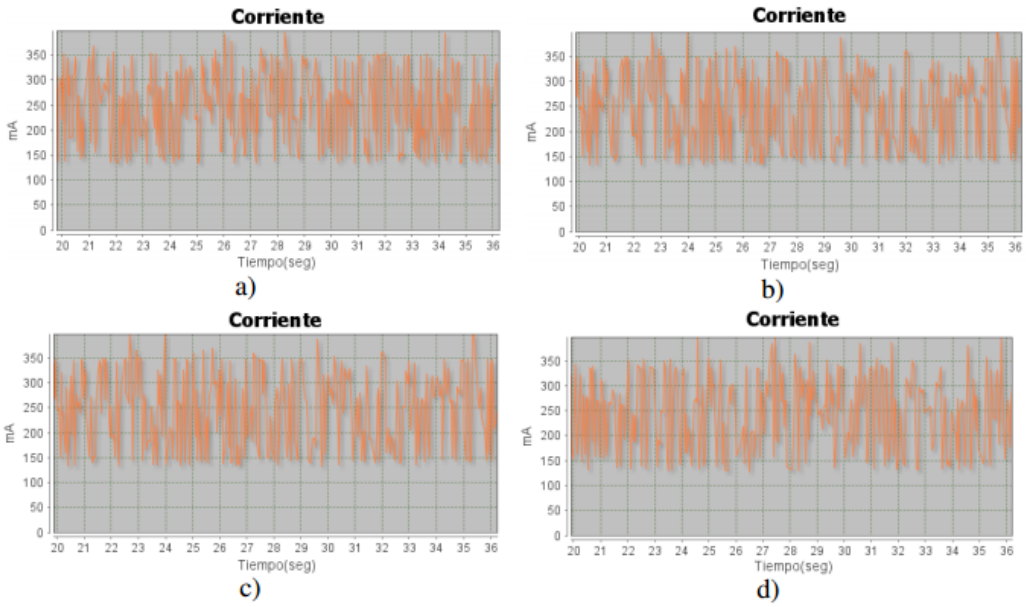


Figura 6.8: Medición acelerómetro prueba 1 - variación del Modificación del tiempo de retardo a) 200ms b)500ms c)1000ms d) 5000ms.

Como se observa en la figura 6.8 los picos son generados a una tasa mayor a la tasa de eventos indicada, es por esto que se observa la aplicación en el dispositivo teniendo como resultado que la tasa de eventos indicada en el *registerListener()* no era la misma con la cual se desplegaban los valores en la interfaz, lo que determina que esta tasa de eventos es una sugerencia que se le hace al *SensorEventListener* ya que no se efectúan las actualizaciones en el tiempo establecido. No se puede apreciar una diferencia gráfica entre las gráficas del consumo a diferentes tasas de eventos, es por esto que se analiza directamente los promedios utilizando el gráfico de barras de la figura 6.9.

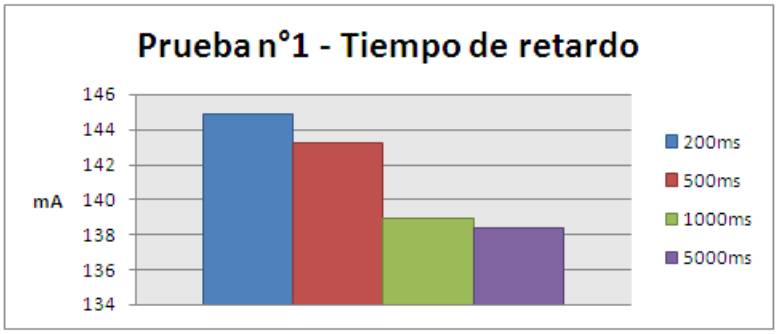


Figura 6.9: Promedio consumo acelerómetro prueba 1 - Modificación del tiempo de retardo.

De la figura 6.9 se puede analizar lo siguiente:

- Al utilizar una mayor tiempo de retardo en el registerListener() se produce un menor consumo.
- La diferencia entre utilizar un tiempo de 200ms y una de 5000ms es de 6.4mA, correspondiente a un tiempo de descarga en el primer caso de 10,3544h y en el segundo en 10,8379h, si se utiliza la batería de 1500mAh del Nexus S. Esta diferencia corresponde a 29 minutos.
- Aunque las actualizaciones no se realizan al tiempo establecida, si se puede comprobar que entre mayor es la tasa menor es el consumo, por lo cual si disminuyen las actualizaciones pero no al tiempo que se ha establecido.

*b*) **Comprobación de hipótesis**: Se comprueba que se presenta una disminución del consumo debido a que se presentan menos eventos por segundo.

*c*) **Conclusiones**:

- La diferencia entre utilizar un tiempo de 200ms y una de 5000ms es de 6.4mA, correspondiente a tener 29 minutos más de batería en el dispositivo Nexus S.
- La tiempo de retardo indicado en el registerListener() es solo una sugerencia para el sistema ya que este toma un tiempo menor al que se le asigna.

### 6.3.2. Prueba 2: Consumo interfaz sin referencia al SENSOR_SERVICE.

Se realiza la medición del consumo de la interfaz desactivando la referencia del SENSOR_SERVICE en el onCreate(). La descripción completa de esta prueba se encuentra en el anexo E tabla E.6.

#### Ejecución

*a*) Caracterización del consumo inicial: se realiza una nueva medición del consumo de la interfaz, obteniendo una promedio de corriente y potencia de 114,7514847mA y 0,440070396W respectivamente.

*b*) Ejecución inicial: Se realiza la medición del consumo de energía de la interfaz bajo el escenario descrito. corriendo la aplicación y desactivando la referencia al sensor, como resultado se obtuvo los datos corriente y potencia de 111,5348651mA y 0,425164473W respectivamente.

*c*) Ejecución secundaria: no se realiza una ejecución secundaria ya que esta prueba no se definen parámetros variables.

#### Análisis de mediciones

*a*) **Análisis comparativo:** Al comparar los datos obtenidos en la nueva medición de la caracterización inicial y el consumo de la interfaz desactivando la referencia se puede tiene una diferencia aproximada de 3mA. La gráfica del consumo de la nueva interfaz se puede apreciar en la figura 6.10.

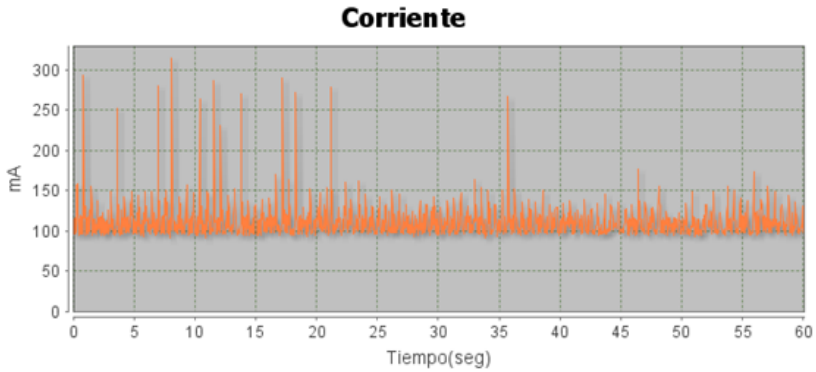


Figura 6.10: Medición consumo acelerómetro prueba 2.

*b*) **Comprobación de hipótesis** Se comprueba que obtener una referencia del SENSOR_ SERVICE produce un consumo aun sin registrar el *SensorEventListener*.

*c*) **Conclusiones**

- La referencia SENSOR_ SERVICE produce un consumo aproximado de 3mA.

### 6.3.3. Prueba 3: Modificación de tiempos de despliegue de datos en pantalla

Se condiciona el despliegue de los datos en pantalla a un tiempo determinado. La descripción completa de esta prueba se encuentra en el anexo E tabla E.7.

#### Ejecución

*a*) **Caracterización del consumo inicial:** se utiliza el promedio del consumo del Acelerómetro realizadoen la prueba 2 en donde se encontró el promedio de la interfaz de la aplicación "Acelerometo_1", los valores de corriente y potencia son111,5348651mA y 0,425164473W.

*b*) **Ejecución inicial:** Se realiza la medición del consumo de energía bajo el escenario descrito y corriendo la aplicación, como resultado se obtuvo los datos de la tabla 6.10.

| Tiempo de despliegue (ms) | Promedio corriente (mA) | Promedio corriente sin interfaz (mA) |
|---|---|---|
| 250 | 135,765 | 24,230 |
| 500 | 124,983 | 13,448 |
| 1000 | 119,668 | 8,133 |
| Sin despliegue | 117,216 | 5,681 |

Tabla 6.10: Promedio de corriente acelerómetro prueba 3 - Modificacion tiempos de despliegue de datos en pantalla.

*c*) **Ejecución secundaria**: En base a la ejecución inicial se decide caracterizar de nuevo el consumo del algoritmo 1 (referencia del capitulo 5) y posteriormente compararlo con el de la ejecución inicial. Los promedios obtenidos se pueden apreciar en la tabla 6.11.

| Tiempo de despliegue (ms) | Promedio corriente (mA) | Promedio corriente sin interfaz (mA) |
|---|---|---|
| 250 | 133,156 | 21,621 |
| 1000 | 118,179 | 6,644 |
| Sin despliegue | 114,051 | 2,516 |

Tabla 6.11: Promedio de corriente acelerómetro prueba 3 - Modificación tiempos de despliegue de datos en pantalla en algoritmo 1.

### Análisis de mediciones

*a*) **Análisis comparativo:** Como se observan los 4 gráficos resultante de la ejecución de la figura 6.11, el consumo del dispositivo presenta una disminución gradual al aumentar el tiempo de despliegue de los datos en pantalla, notando que los picos que se presentan en la figura 6.8 a) disminuyen considerablemente.

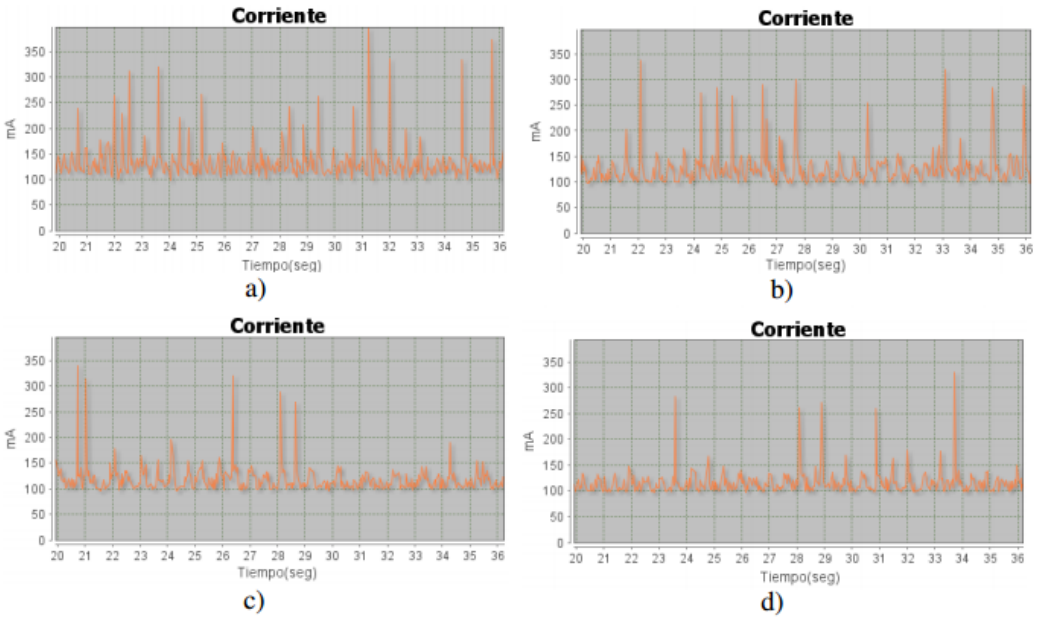


Figura 6.11: Medición acelerómetro prueba 3 - Variación de tiempo de despliegue en pantalla en algoritmo 1. a) 200ms b)500ms c)1000ms d)sin mostrar datos.

La ejecución en la cual no se despliegan datos en pantalla (figura 6.11 *d)*), nos indica que los grandes picos obtenidos en la caracterización inicial y en la figura 6.8 eran generados por la alta velocidad del cambio de valores en la pantalla, siendo así el consumo neto del acelerómetro el presentado en la figura 6.11 d).

En la gráfica de barras de la figura 6.14 se observan el promedio de consumo de las 4 ejecuciones de esta prueba y adicionalmente el consumo generado en la prueba 1 donde se tiene una tasa de eventos de 200ms (figura 6.14 a)).

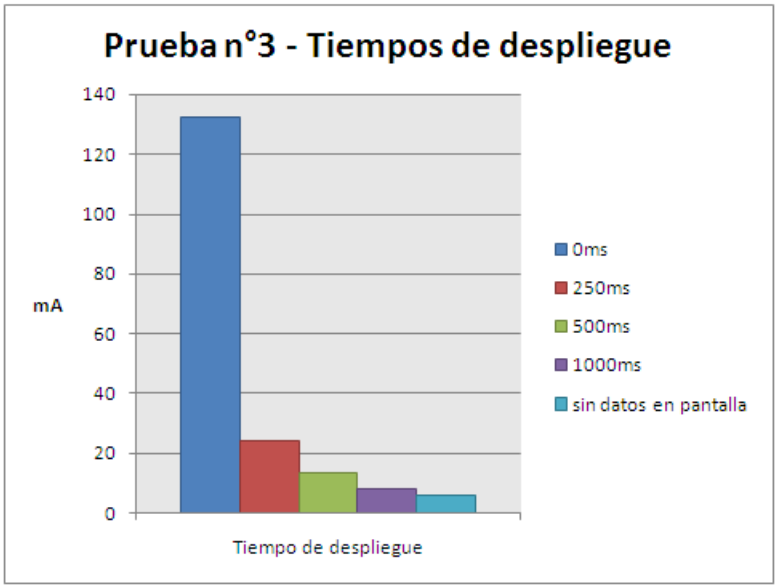


Figura 6.12: Promedio de corriente acelerómetro prueba 3- Variación de tiempo de despliegue en pantalla en algoritmo 1.

Del grafico presentado en la figura 6.14 se puede analizar lo siguiente:

- No tener un control del tiempo para desplegar los datos puede generar un gran consumo que no es atribuido al sensor del acelerómetro.
- Disminuir el tiempo de despliegue de los datos en pantalla presenta una disminución considerable del consumo.
- El tiempo de despliegue de 1000ms es una buena tasa para mostrar los datos de forma que los usuarios puedan percibir bien los cambios, produciendo un consumo muy aproximado al del acelerómetro sin desplegar datos (aproximadamente 2.5mA de más).

Al realizar la ejecución secundaria se encuentra que los picos que se generan son muy parecidos a los presentes en la ejecución inicial. En la figura 6.13 se puede observar la comparación entre la ejecución inicial y la secundaria sin desplegar datos en pantalla.

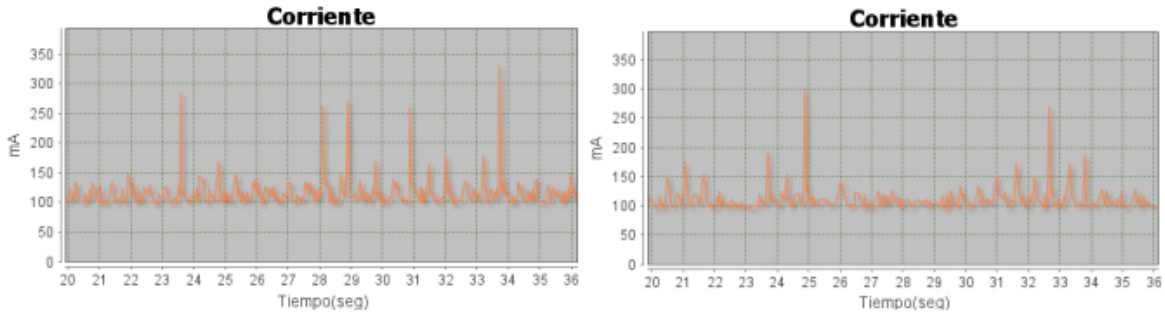


Figura 6.13: Acelerómetro prueba 3 - Ejecución inicial y secundaria.

Se realiza la comparación entre los 2 algoritmos resultando el gráfico de barras de la figura 6.14.

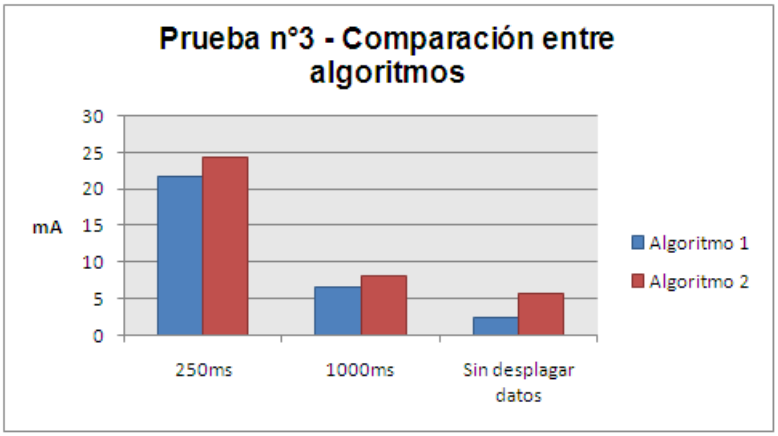


Figura 6.14: Promedio de corriente acelerómetro prueba 3 - Algoritmos 1 y 2.

Del gráfico presentado en la figura 6.14 se puede comprobar que el algoritmo 1 presenta menor consumo en las 3 comparaciones.

3. **Comprobación de hipótesis** Se comprueba que muchos de los picos generados son debidos a los datos mostrados a gran velocidad en la interfaz. Además se comprueba que el algoritmo 1 sigue presentando menor consumo que el algoritmo 2.

4. **Conclusiones**

   - Desplegar datos en pantalla sin tener un tiempo para que estos sean desplegados (0ms) produce un aumento de consumo con respecto a la aplicación sin datos en pantalla de 126,629mA, que equivale 11 horas con 50 minutos de uso del dispositivo Nexus S.
   - El acelerómetro genera un consumo neto de 5.681mA sin datos en pantalla, lo que equivale a 11 días de duración en una batería de 1500mAh,
   - 1000ms como tiempo de despliegue es una tasa aceptable para usar el acelerómetro sin tener un consumo elevado.

## 6.4. BRUJULA

### 6.4.1. Prueba 1: Consumo interfaz sin referencia al SENSOR_SERVICE

Se realiza la medición del consumo de la interfaz desactivando la referencia del SENSOR_SERVICE en el onCreate(). La descripción completa de esta prueba se encuentra en el anexo E tabla E.8.

#### Ejecución

1. **Caracterización del consumo inicial**: se realiza una nueva medición del consumo de la interfaz, obteniendo una promedio de corriente y potencia de 48,55661801mA y 0,194997167W respectivamente.

2. **Ejecución inicial**: Se realiza la medición del consumo de energía de la interfaz bajo el escenario descrito. corriendo la aplicación y desactivando la referencia al sensor, como resultado se obtuvo los datos corriente y potencia de 44,94570238mA y 0,180030339W respectivamente.

3. **Ejecución secundaria**: no se realiza una ejecución secundaria ya que esta prueba no se definen parámetros variables.

**Análisis de mediciones**

1. **Análisis comparativo:** Al comparar los datos obtenidos en la nueva medición de la caracterización inicial y el consumo de la interfaz desactivando la referencia se puede tiene una diferencia aproximada de 3,5mA. La gráfica del consumo de la interfaz se puede apreciar en la figura 6.15.

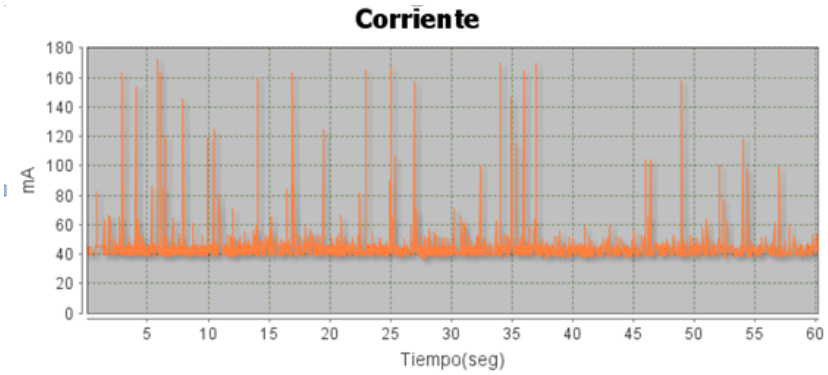


Figura 6.15: Medición consumo brújula prueba 1.

2. **Comprobación de hipótesis**

   Se comprueba que obtener una referencia del SENSOR_SERVICE produce un consumo aun sin registrar el *SensorEventListener*.

3. **Conclusiones**

   - La referencia SENSOR_SERVICE produce un consumo aproximado de 3.5mA.

### 6.4.2. Prueba 2: Modificación de tiempos de despliegue de datos en pantalla

Se condiciona el despliegue de los datos en pantalla a un tiempo determinado y sin desplegar datos en pantalla. La descripción completa de esta prueba se encuentra en el anexo E tabla E.9.

**Ejecución**

1. **Caracterización del consumo inicial**: se utiliza el promedio del consumo del brújula realizado en la prueba 2 en donde se encontró el promedio de la interfaz de la aplicación “Brujula_ inicial”, los valores de corriente y potencia son 44,94570238mA y 0,180030339W.

| Tiempo de despliegue (ms) | Promedio corriente (mA) | Promedio corriente sin interfaz (mA) |
|---|---|---|
| 250 | 61,557 | 16,611 |
| 500 | 59,516 | 14,570 |
| 1000 | 57,689 | 12,743 |
| Sin despliegue | 56,650 | 11,705 |

Tabla 6.12: Promedio de corriente brújula prueba 2 - Modificación tiempos de despliegue de datos en pantalla.

2. **Ejecución inicial**: Se realiza la medición del consumo de energía bajo el escenario descrito y corriendo la aplicación con el algoritmo 3, como resultado se obtuvo los datos de la tabla 6.12.

3. **Ejecución secundaria**: En base a la ejecución inicial se decide caracterizar de nuevo el consumo del algoritmo 2(referencia del capitulo 5) y posteriormente compararlo con el consumo neto de la ejecución inicial, es decir sin desplegar datos en pantalla. Los promedios obtenidos se pueden apreciar en la tabla 6.13.

| Tiempo de despliegue (ms) | Promedio corriente (mA) | Promedio corriente sin interfaz (mA) |
|---|---|---|
| Sin despliegue | 56,271 | 11,325 |

Tabla 6.13: Promedio de corriente brújula prueba 2 - Modificación tiempos de despliegue de datos en pantalla. Algoritmo 2.

### Análisis de mediciones

1. **Análisis comparativo:** Al utilizar el algoritmo 3 se hace uso de 2 sensores donde uno de estos es el acelerómetro y el otro el magnetómetro, como se pudo observar en la prueba 1 del acelerómetro se encontró que los grandes picos eran atribuidos a la gran velocidad con la cual se despliegan los datos en pantalla, es por esto que este efecto también ocurre en la brújula al utilizar la misma clase *SensorManager*.

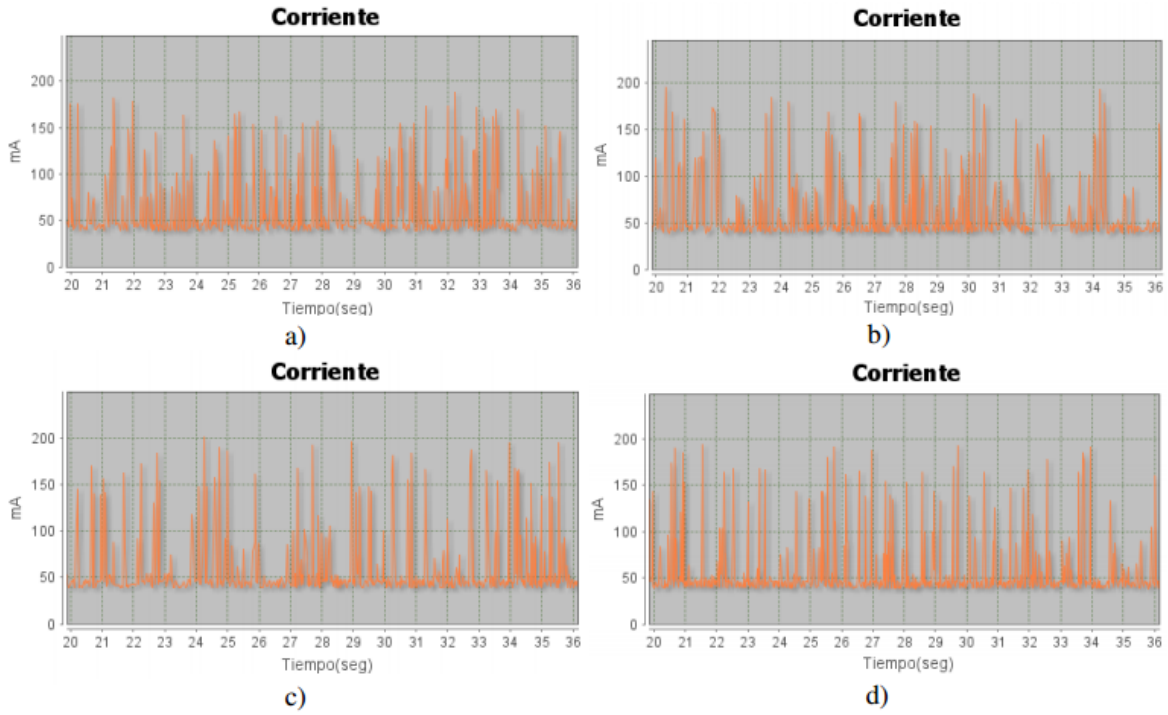


Figura 6.16: Medición brújula prueba 2 - variación de tiempo de despliegue en pantalla. a) 200ms b)500ms c)1000ms d)sin mostrar datos.

Como se observan los 4 gráficos resultantes de la ejecución de la figura 6.16, el consumo del dispositivo

presenta una disminución gradual al aumentar el tiempo de despliegue de los datos en pantalla, notando que los picos que se presentan en la figura 5.23 a) disminuyen considerablemente.

La ejecución en la cual no se despliegan datos en pantalla correspondiente a la figura 6.16 d), indica que los grandes picos obtenidos en la caracterización inicial del capitulo 5 eran generados por la alta velocidad del cambio de valores en la pantalla al igual que en el acelerómetro, siendo así el consumo neto de la brújula el presentado en la figura 6.16 d).

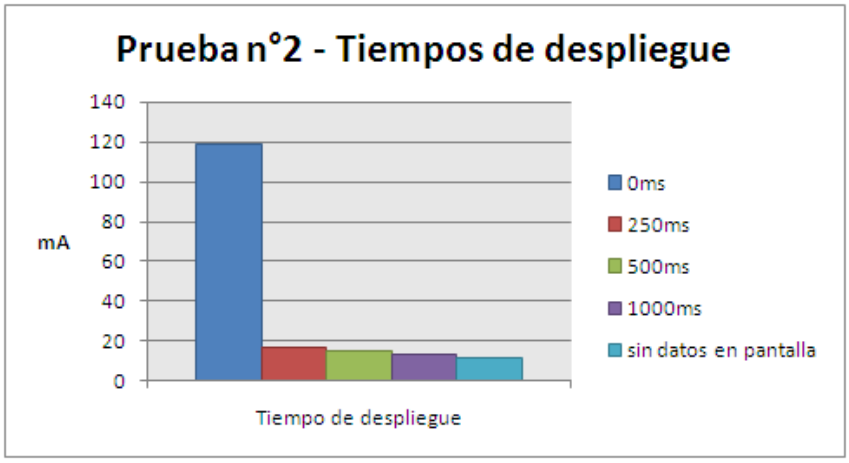


Figura 6.17: Promedio de corriente brújula prueba 2.

En la gráfica de barras de la figura 6.17 se observan el promedio de consumo de las 4 ejecuciones de esta prueba y adicionalmente el consumo generado por la brújula mostrado en la sección (5.2.4) en la figura 5.27 c) de la caracterización inicial.

Del gráfico presentado en la figura 6.17 se puede analizar lo siguiente:

- No tener un control del tiempo para desplegar los datos puede generar un gran consumo que no es atribuido al ninguno de los dos sensores utilizados en al brújula.
- Disminuir el tiempo de despliegue de los datos en pantalla presenta una disminución considerable del consumo.
- El tiempo de despliegue de 1000ms es una buena tasa para mostrar los datos de forma que los usuarios puedan percibir bien los cambios, produciendo un consumo muy aproximado al de la brújula sin desplegar datos (aproximadamente 1mA de más).

Al realizar la ejecución secundaria se encuentra que los picos que se generan son muy parecidos a los presentes en la ejecución inicial. En la figura 6.18 se puede observar la comparación entre la ejecución inicial y la secundaria utilizando el algoritmo 2 sin desplegar datos en pantalla.

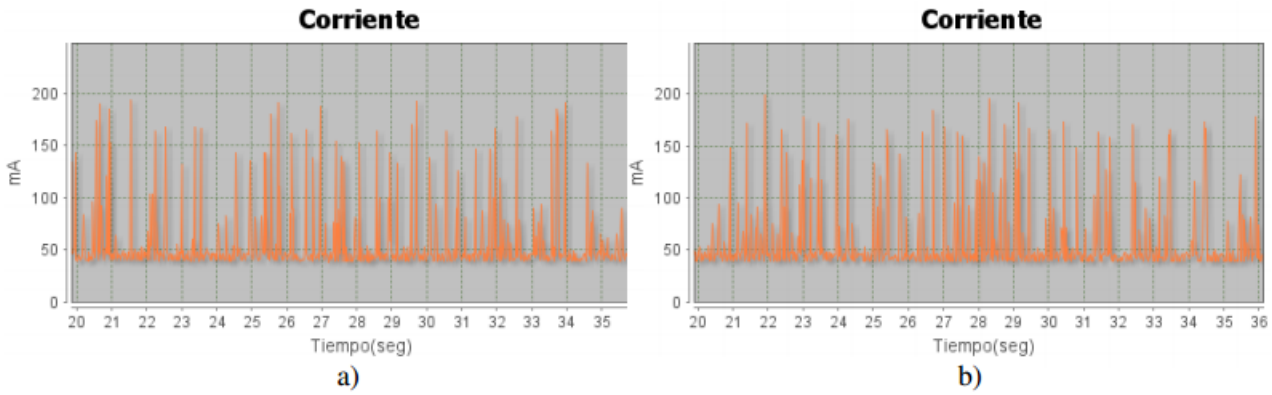


Figura 6.18: Brújula prueba 2 - ejecución inicial y secundaria.

Se realiza la comparación entre los 2 algoritmos con los promedios de consumo sin desplegar datos en pantalla, resultando el gráfico de barras de la figura 6.19.

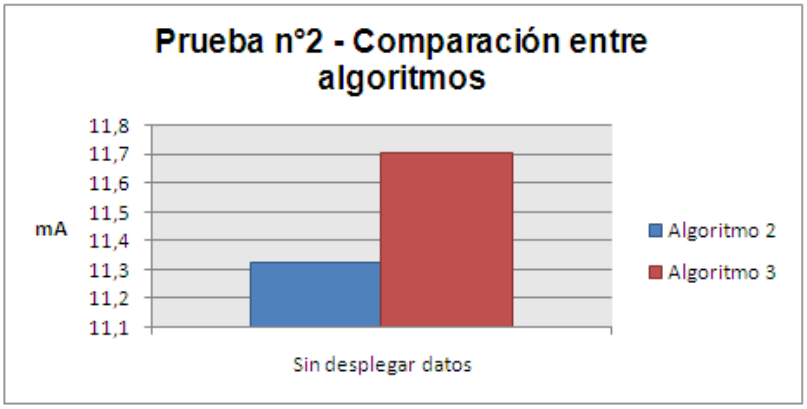


Figura 6.19: Promedio de corriente brújula prueba 2 - algoritmos 2 y 3.

Del grafico presentado en la figura 6.19 se puede comprobar que el algoritmo 2 presenta menor consumo que el algoritmo 3.

**Comprobación de hipótesis** Se comprueba que muchos de los picos generados son debidos a los datos mostrados a gran velocidad en la interfaz. Además se comprueba que el algoritmo 2 sigue presentando menor consumo que el algoritmo 3.

**Conclusiones**

- Desplegar datos en pantalla sin tener un tiempo para que estos sean desplegados (0ms) produce un aumento de consumo con respecto a la aplicación sin datos en pantalla de 107,56mA, que equivale 13 horas con 56 minutos de uso del dispositivo Galaxy Ace.
- La brújula genera un consumo neto de 11,705mA sin datos en pantalla, lo que equivale a 5 días, 8 horas y 10minutos de duración en una batería de 1500mAh.
- 1000ms como tiempo de despliegue es una tasa aceptable para usar la brújula sin tener un consumo elevado.

## 6.5. TECLADO

### 6.5.1. Prueba 1: Teclado predictivo

Utilizar teclado predictivo permite escribir mas palabras presionando menos veces la pantalla, esto genera un menor consumo. Se escribe la frase: “Prueba sobre el teclado predictivo”. La descripción completa de esta prueba se encuentra en el anexo E tabla E.10.

#### Ejecución

1. **Caracterización del consumo inicial**: se utiliza la caracterización inicial del teclado realizada en el capítulo 5 en la cual se promedió el consumo de los picos generados por las letras y su periodo de tiempo. Adicionalmente se realiza la medición del consumo de la interfaz de la aplicación “EscenarioTeclado” utilizando el teclado predictivo. Obteniendo como resultado Una corriente de 119,7752689mA.

2. **Ejecución inicial**:Se realiza la medición del consumo de energía bajo el escenario descrito y corriendo la aplicación de prueba sobre el teclado QWERTY normal y QWERTY predictivo.

3. **Ejecución secundaria**: no se realiza una ejecución secundaria ya que esta prueba no se definen parámetros variables.

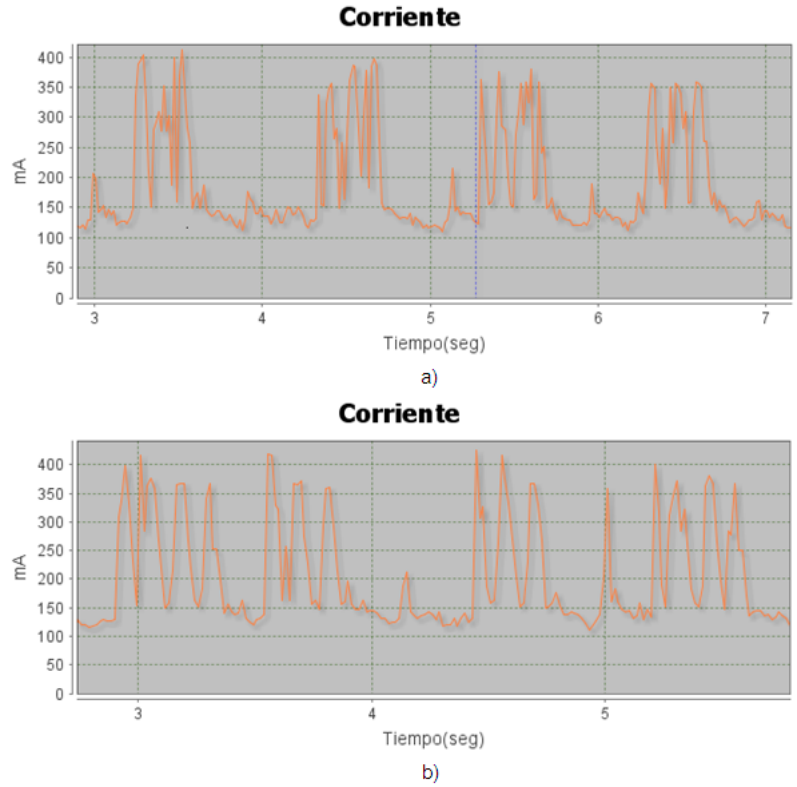


Figura 6.20: Pulsos teclado QWERTY. a) no predictivo b) predictivo.

**Análisis de mediciones**

1. **Análisis comparativo:**

   Para esta prueba se escribió la frase “prueba sobre el teclado predictivo” en el teclado QWERTY en modo predictivo y en el teclado QWERTY normal, presentando como resultado 25 y 34 pulsos respectivamente, los cuales corresponden al número de veces que fue necesario presionar la pantalla sobre letras y espacios. Cabe aclarar que el consumo de la interfaz en modo predictivo fue de 119,7752689mA y en modo no predictivo es de 116,407343mA, en donde se puede observar que su diferencia es aproximada de 3mA, el cual es un valor pequeño que no generaría un diferencia en el consumo significativa Es por esto que para el consumo general la diferencia entre los consumos de las interfaces no se relevante para el análisis. Al analizar los pulsos generados en los dos casos,los cuales se pueden observar en la figura 6.20,se aprecia que los picos presentan periodos similares de tiempo y un nivel de corriente entre 150mA y 400mA.

   Al promediar los consumos de los pulsos en los periodos de tiempo se encuentra que estos oscilan entre 292,060mA y 253,898mA con interfaz, y 172,285mA y 134,122mA sin interfaz, aproximadamente.En la tabla 6.14 se observan los promedios del consumo sin interfaz y los tiempos de duración de los pulsos.

| Teclado | | Pulso 1 | Pulso 2 | Pulso 3 | Pulso 4 | Promedio |
|---|---|---|---|---|---|---|
| QWERTY | Promedio corriente sin interfaz (mA) | 163,192 | 162,300 | 155,656 | 158,409 | 159,889 |
| | Tiempo (s) | 0,437 | 0,358 | 0,406 | 0,312 | 0,378 |
| QWERTY predictivo | Promedio corriente sin interfaz (mA) | 172,285 | 159,045 | 134,122 | 148,248 | 153,425 |
| | Tiempo (s) | 0,359 | 0,39 | 0,405 | 0,406 | 0,390 |

Tabla 6.14: Promedios de pulsos teclados predictivo y no predictivo.

Al analizar los valores de los consumos de los pulsos en la tabla 6.14 se puede percibir que el consumo que se presenta es relativo, debido a que los pulsos no siempre son iguales por causa de diversos parámetros que producen variaciones en el consumo, como son: el ancho del dedo del usuario y el fragmento de la superficie del dedo que este utilice para hacer la presión; sin embargo al promediar los valores de los pulsos en cada caso se encuentra que son valores muy similares, por tanto se decide analizar el consumo del teclado QWERTY, en los 2 modos, en función del número de pulsos que se generen escribiendo las frases, palabras o letras deseadas, lo que significaría que entre más pulsos sean creados más consumo se genera. El tiempo utilizado en escribir la frase sin contar las pausas se calcula con el promedio de tiempo que se encontró por letra en la tabla 5.444 multiplicado por el número de pulsos. En la tabla 6.15 se presentan los tiempos utilizados en los dos tipos de teclado QWERTY.

| Teclado | Pulsos | Promedio tiempo por letra (s) | Tiempo de escritura (s) |
|---|---|---|---|
| QWERTY | 34 | 0,378 | 12,852 |
| QWERTY predictivo | 25 | 0,390 | 9,75 |

Tabla 6.15: Numero de pulsos y promedio de tiempo.

Finalmente se puede afirmar que el teclado QWERTY predictivo consume energía durante un menor tiempo comparado con el QWERTY normal. Adicionalmente, se observa que el teclado predictivo

mejora sus predicciones a medida que el usuario escribe en él, ya que este aprende el modo de escritura del usuario lo que permite que las predicciones sean cada vez más rápidas y exactas. Aunque este teclado utiliza un mayor procesamiento lógico que produce un consumo este es compensado con el hecho de disminuir los picos.

2. **Comprobación de hipótesis** Se comprobó la hipótesis: al utilizar el teclado QWERTY predictivo se generarían menos pulsos que en el QWERTY sin modo predictivo, lo que significa que se genera menos consumo. La hipótesis se comprobó.

3. **Conclusiónes**

   - El consumo del teclado depende del número de pulsos que se generen por presionar la pantalla y la duración de estos.
   - Los pulsos generados por el teclado QWERTY predictivo y no predictivo consumen en promedio lo mismo.
   - Con el teclado QWERTY predictivo se usa menos tiempo para escribir una palabra que el en teclado QWERTY no predictivo.
   - El teclado QWERTY predictivo consume menos después de un tiempo de uso ya que aprende los patrones de palabras usadas por los usuarios haciendo sus predicciones cada vez mas precisas.
   - El consumo del teclado varía dependiendo del modo de uso que cada usuario pueda presentar.

### 6.5.2. Prueba 2: Teclados más utilizados

Se utiliza el teclado SwiftKey 3, el cual es descargado del google Play y el Swype versión beta(descargado de beta.swype.com). Se realiza la comparación del consumo de estos 2 teclados con el predictivo del dispositivo. Se utiliza la misma frase de la prueba 1. En la figura 6.21 se aprecia las interfaces de los teclados. La descripción completa de esta prueba se encuentra en el anexo E tabla E.11.


Figura 6.21: Teclados a) SwiftKey 3. b) Swype versión beta.

**Ejecución**

1. **Caracterización del consumo inicial**: se toma la prueba 1 del teclado como base para la comparación, además se mide el consumo de las interfaces de los teclados SwiftKey 3 y Swype beta, resultando 120,92981mA y 118,74316mA respectivamente.

2. **Ejecución inicial**:Se realiza la medición del consumo de energía bajo el escenario descrito sobre el teclado SwiftKey 3, obteniendo como resultado la gráfica de consumo de la figura 6.22.

3. **Ejecución secundaria**: Se repite la medición de la ejecución utilizando el teclado por el Swype beta.Obteniendo como resultado la gráfica de consumo de la figura 6.23.

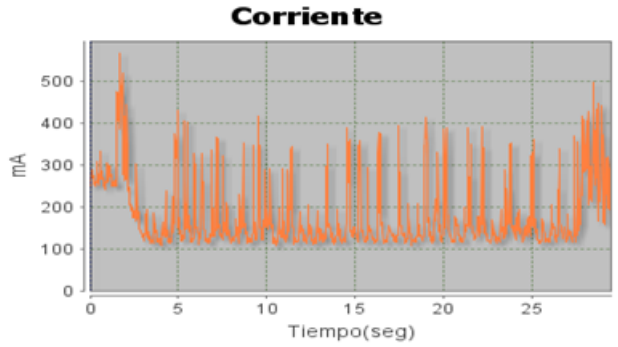


Figura 6.22: Medición consumo SwiftKey 3.

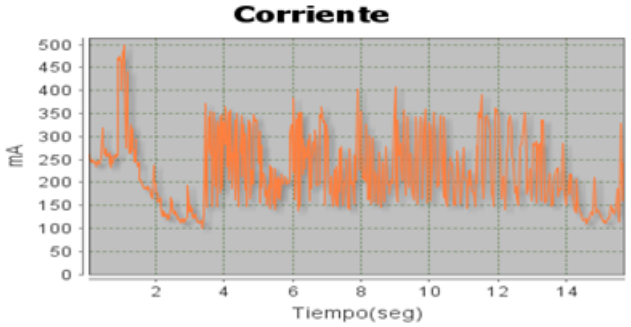


Figura 6.23: Medición consumo Swype versión beta.

**Análisis de mediciones:**

1. **Análisis comparativo**

   *a*) **SwiftKey 3** Para esta prueba se encontró inicialmente una diferencia notoria en las gráficas del consumo con respecto al teclado QWERTY predictivo, notando que en el teclado SwiftKey 3 los pulsos de consumo presentan a simple vista un periodo y un nivel de corriente un poco menor, como se puede apreciar en la figura 6.24.

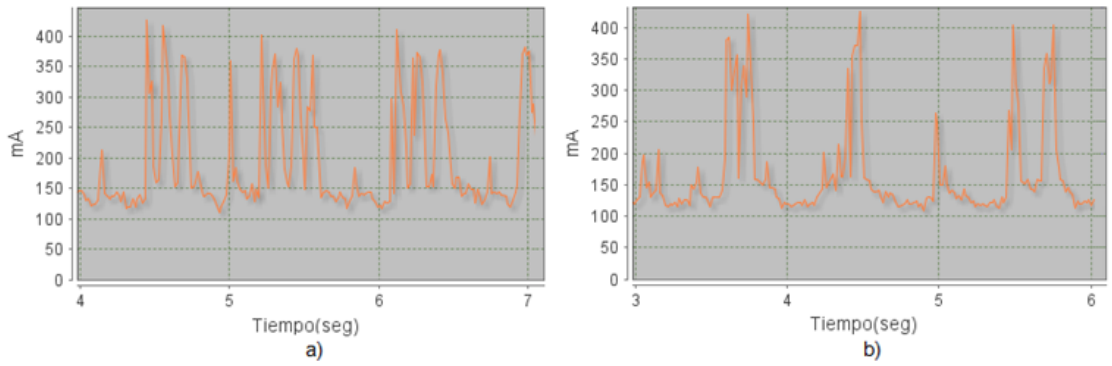


Figura 6.24: Pulsos teclados a)QWERTY predictivo. b)SwiftKey 3.

Para comprobar lo anterior se calcula el promedio de consumo de los pulsos y su tiempo de duración, los valores están consignados en la tabla 6.16.

| Teclado | | Pulso 1 | Pulso 2 | Pulso 3 | Pulso 4 | Promedio |
|---|---|---|---|---|---|---|
| QWERTY predictivo | Promedio corriente sin interfaz (mA) | 172,285 | 159,045 | 134,122 | 148,248 | 153,425 |
| | Tiempo (s) | 0,359 | 0,39 | 0,405 | 0,406 | 0,390 |
| SwiftKey 3 | Promedio corriente sin interfaz (mA) | 154,453 | 93,216 | 111,065 | 98,074 | 114,202 |
| | Tiempo (s) | 0,203 | 0,218 | 0,280 | 0,267 | 0,242 |

Tabla 6.16: Promedios de pulsos teclado QWERTY predictivo y SwiftKey 3.

Finalmente el teclado SwiftKey 3 produce un consumo menor al teclado QWERTY predictivo.

*b*) **Swype**: El teclado Swype contiene un método de escritura en el cual se desliza el dedo en la pantalla sobre las letras de la palabra a escribir, esto hace que no se presenten picos por cada letra sino muchos picos seguidos por palabra como se puede observar en la figura 6.25.

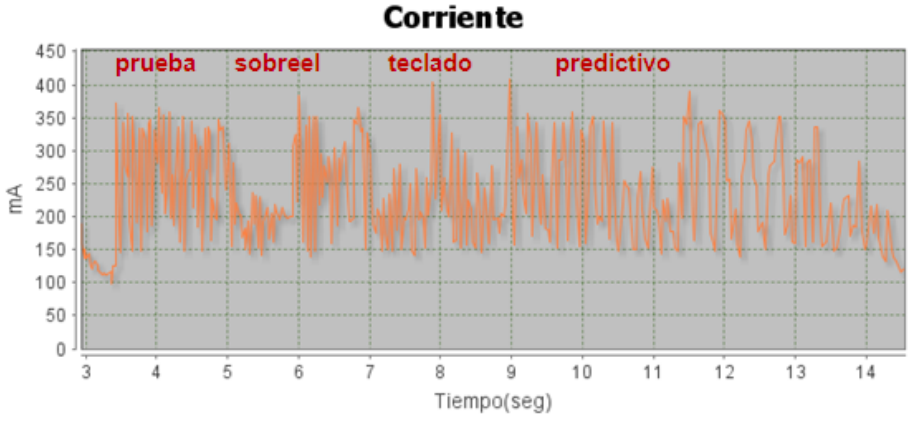


Figura 6.25: Medición consumos teclado swype.

Los valores del promedio del consumo por palabras se puede observar la tabla 6.17 y el tiempo utilizado para estas en la tabla 6.18.

| Palabra | prueba | sobre | el | teclado | predictivo | Promedio |
|---|---|---|---|---|---|---|
| Promedio corriente (mA) | 260,931 | 261,819 | 241,39 | 253,731 | 250,287 | **253,632** |
| Promedio corriente sin interfaz (mA) | 142,188 | 143,076 | 122,647 | 134,988 | 131,543 | **134,889** |

Tabla 6.17: Promedios corriente por palabras teclado swype.

| Palabra | prueba | sobre | el | teclado | predictivo | Total |
|---|---|---|---|---|---|---|
| Tiempo por palabra (s) | 1,804 | 1,17 | 0,562 | 1,701 | 2,169 | 7,406 |

Tabla 6.18: Promedios tiempos palabras teclado swype.

Para analizar este teclado se tiene en cuenta el promedio de consumo que se presenta por escribir toda la frase y el tiempo que se utilizó o presionó la pantalla al escribir. Por tanto al tener esto, se obtiene la tabla 6.19 en donde se presentan los promedios generales de consumo y el tiempo estimado que se utiliza para escribir la frase sin contar las pausas por teclado, adicionalmente se presenta el consumo que tienen las interfaces de cada uno.

| Teclado | Promedio interfaz (mA) | Promedio corriente sin interfaz (mA) | Tiempo de escritura (s) |
|---|---|---|---|
| QWERTY | 116,407 | 159,889 | 12,852 |
| QWERTY predictivo | 119,775 | 153,425 | 9,75 |
| SwiftKey 3 | 120,929 | 114,202 | 4,598 |
| Swype beta | 118,743 | 134,889 | 7,406 |

Tabla 6.19: Comparación consumos teclados.

Como se puede observar de la tabla 6.19 el teclado SwiftKey 3 posee el menor promedio de consumo y el menor tiempo de escritura, el teclado Swype posee un consumo un poco mayor que el SwiftKey pero menor que el QWERTY predictivo, es por eso que los dos teclados analizados cuentan con un menor consumo que el teclado que trae el dispositivo por defecto. Con respecto a las interfaces se nota que el consumo entre estas es muy parecido teniendo una variación de máxima de 4mA, por lo tanto no es una variable que afecte significativamente los resultados anteriores.

2. **Comprobación de hipótesis**

- Se comprobó la hipótesis: Al ser SwiftKey 3 un teclado con un uso muy grande en los usuarios de android, deben presentar un menor consumo comparado con el teclado predictivo que trae por defecto el dispositivo móvil.

- Se descarta la hipótesis: .*El Swype produce un mayor mayor consumo por su método de escritura, ya que se mantiene un mayor tiempo en contacto con la pantalla para escribir las palabras en un teclado"*, debido al que Swype no produce mayor consumo comparado con los teclados QWERTY y QWERTY predictivo.

3. **Conclusiónes**

   - El teclado SwiftKey 3 es el teclado que presenta menor consumo.
   - Las interfaces de los teclados poseen un valor de consumo muy parecido entre ellas.
   - El teclado Swype aunque tiene un consumo mayor que el SwiftKey 3 presenta un menor consumo que el QWERTY predictivo.
   - El teclado Swype no presenta un consumo por pulsos individuales sino por palabras.
   - El teclado SwiftKey 3 presenta pulsos por letra con menor amplitud y menor duración que en el QWERTY.
   - El teclado SwiftKey 3 presenta menos pulsos por letra que el teclado QWERTY.

## 6.6. PRACTICAS DE GESTIÓN DE ENERGÍA EN APLICACIONES

Como resultado de la ejecución y análisis de pruebas se presentan las prácticas de gestión de energía en programación y uso de aplicaciones sobre los periféricos GPS, sistema de audio (auricular y altavoz), acelerómetro, brújula y teclado.

### 6.6.1. Practicas GPS

1. **Selección de las variables minTime y minDistance:** Para no provocar consumo innecesario y obtener localización del dispositivo a una tasa adecuada, se seleccionan los parámetros minTime y minDistance que se asignan en el registro de actualizaciones requestLocationUpdates() de acuerdo a las siguientes recomendaciones:

   - Dado que el minTime es el tiempo que tomaría una actualización si se cumple una diferencia mínima de distancia equivalente al minDistance, se puede utilizar un minDistance pequeño en cualquier contexto y manejar las actualizaciones con el minTime.
   - minDistance debe ser mayor a cero para evitar actualizaciones cuando el dispositivo esta estático.
   - Determinar el contexto en el cual se utiliza la aplicación, permite asignar el minTime al pronosticar una velocidad de desplazamiento, entendiendo como contexto el modo de uso del dispositivo, es decir si se utiliza en un carro en movimiento, en carretera o en la ciudad, si se utiliza al caminar o se utiliza en un punto estático.

2. **Despliegue de la información:** Se debe hacer el debido procesamiento para mostrar cambios de posición en pantalla solo cuando se recorra un mínimo de distancia, ya que las actualizaciones de pantalla producen picos de corriente altos.

### 6.6.2. Practicas audio

1. **Control de volumen:** esta práctica consiste en detectar si se utilizan los auriculares del dispositivo y aparte de esto se asigna un nivel de volumen de 70 % utilizando la clase AudioManager. Como se muestra en el siguiente código.

```
am = (AudioManager) getSystemService(AUDIO_SERVICE);
int volNot = am.getStreamVolume(am.STREAM_MUSIC);
int vol = am.getStreamMaxVolume(am.STREAM_MUSIC);
 if (am.isWiredHeadsetOn()) {
   //cuando los auriculares está conectados
   am.setStreamVolume(am.STREAM_MUSIC, vol * 7 / 10, 0);
   //70% del volumen
 }
```

   Cuando se escucha música con el altavoz se suele utilizar un nivel de volumen alto, muchas veces del 100 %, al conectar los auriculares los usuarios debe variar el volumen lo que es molesto y muchas veces se termina utilizando un volumen mayor al necesario por tanto esta práctica propone manejar el control del volumen brindándole comodidad al usuario y evitando un consumo innecesario de energía.

2. **Normalizar nivel de volumen de los archivos:** A nivel de uso del dispositivo se encontró que normalizar el nivel de volumen de los archivos de audio permite que el consumo de reproducción de diferentes géneros musicales sea muy parecido, permitiendo evitar la subida del volumen entre un archivo y otro.

### 6.6.3. Practicas acelerómetro

1. **Modificación del tiempo de despliegue de datos en pantalla:** se condiciona el despliegue de los datos en pantalla a un tiempo adecuado para la visualización del usuario dependiendo del tipo de uso que se le dé al acelerómetro. Se puede apreciar en el siguiente código un ejemplo.

```
public void onSensorChanged(SensorEvent event) {

 synchronized (this) {
 //Tiempo de despliegue de 1 segundo.
  if(event.timestamp-current_time>1000000000){
   curX = event.values[0];
   curY = event.values[1];
   curZ = event.values[2];
   actividad.txtAccX.setText("Acelerometro X: "+ curX);
   actividad.txtAccY.setText("Acelerometro Y: "+ curY);
   actividad.txtAccZ.setText("Acelerometro Z: "+ curZ);
   current_time = event.timestamp;
  }
}
```

   El no utilizar esta práctica puede aumentar la corriente 126,629mA generando un consumo significativo.

2. **Selección del algoritmo:** como se menciono en el numeral 5.2.4 hay dos algoritmos que permiten manejar el acelerómetro donde el algoritmo 1 presento un menor consumo que el algoritmo 2 y una estabilidad mayor en reposo, siendo una buena opción y presentando un retardo un poco menor a 200ms. El algoritmo 2 puede ser utilizado en aplicaciones que necesiten una sensibilidad mayor y se necesite detectar el mínimo movimiento, siendo necesarios retardos como los especificados (SENSOR DELAY FASTEST= 0 ms, SENSOR DELAY GAME=20 ms, SENSOR DELAY UI= 67ms).

### 6.6.4. Practicas Brújula

1. Siendo la brújula una aplicación implementada a partir de sensores, se puede aplicar la práctica 1 encontrada para el acelerómetro.

2. **Selección del algoritmo:** La brújula presenta tres algoritmos como se menciona en la numeral 5.2.4, los 2 primeros utilizan el sensor de orientación y el tercero utiliza el acelerómetro y el magnetómetro. El algoritmo 1 produce un menor consumo y presenta una estabilidad mayor en reposo siendo una buena opción sin presentar una diferencia significativa en la precisión de la orientación. En caso escoger el algoritmo 2 o el algoritmo 3, estos no presenta una diferencia significativa en su consumo.

### 6.6.5. Practicas teclado

En el caso del teclado se encontraron las siguientes características sobre el uso del teclado:

- El consumo del teclado depende del numero de pulsos que se generen por presionar la pantalla y la duración de estos.
- Con los teclados predictivos se usa menos tiempo para escribir una palabra lo que disminuye el consumo.
- El teclado predictivo consume menos después de un tiempo de uso ya que aprende los patrones de palabras usadas por los usuarios haciendo sus predicciones cada vez mas precisas.

En la prueba 2 del teclado se concluyó que los teclados SwiftKey 3 y Swype beta producen un consumo menor que el teclado por defecto de sistema operativo, presentando pulsos de menor duración.

# Capítulo 7

# Conclusiones, contribuciones y trabajo futuro

Este capítulo recopila las conclusiones, contribuciones y trabajos futuros identificados. En la sección de conclusiones se retoman las presentadas al final de cada capítulo. En cuanto a contribuciones, se presentan las más relevantes a nivel de la arquitectura, herramientas de soporte y experiencias desarrolladas en ambientes reales de aprendizaje. Por último, se presentan las lecciones aprendidas, como conclusiones generales del trabajo desarrollado.

## 7.1. Conclusiones

### 7.1.1. Conclusiones del marco conceptual

Partiendo de los conceptos, tecnologías y trabajos relacionados que se han recopilado en el marco conceptual se puede concluir que:

- Se seleccionan para estudio los periféricos GPS, sistema de audio (auricular y altavoz), acelerómetro, brújula y teclado por su valor en múltiples campos de aplicación que estan tomando relevancia en el mercado de aplicaciones móviles.
- El presente estudio se realiza sobre el sistema operativo movil Android, el cual además de ser de código abierto es el sistema operativo más utilizado e el momento.
- La capacidad o carga eléctrica almacenada en la batería es una medida que permite calcular el tiempo de duración de la batería respecto a un consumo constante corriente.
- Las baterías de ion de litio (Li-ion) son las más utilizadas en los dispositivos móviles actuales, debido a su buena capacidad de almacenamiento, presenta una tasa de auto-descarga baja y un voltaje por célula alto.
- En los trabajos relacionados no se encuentra documentada una forma completa para realizar la evaluación del consumo de energía en los dispositivos móviles.
- El modelo hardware de medición de consumo de energía y las técnicas de análisis utilizadas por Rice et al. [26] servirán como referencia para la construcción e implementación de un sistema de medición acorde a los requerimientos de la investigación.
- La utilización de diferentes contextos en que puede utilizarse el dispositivo móvil estudiado por Nishihara et al. [4] nos servirá para la definición de algunos de los escenarios a utilizar en las pruebas.

- La utilización de lotes de programación (Batch-Scheduling) sobre aplicaciones recurrentes por Marina et al. [28], es un método interesante que se podrá implementar en el desarrollo de aplicaciones de prueba.

- Se encontró que los trabajos relacionados son pocos, factor que determina el posible carácter innovador de la investigación a desarrollar.

### 7.1.2. Conclusiones generales del modelo para la evaluación de consumo de energía para dispositivos móviles

De la creación del modelo para la evaluación del consumo de energía sobre dispositivo móviles y la aplicación de este en el desarrollo de este trabajo se puede concluir que:

- El contar con un modelo facilita y organiza la evaluación del consumo de energía permitiendo crear alternativas para la gestión de energía.

- Los criterios para la adquisición de medidas facilitan la implementación o clasificación de herramientas necesarias para poder realizar las mediciones.

- La utilización de métodos para la sincronización de eventos permiten identificar eventos a través del consumo en una aplicación y así facilitando su análisis.

### 7.1.3. Conclusiones generales de la herramienta para la medición de consumo de energía en dispositivos móviles

Del diseño e implementación de la herramienta para la medición de consumo de energía en dispositivos móviles se puede concluir que:

- Con base en los criterios para la adquisición de medidas descritos en el apartado 3.2 y algunas características de las herramientas existentes, se diseñó e implementó una herramienta para la medición de energía.

- La utilización de gráficas en tiempo real permite tener una noción del consumo del dispositivo y de los eventos presentes en este en cualquier momento, sirviendo de base para el análisis del consumo de cualquier tipo de uso del dispositivo.

- La utilización del ADC en un microcontrolador requiere de un método adicional que ayude a estabilizar las mediciones mejorando la precisión.

- La herramienta cuenta con una precisión que varia 4,88mV entre medidas, una frecuencia de medición de 66,667Hz, permite realizar graficación en tiempo real de gráficas de voltaje y corriente, y posee una persistencia de la información que permite guardar archivos .csv y .jpg.

### 7.1.4. Conclusiones generales de la planificación de pruebas

Se realiza la planificación de pruebas caracterizando el consumo de energía inicial obteniendo las siguientes conclusiones de cada periférico :

### GPS

- A pesar que se especifica en el registro de actualizaciones un tiempo de 1000ms, al observar la aplicación se percibe que las actualizaciones no presentan este periodo.
- Se generan picos de magnitud significativa cuando se activan las actualizaciones de localización, lo cual puede deberse a las actualizaciones como tal o al cambio de valores en la interfaz.

### Audio

- *MediaPlayer* es la clase más utilizada en aplicaciones para reproducción de Audio ya que soporta archivos de audio de cualquier tamaño y requiere cantidad de código con respeto a la clase *AudioTrack*.
- En la medición del consumo de la interfaz se puedo observar que se presentan picos inesperados atribuidos al procesamiento de la aplicación, es por esto que es difícil distinguir los picos generados por la reproducción del archivo.
- No es tan relevante estudiar los picos generados por la reproducción de canciones debido a que estas presentar diferentes patrones de consumo, por tanto se analiza el promedio de consumo.
- Reproducir un archivo de música con audífonos de forma interna o externo no produce una diferencia significativa en el consumo.
- Al reproducir un archivo de música sin audífonos el consumo de un archivo interno es mayor al consumo de un archivo externo a la aplicación, aproximadamente de 5mA.

### Acelerómetro

- Existen 2 algoritmos que permiten la obtención de datos del acelerómetro, aunque uno ha sido declarado obsoleto en el *Android Developers*.
- El acelerómetro al ser activado presenta picos significativos dado la constante variación del sensor, aun estando el dispositivo en reposo (sin movimiento), mostrando una variación rápida y constante de valores en la interfaz de la aplicación.
- El algoritmo 1 presenta una menor variación de datos en reposo, lo que significa que el algoritmo 2 presenta una mayor sensibilidad para detectar el cambio de valores en el acelerómetro, aun estando en reposo.
- El algoritmo 1 presenta un mayor consumo con movimiento que sin movimiento (la diferencia es de 20mA aproximadamente) y el algoritmo 2 no posee diferencia entre los 2 estados, por tanto genera más consumo que el algoritmo 1.
- No se percibe diferencia en la exactitud de valores entregados por el acelerómetro entre los dos algoritmos.

### Brújula

- Existen 3 algoritmos que permiten la obtención de datos de la brújula, aunque dos han sido declarado obsoleto según en el *Android Developers*.

- La brújula al ser activada presenta picos significativos dado la constante variación del sensor, aun estando el dispositivo en reposo (sin movimiento), mostrando una variación rápida y constante de valores en la interfaz de la aplicación.
- Los algoritmos 1 y 2 utilizan el sensor de orientación, en cambio el algoritmo 3 hace uso de 2 se sensores, el acelerómetro y el magnetómetro.
- Las mediciones del dispositivo estando estático y en movimiento no presentaron diferencias significativas del consumo en los 3 algoritmos.
- Los algoritmos 1 y 2 presentan un menor consumo que el algoritmo 3.

**Teclado**

- Existen muchas aplicaciones de teclado disponibles en el Google Play, lo que permite realizar un estudio adecuado del teclado con las aplicaciones más utilizadas.
- La interfaz del teclado presenta un mayor consumo al estar en posición horizontal que el vertical el dispositivo.
- Los pulsos generados por presionar las teclas presentan valores de consumo que varían, aunque presentan patrones similares, se nota que el consumo depende de factores como el área y tiempo de presión en la pantalla.
- El consumo generado depende del tiempo que dure el pulso de consumo, es decir, entre más rápido se escriba menor sería el ancho de este pulso.
- El consumo general de utilizar el teclado depende del numero de pulsos que se genere, es decir el número de veces que se presiona sobre la pantalla.

### 7.1.5. Conclusiones generales de la ejecución y análisis de pruebas

Se realiza la ejecución y análisis de pruebas de consumo de energía obteniendo las siguientes conclusiones de cada periférico :

**GPS**

- Aunque aumentar los parámetros minTime y minDistance disminuyen el consumo del GPS esta disminución no presenta un gran impacto sobre el consumo del dispositivo.
- El minTime es el tiempo que tomaría una actualización si se cumple una diferencia mínima de distancia equivalente al minDistance.
- Aun cuando el número de picos disminuye, el GPS presenta un valor de corriente mínima que se encuentra alrededor de 130mA con interfaz (30mA sin interfaz) aproximadamente, correspondiente al funcionamiento del periférico.
- Parámetros externos como los son las nubes, línea de vista con los satélites, constantes cambios en la geografía del trayecto, entre otros, son difíciles de controlar y afectan significativamente el consumo del GPS creando un esfuerzo mayor al dispositivo.

- Variar los parámetros minTime y minDistance con respecto a la velocidad del dispositivo puede generar un menor consumo en este.
- Determinar el contexto en el cual es utilizada la aplicación de localización puede ser una buena práctica para asignar los valores de minTime y minDistance.

**Audio**

- No se presenta una diferencia significativa entre las canciones utilizadas ya que fueron normalizadas.
- Entre una canción de salsa (Mosaico del grupo niche), una de reggaeton (Danza kuduro de Don Omar), una de rock (Back in black de AC/DC) y una de clásica (Sinfonía n°5 de Beethoven), la de reggaeton presenta un mayor consumo aun estando todas normalizadas a 95dB de volumen.
- La canción Back in Black de AC/DC grabada en vivo consume más que en estudio.
- La canción Back in Black de AC/DC grabada en vivo permite una duración de la batería 22 minutos mayor a la grabada en estudio si se reproducen continuamente.
- La reproducción con altavoz en nivel de 100 % de volumen consume más del doble que la reproducción con audífonos en nivel de 70 % de volumen
- La diferencia entre la canción de reggaetón Danza kuduro de Don Omar con volumen al 100 % y 75 % no normalizada es de 17,831mA.
- El control de volumen aplicado a la reproducción de la canción de reggaetón Danza kuduro de Don Omar disminuye utilizando los audífonos disminuye el consumo 21,887mA en comparación al alta voz.
- Controlar el volumen fijando niveles de volumen adecuados para el altavoz y los audífonos es una práctica que permite disminuir el consumo de energía.

**Acelerómetro**

- La diferencia entre utilizar un tiempo de 200ms y uno de 5000ms es de 6.4mA, correspondiente a tener 29 minutos más de batería en el dispositivo Nexus S.
- El tiempo de retardo indicado en el registerListener() es solo una sugerencia para el sistema ya que este toma un tiempo menor al que se le asigna.
- La referencia SENSOR_SERVICE produce un consumo aproximado de 3mA.
- Desplegar datos en pantalla sin tener un tiempo para que estos sean desplegados (0ms) produce un aumento de consumo con respecto a la aplicación sin datos en pantalla de 126,629mA, que equivale 11 horas con 50 minutos de uso del dispositivo Nexus S.
- El acelerómetro genera un consumo neto de 5.681mA sin datos en pantalla, lo que equivale a 11 días de duración en una batería de 1500mAh.
- 1000ms como tiempo de despliegue es una tasa aceptable para usar el acelerómetro sin tener un consumo elevado

**Brújula**

- Desplegar datos en pantalla sin tener un tiempo para que estos sean desplegados (0ms) produce un aumento de consumo con respecto a la aplicación sin datos en pantalla de 107,56mA, que equivale 13 horas con 56 minutos de uso del dispositivo Galaxy Ace.
- La brújula genera un consumo neto de 11,705mA sin datos en pantalla, lo que equivale a 5 días, 8 horas y 10 minutos de duración en una batería de 1500mAh.

**Teclado**

- El consumo del teclado depende del número de pulsos que se generen por presionar la pan-talla y la duración de estos.
- Los pulsos generados por el teclado QWERTY predictivo y no predictivo consumen en pro-medio lo mismo.
- El teclado QWERTY predictivo consume menos después de un tiempo de uso ya que aprende los patrones de palabras usadas por los usuarios haciendo sus predicciones cada vez mas precisas.
- El consumo del teclado varía dependiendo del modo de uso que cada usuario pueda presentar.
- El teclado SwiftKey 3 es el teclado que presenta menor consumo.
- Las interfaces de los teclados poseen un valor de consumo muy parecido entre ellas.
- El teclado Swype aunque tiene un consumo mayor que el SwiftKey 3 presenta un menor consumo que el QWERTY predictivo.
- El teclado Swype no presenta un consumo por pulsos individuales sino por palabras.
- El teclado SwiftKey 3 presenta pulsos por letra con menor amplitud y menor duración que en el QWERTY.
- El teclado SwiftKey 3 presenta menos pulsos por letra que el teclado QWERTY.

## 7.2. Contribuciones

### 7.2.1. Modelo para la evaluación de consumo de energía para dispositivos móviles

Se ha creado un modelo que permite evaluar el consumo de energía y crear practicas para su gestión a través de la experimentación, que incluye los siguientes aportes individuales:

- El modelo provee los criterios necesarios para una medición adecuada del consumo de energías en dispositivos móviles.
- El modelo presenta dos métodos para la sincronizan de eventos implementados en las aplicaciones desarrolladas, estos métodos son los picos de consumo y las marcas con retardos.
- El modelo presenta tres procesos que facilitan la creación de practicas de gestión de energía, estos son el proceso para la planificación de pruebas (P.P.P.), el proceso para la ejecución de pruebas (P.E.P.) y el proceso para el análisis de mediciones (P.A.M.).

### 7.2.2. Herramienta para la medición de consumo de energía en dispositivos móviles

Se ha diseñado e implementado una herramienta para la medición de consumo de energía que cumple los criterios para la adquisición de medidas del modelo y retoma algunas características de las herramientas existentes. Los aportes y componentes de la herramienta son los siguientes:

- Un dispositivo hardware que consta de una tarjeta de adquisición de datos con conexión USB hacia un computador.
- Diseño de vías de conexión para PCB y archivos *gerber* para su fabricación.
- Un dispositivo para la conexión de baterías de forma externa a un dispositivo móvil.
- Una aplicación cliente desarrollada en el lenguaje java que permite recopilar la información enviada desde la tarjeta de adquisición de datos y graficarla en tiempo real, permitiendo guardar esta información en archivos .csv y .jpg. Adicionalmente permite abrir los archivos guardados y realizar cálculos de promedios de corriente, voltaje y potencia sobre un intervalo de tiempo.

### 7.2.3. Conjunto de pruebas de medición

- En busca de practicas de programación se diseñó un conjunto de pruebas utilizado el proceso para la planificación de pruebas del modelo para la evaluación de consumo de energía para dispositivos móviles sobre los periféricos GPS, sistema de audio (auricular y altavoz), acelerómetro, brújula y teclado.

### 7.2.4. Caracterización del consumo de energía

- Se caracteriza el consumo de energía de cada periférico, en diferentes escenarios bajo el modelo de evaluación de consumo propuesto utilizando el dispositivo Nexus S de Samsung y la herramienta para la medición del consumo de energía en dispositivos.

### 7.2.5. Practicas de Gestión de energía en aplicaciones

- Se encontraron practicas para la gestión de energía en aplicaciones en cuanto a su uso y programación como resultado del conjunto de pruebas diseñadas y ejecutadas con base en el Modelo para la evaluación de consumo de energía para dispositivos móviles sobre los periféricos GPS, sistema de audio (auricular y altavoz), acelerómetro, brújula y teclado.

## 7.3. Lecciones Aprendidas

- En el diseño hardware se realizan cálculos con valores teóricos de los componentes los cuales en la realidad son imprecisos y es necesario realizar una calibración con valores reales utilizado una herramienta de una buena precisión como un multimetro muy exacto.
- La librería Jpicusb que permite la comunicación usb con el microcontrolador solo es compatible con el jdk de 32 bit.
- La utilización de bucles infinitos en aplicaciones android produce un consumo muy alto aproximadamente 300mA.
- La pantalla táctil produce un gran consumo al ser el periférico con el que mas se interactua.

- El sensor de orientación en el dispositivo Nexus S no presenta mediciones cuando la batería se conecta externamente al dispositivo.
- EL dispositivo Samsung Galaxy ACE con android 2.3.6 presenta menores consumos en los mismos perifericos comparado con el Nexus S con android 4.1.2.
- Los conectores que traen las baterias varian su tamaño y su posición dependiendo de la marca del dispositivo.

## 7.4. Trabajos Futuros

- Para mejorar la resolución de la herramienta se puede utilizar un microcontrolador con un ADC de más de 10 bits, por ejemplo se puede utilizar un PIC18F4553 el cual cuenta con un ADC de 12 bits y es compatible con la herramienta y software desarrollado.
- Estudiar practicas de gestión de energía sobre la pantalla táctil ya que es uno de los periféricos con los que mas se interactua.
- Estudiar practicas de gestión de energía sobre el servicio de localización utilizando los proveedores posibles a través de la clase *LocationProvider* y la clase *Criterial* para manipular su utilización de forma dinámica.
- Aplicar el modelo para la evaluación del consumo de energía en dispositivos móviles a dispositivos que soporten otros sistemas operativos.

## 7.5. Actividades de divulgación y socialización del conocimiento

Se tuvo la oportunidad de realizar un publicación internacional sobre el trabajo, además de varias ponencias y entrevistas en médios.

### 7.5.1. Evento internacional

Paper: *"Consumer-oriented mobile device and model for energy management in android mobile phones"*, en el evento **"The Seventh International Conference on Digital Society-ICDS 2013"**. Realizado en Nice, France del 24 de Febrero al 1 de Marzo. `http://www.iaria.org/conferences2013/ProgramICDS13.html`

### 7.5.2. Evento regional

Presentación del avance del trabajo de grado en el evento "Tic-s Park 201"'. Realizado en Cali, Colombia?Universidad Católica, realizado el 10 de Noviembre de 2012.

### 7.5.3. Evento local

Presentaciones en el **Semillero WapColombia** en la Universidad del Cauca el viernes 27 de Junio y el 30 de Noviembre de 2012.

### 7.5.4. Divulgación en medios

- Nota de **Canal ET** (Casa Editorial El Tiempo) *ESPECIAL PERIODÍSTICO TECNOLOGÍA MOVIL CAUCA*, transmitido el día 28 de Enero de 2013. url: `http://www.youtube.com/watch?v=UA0yvuoZMyY&feature=youtu.be`
- Nota de prensa de la Universidad del Cauca llamada *"Grupo de Ingeniería Telemática desarrolla proyecto sobre consumo de energía en dispositivos móviles"*, publicada el día 19 de marzo de 2013. url: `http://www.unicauca.edu.co/versionP/noticias/investigaci%C3%B3n/gti-desarrolla-proyecto-sobre-consumo-energia-moviles`

# Referencias


[1] P. Ranganathan, "Recipe for efficiency: Principles of power-aware computing," *In Communications of the ACM*, vol. 53, pp. 60–67, 2010.

[2] M. Musolesi, M. Piraccini, K. Fodor, A. Corradi, and A. Campbell, "Supporting energy-efficient uploading strategies for continuous sensing applications on mobile phones," in *Pervasive Computing* (P. Floréen, A. Krüger, and M. Spasojevic, eds.), vol. 6030 of *Lecture Notes in Computer Science*, pp. 355–372, Springer Berlin / Heidelberg, 2010.

[3] M. Youssef, M. A. Yosef, and M. El-Derini, "Gac: Energy-efficient hybrid gps-accelerometer-compass gsm localization," *GLOBECOM 2010 2010 IEEE Global Telecommunications Conference*, pp. 1–5, 2010.

[4] K. Nishihara, K. Ishizaka, and J. Sakai, "Power saving in mobile devices using context-aware resource control," in *Proceedings of the 2010 First International Conference on Networking and Computing*, ICNC '10, (Washington, DC, USA), pp. 220–226, IEEE Computer Society, 2010.

[5] I. Buchmann, "What is the perfect battery?." [Online]. Available: http://www.buchmann.ca/article4-page1.asp, April 2001. [Accessed Aug 10, 2011].

[6] Gartner, "Gartner says sales of mobile devices grew 5.6 percent in third quarter of 2011; smartphone sales increased 42 percent." [Online]. Available: http://www.gartner.com/it/page.jsp?id=1848514, November 2011. [Accessed Dec 1, 2011].

[7] R. Wauters, "37ofios apps." [Online]. Available: http://techcrunch.com/2011/10/21/37-of-published-android-apps-were-later-removed-compared-to-24-of-ios-apps/, October 2011. [Accessed Nov 10, 2011].

[8] C. Serrano, *Modelo para la Construcción de Soluciones*, pp. 43–58. Ed. Popayán, Cauca: ed. Popayán., 2008.

[9] T. K. Kundu and K. Paul, "Improving android performance and energy efficiency," in *Proceedings of the 2011 24th International Conference on VLSI Design*, VLSID '11, (Washington, DC, USA), pp. 256–261, IEEE Computer Society, 2011.

[10] U. Battery, "Glosary." [Online]. Available: http://batteryuniversity.com/learn/article/definitions. [Accessed sep 2, 2011].

[11] J. Sharkey., "Coding for life – battery life, that is." [Online]. Available: http://www.google.com/events/io/2009/sessions/CodingLifeBatteryLife.html, 2009. [Accessed Dec 1, 2011].

[12] U. Battery, "What is the Best Battery?." [Online]. Available: http://batteryuniversity.com/learn/article/whats_the_best_battery. [Accessed sep 2, 2011].

[13] Syscom, "¿la batería de litio es la batería ideal?." [Online]. Available: http://www.syscom.com.mx/secciones/distribuidores/La-bateria-de-Litio-Ion-es-la-bateria-ideal.html. [Accessed Aug 13, 2011].

[14] R. Bajaj, S. L. Ranaweera, and D. P. Agrawal, "Gps: Location-tracking technology," *Computer*, vol. 35, pp. 92–94, Apr. 2002.

[15] H. Martínez and G. Moreno, "Index of aplicación de gestión y seguimiento vía gps sobre dispositivo móvil i plataforma android," *January*, pp. 77–78, 2011.

[16] G. Bai, "Mobile phone programming - based on mobile sensor api for user interface," p. 8, May 2010.

[17] D. Engineering, "A beginners guide to accelerometers." [Online]. Available: http://www.dimensionengineering.com/accelerometers.htm. [Accessed Aug 22, 2011].

[18] A. Developers, "android.sensorevent." [Online].Available: http://developer.android.com/reference/-android/hardware/SensorEvent.html#values. [Accessed Sep 10, 2011].

[19] A. Developers, "What is android?." [Online]. Available: http://developer.android.com/guide/basics/what-is-android.html. [Accessed Sept 5, 2011].

[20] J. Aranaz, *Desarrollo de aplicaciones móviles sobre la plataforma android de google*, pp. 37 – 39. Proyecto de fin de carrera Universidad Carlos III de Madrid, 2009.

[21] A. Developers, "What is the ndk?." [Online]. Available: http://developer.android.com/sdk/ndk/overview.html. [Accessed Feb 20, 2012].

[22] A. Developers, "android.hardware." [Online].Available: http://developer.android.com/reference/-android/hardware/package-summary.html. [Accessed Sep 10, 2011].

[23] A. Developers, "android.location." [Online]. Available: http://developer.android.com/reference/-android/location/package-summary.html. [Accessed Sep 10, 2011].

[24] A. Developers, "android.media." [Online]. Available: http://developer.android.com/reference/-android/media/package-summary.html. [Accessed Sep 10, 2011].

[25] A. Developers, "android.view.inputmethod." [Online]. Available: http://developer.android.com/reference/-android/view/inputmethod/package-summary.html. [Accessed Sep 10, 2011].

[26] A. Rice and S. Hay, "Decomposing power measurements for mobile devices," *2010 IEEE International Conference on Pervasive Computing and Communications PerCom*, pp. 70–78, 2010.

[27] K. Paul and T. K. Kundu, "Android on mobile devices: An energy perspective," *2010 10th IEEE International Conference on Computer and Information Technology*, vol. 3, no. Cit, pp. 2421–2426, 2010.

[28] M. Marina and M. Calder, "Batch scheduling of recurrent applications for energy savings on mobile phones," *Sensor Mesh and Ad Hoc Communications and Networks SECON 2010 7th Annual IEEE Communications Society Conference on*, pp. 1–3, 2010.

[29] K.-H. Kim, A. W. Min, D. Gupta, P. Mohapatra, and J. P. Singh, "Improving energy efficiency of wi-fi sensing on smartphones.," in *INFOCOM*, pp. 2930–2938, IEEE, 2011.

[30] I. Zahid, M. Ali, and R. Nassr, "Android smartphone: Battery saving service," *2011 International Conference on Research and Innovation in Information Systems (ICRIIS)*, pp. 1–4, 2011.

[31] T. Instruments, "Lm336-2.5." [Online]. Available: http://www.ti.com/lit/ds/symlink/lm136-2.5-n.pdf. [Accessed mar 6, 2012].

[32] Cadsoftusa, "Cadsoft eagle pcb design software." [Online]. Available: http://www.cadsoftusa.com/eagle-pcb-design-software/?language=en. [Accessed Mar 8, 2012].

[33] I. CCS, "Code optimizing c compiler." [Online]. Available: http://www.ccsinfo.com/content.php?page=compilers. [Accessed Feb 10, 2012].

[34] O. Corporation, "Netbeans ide." [Online]. Available: http://netbeans.org/features/index.html. [Accessed Feb 10, 2012].

[35] G. Play, "Android system info." [Online]. Available: https://play.google.com/store/apps/-details?id=com.electricsheep.asi&hl=es, 2012. [Accessed Dec 16, 2012].

[36] G. Milette and A. Stroud, *Professional Android Sensor Programming*. Programmer to programmer, Wiley, 2012.

[37] A. Developers, "android.media.soundpool." [Online]. Available: http://developer.android.com/reference/-android/media/SoundPool.html. [Accessed nov 10, 2012].

[38] A. Developers, "android.media.mediaplayer." [Online]. Available: http://developer.android.com/reference/-android/media/MediaPlayer.html. [Accessed nov 10, 2012].

[39] A. Developers, "android.media.audiotrack." [Online]. Available: http://developer.android.com/reference/-android/media/AudioTrack.html. [Accessed nov 10, 2012].

# PRÁCTICAS DE GESTIÓN DE ENERGÍA EN APLICACIONES PARA DISPOSITIVOS ANDROID, SOPORTADAS EN PERIFÉRICOS ESPECÍFICOS

# ANEXOS


Universidad
del Cauca

Trabajo de Grado

**Manuel Fernando Fuentes Amaya**
**Johan Alberto Gómez Girón**

Director: PhD. Gustavo Adolfo Ramírez González

**Universidad del Cauca**
**Facultad de Ingeniería Electrónica y Telecomunicaciones**
**Departamento de Telemática**
**Línea de Investigación en Servicios avanzados en Telecomunicaciones**
**Popayán, Abril de 2013**

# Anexo A

# Diagramas de la herramienta para la medición del consumo de energía en dispositivos móviles

Este anexo contiene los diagramas y gráficos correspondientes a la herramienta para la medición del consumo de energía en dispositivos móviles.El anexo se divide en tres partes principales, la primera correspondientes al hardware, la segunda corresponde al firmware del microcontrolador y la tercera corresponde a la aplicación cliente.

## A.1. Diagrama hardware

El diagrama hardware final de la herramienta se pueden observar en la figura A.1.

### A.1.1. Lista de elementos

Los elementos hardware utilizados en la herramienta se muestran en la tabla A.1.

| LISTA DE ELEMENTOS | | |
|---|---|---|
| D1, D2: 1N4148 | R1 , R2: 100kΩ | C1: 470nf |
| IC1: PIC18F4550 | R3 , R7, R14: 2.5kΩ | C2, C3: 20pf |
| IC2: AD620N | R4 : 1.5k | C4 : 10pf |
| IC3: LM336LP 2.5 | R5 : 5kΩ | C5, C6: 0.1uf |
| JP1: ConectorBatería | R6 : 220 | |
| JP2: Conector Fuente dual | R8 , R9, R10, R11, R12: | |
| JP3:ConectorDispositivo | 0.11Ω | |
| LED1, LED2: leds | R13: 1 M | |
| X1: USB_PC | | |

Tabla A.1: Lista de elementos herramienta hardware.

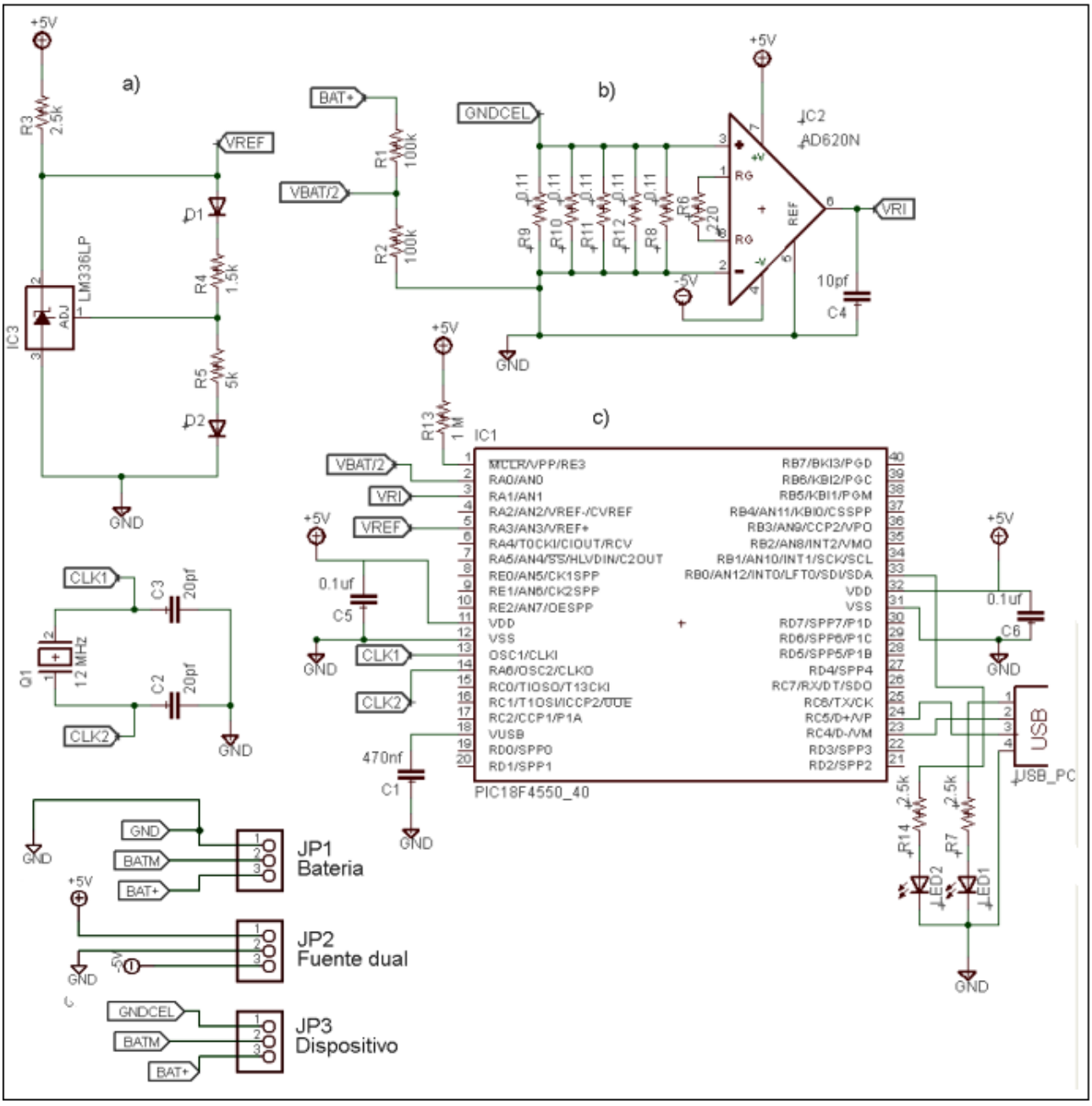


Figura A.1: Diagrama Hardware: a) Voltaje de referencia, b)Módulo de medición de voltaje y corriente, c) Modulo de control de medición, JP1) Conectores entrada batería dispositivo, JP2) Conectores entrada fuente dual, JP3) Conectores dispositivo.

## A.2. Modelado del Firmware del microcontrolador

### Diagrama del algoritmo implementado

La figura A.2 muestra el algoritmo implementado para el firmware del microcontrolador.

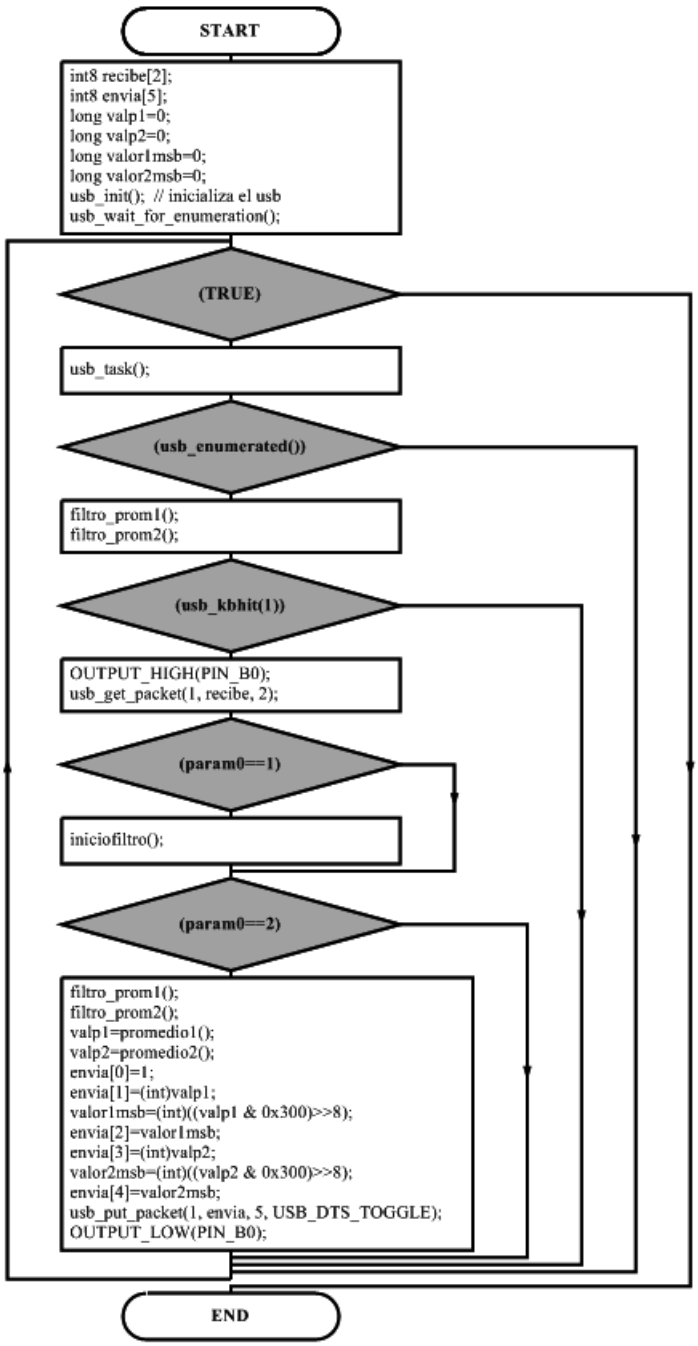


Figura A.2: Diagrama de flujo Firmware Microcontrolador.

### Diagrama de flujo filtro promedio

La figura A.3 muestra el algoritmo utilizado en los métodos filtropromedio1() y filtropromedio2(), el cual permite estabilizar las mediciones del ADC.

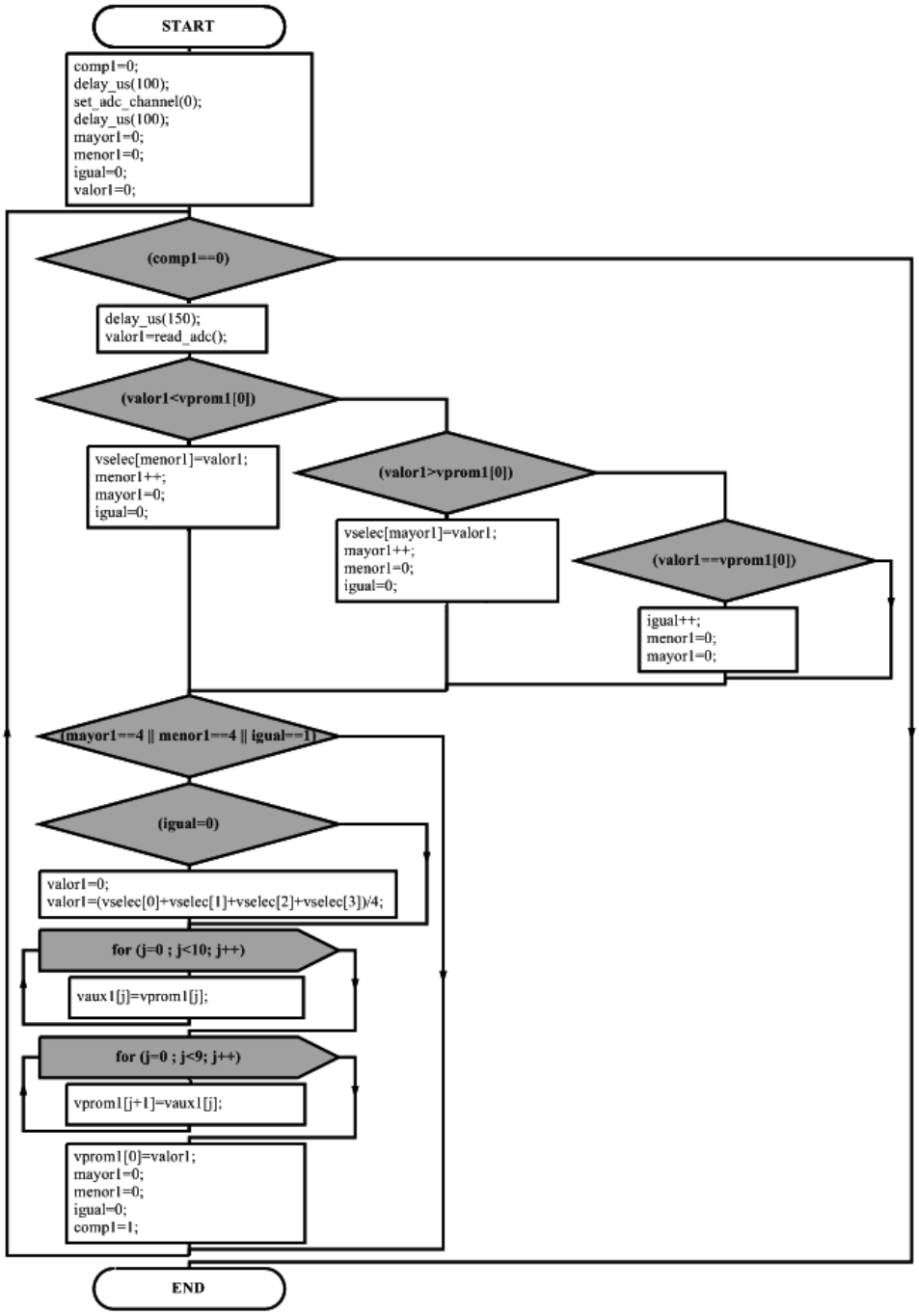


Figura A.3: Diagrama de flujo filtro promedio.

**Diagrama de flujo del Cálculo del promedio**

La figura A.4 muestra el algoritmo utilizado en los método promedio1() y promedio2(), el cual permite mejorar la precision de las mediciones del ADC.

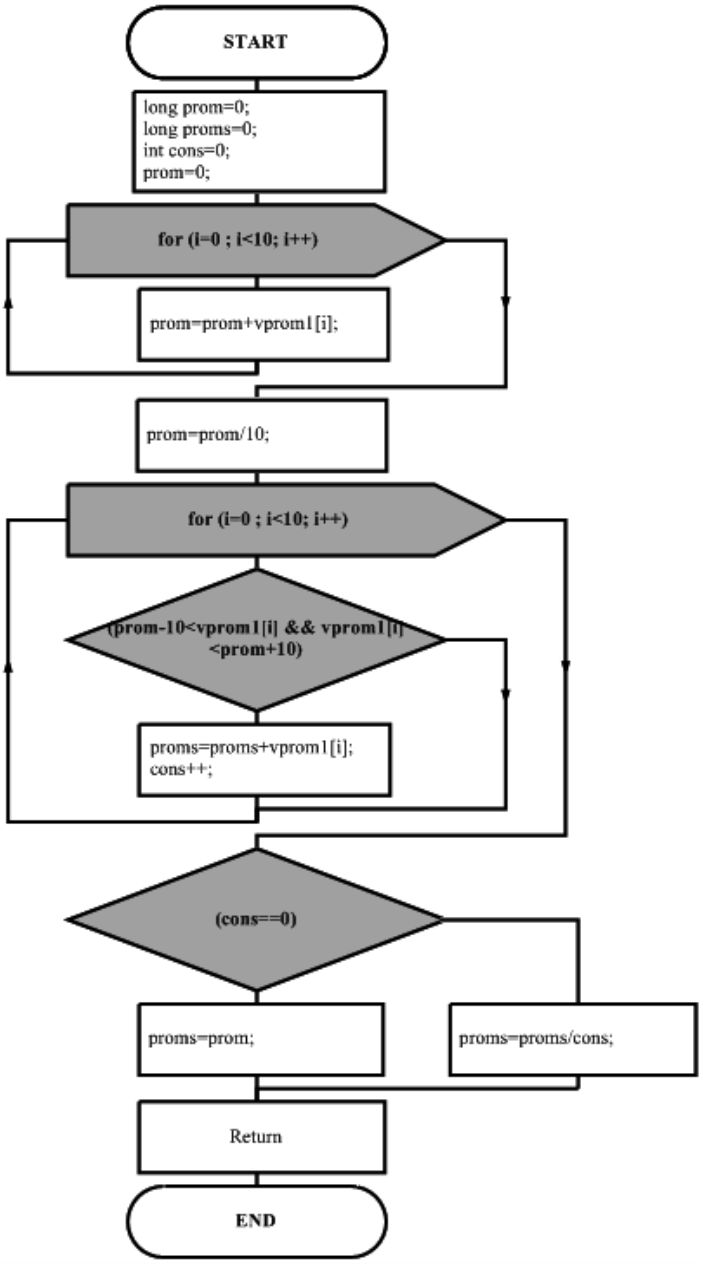


Figura A.4: Diagrama de flujo del Cálculo del promedio

## A.3. Modelado de la aplicación cliente

### A.3.1. Interfaces gráficas

La figuras A.5 y A.6 corresponden a dos de las interfaces de la aplicación cliente.

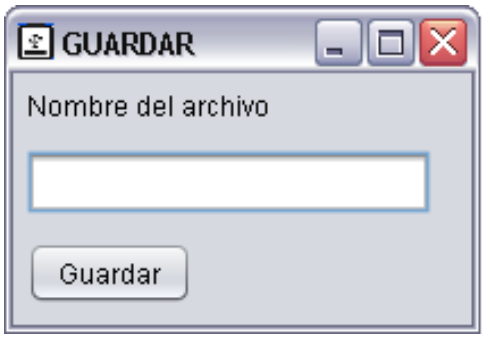


Figura A.5: Interfaz gráfica Panel_guardar.


Figura A.6: Interfaz gráfica Info.

### A.3.2. Diagramas de clase

A continuación se presentan los diagramas de clase de la aplicación cliente.

#### Paquete Medicion_vista


Figura A.7: Diagrama de clase Calcular_promedios.

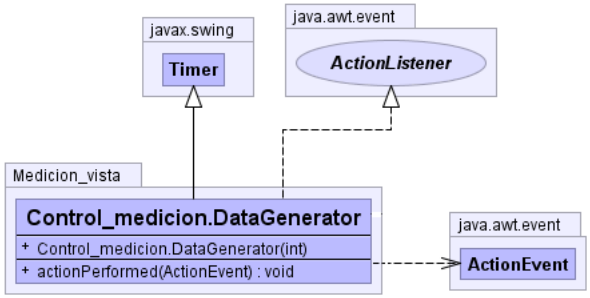


Figura A.8: Diagrama de clase DataGenerator.

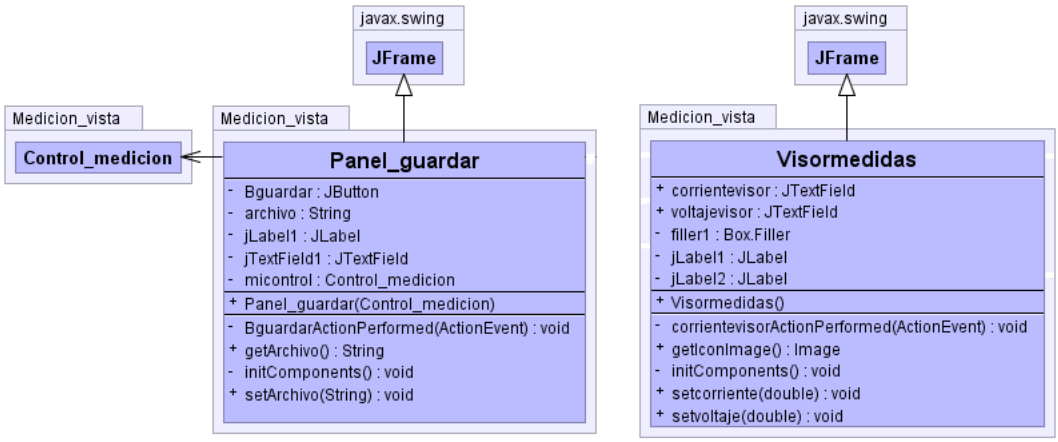


Figura A.9: Diagrama de clases Visormedidas y panel_Guardar.

### Paquete Conexión

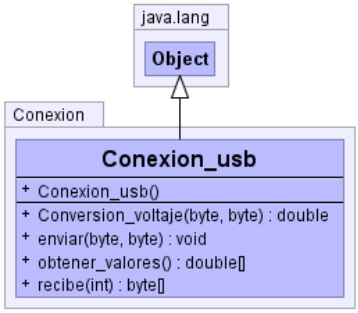


Figura A.10: Diagrama de clase Conexion_usb.

### Paquete Modelo

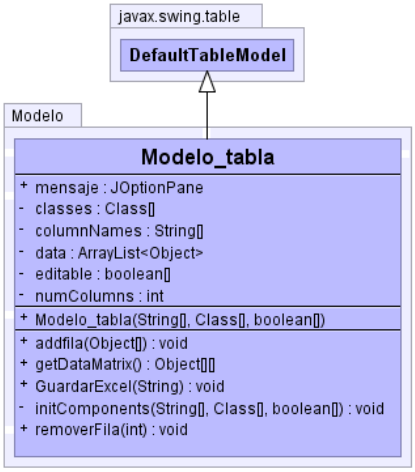


Figura A.11: Diagrama de clase Modelo_tabla.

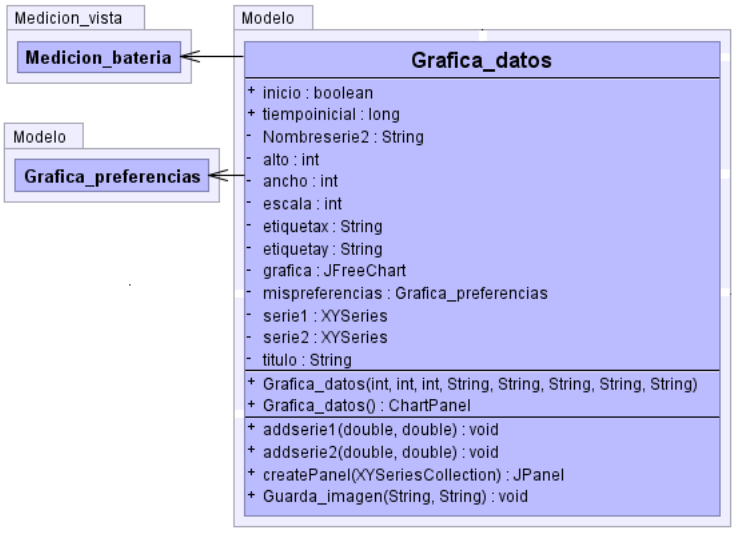


Figura A.12: Diagrama de clase Grafica_datos.

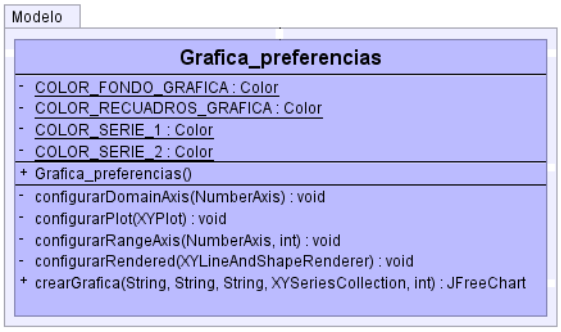


Figura A.13: Diagrama de clase Grafica_preferencias.

### Diagramas de secuencia

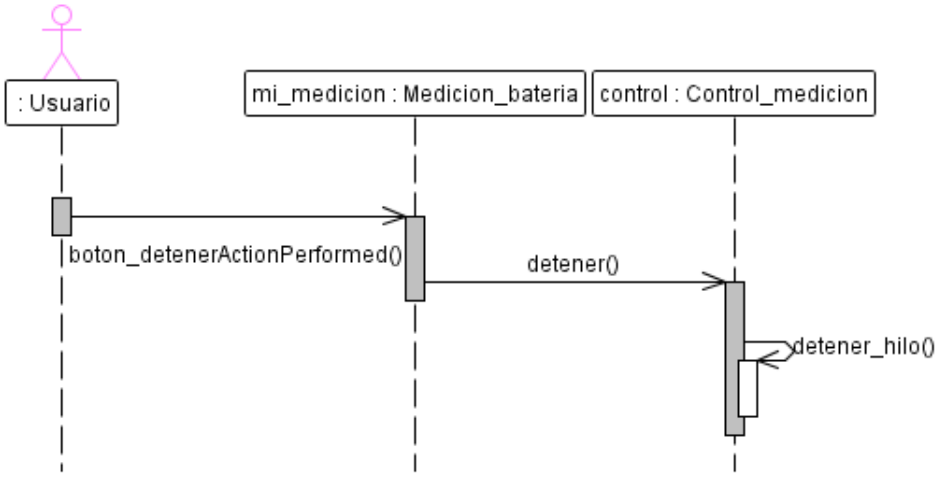


Figura A.14: Diagrama de secuencia - Detener.

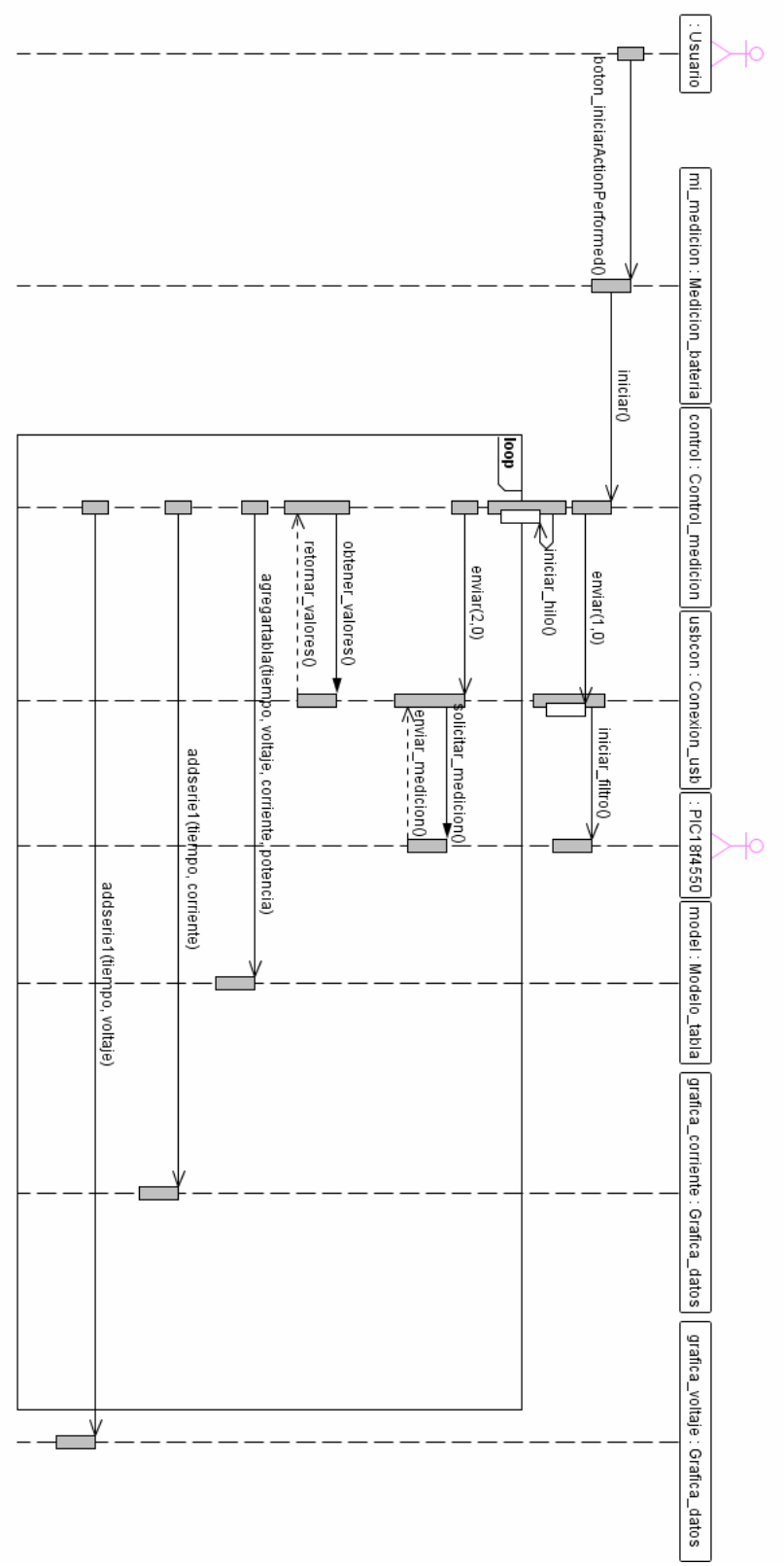


Figura A.15: Diagrama de secuencia - Iniciar.

Diagrama de secuencia - Iniciar.

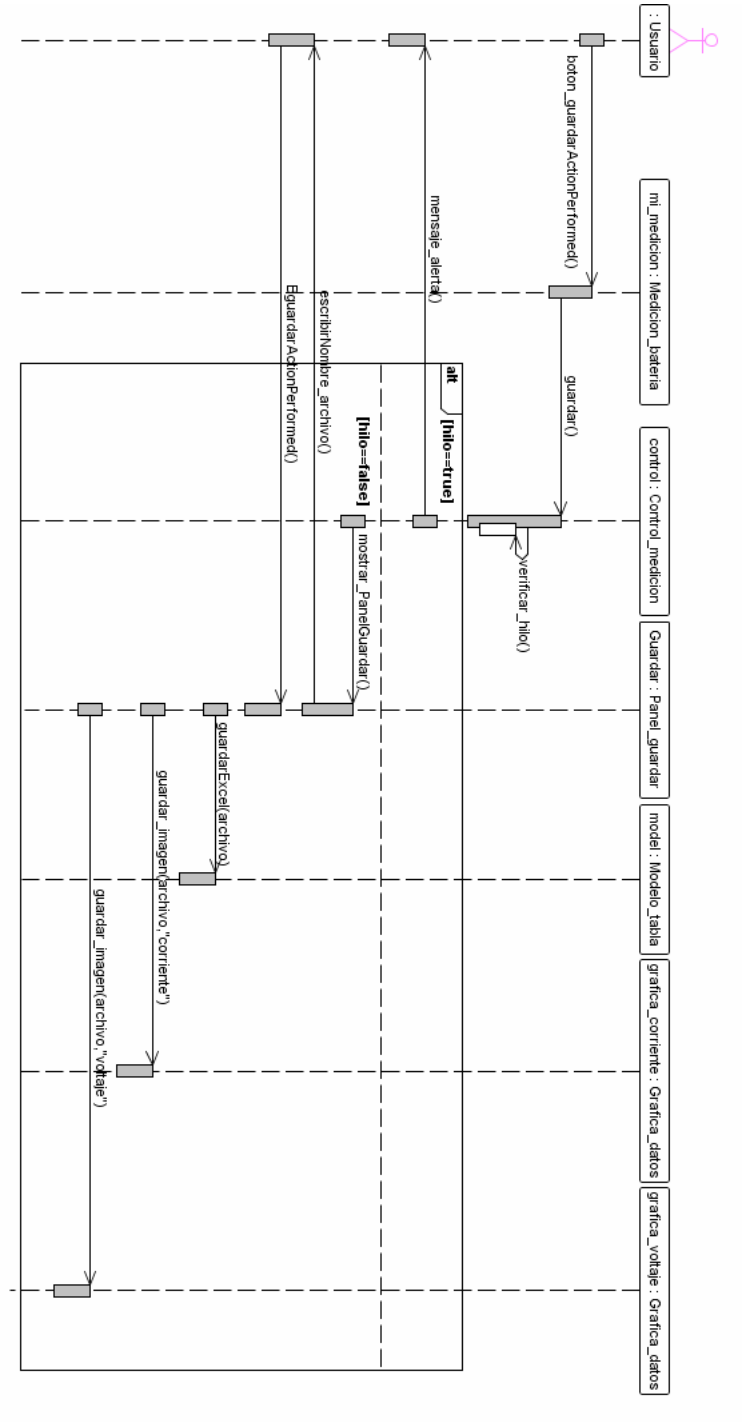


Figura A.16: Diagrama de secuencia - Guardar.

# Anexo B

# Manual de usuario Herramienta para la Medición de Consumo de Energía

Este anexo aporta el manual de usuario de la herramienta para la Medición de Consumo de Energía desarrollada en el capítulo 4. El manual consta de dos partes correspondientes a las conexión de la herramienta hardware y las instrucciones de uso de la aplicación cliente medidor batería 1.0.

## B.1. HERRAMIENTA HARDWARE

La herramienta hardware es un dispositivo que se conecta entre el dispositivo móvil y al batería de este, permite tomar medidas físicas de voltaje y corriente. Consta de 3 componentes los cuales son: Tarjeta PCB, Batería falsa y conector para batería.

### Tarjeta PCB

En la figura B.6 se observan los puertos de conexión que posee la herramienta hardware en la tarjeta PCB.

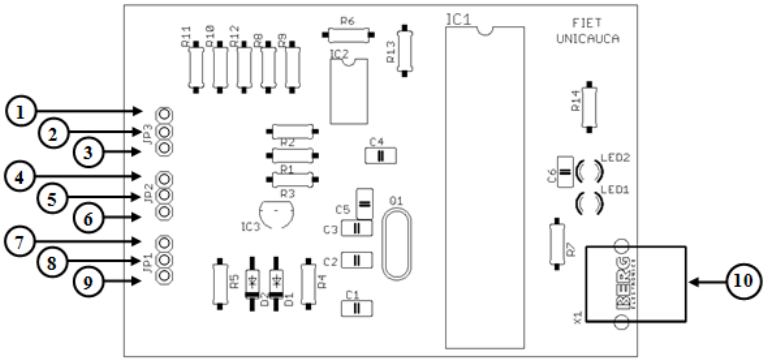


Figura B.1: Puertos de conexión tarjeta PCB - Herramienta hardware.

1. DISPOSITIVO $(-)$: Conector negativo del dispositivo
2. DISPOSITIVO NEUTRO: Conector central del dispositivo
3. DISPOSITIVO (+): Conector positivo del dispositivo

4. FUENTE -5: -5 voltios fuente dual
5. GND: Tierra fuente dual
6. FUENTE +5: +5 voltios fuente dual
7. BATERIA $(-)$: Conector negativo de la batería.
8. BATERIA NEUTRO: Conector central de la batería.
9. BATERIA (+): Conector positivo de la batería.
10. CONECTOR USB: Conector que va al puerto USB del PC por medio de un cable USB.

**Batería falsa**

La batería falsa es un mecanismo que reemplaza la batería sobre el dispositivo móvil. Esta consta de 3 conectores como se puede observar en la figura B.8.


Figura B.2: Batería falsa - Herramienta hardware.

1. BATERÍA FALSA $(-)$: Conector negativo de la batería falsa.
2. BATERÍA FALSA NEUTRO: Conector central de la batería falsa.
3. BATERÍA FALSA (+): Conector positivo de la batería falsa.

**Conexión a Tarjeta PCB:** La batería falsa tiene la siguiente conexión con la tarjeta PCB:

- BATERÍA FALSA $(-)$ con DISPOSITIVO $(-)$.
- BATERÍA FALSA NEUTRO con DISPOSITIVO NEUTRO.
- BATERÍA FALSA (+) con DISPOSITIVO (+).

**Nota:** La distribución de los pines positivo y negativo pueden cambiar en algunas baterías, favor observar antes de conectar.

### Conector para batería

Este conector permite colocar la batería de un dispositivo móvil de forma externa, y así realizar la conexión de esta con la tarjeta PCB. Este consta de 2 partes y de 3 conectores, así como se describe a continuación

**Las partes del conector son los siguientes y se pueden observar en la figura B.9:**

1. Resorte de apoyo: Permite graduar el tamaño de la batería sobre el conector, ya que las batería pueden varía de tamaño dependiendo el dispositivo móvil.

2. Caja para batería: Caja en donde se aloja la batería, además posee las 3 conexiones.

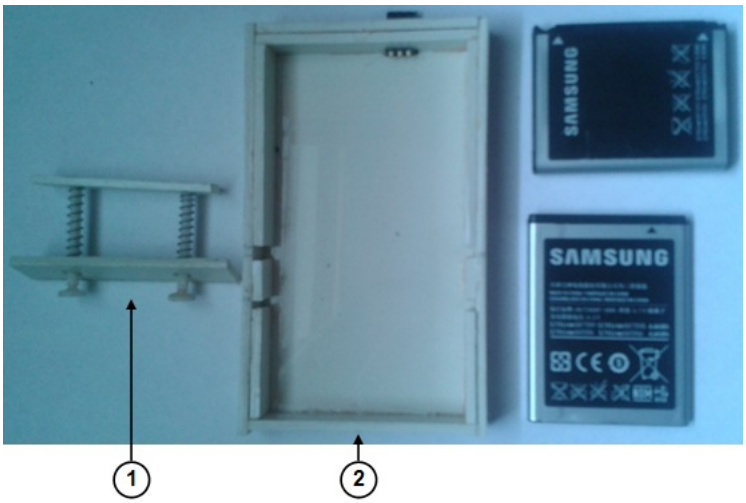


Figura B.3: Partes del conector para batería - Herramienta hardware.

La correcta conexión de las anteriores partes se puede observar en la figura B.10.:

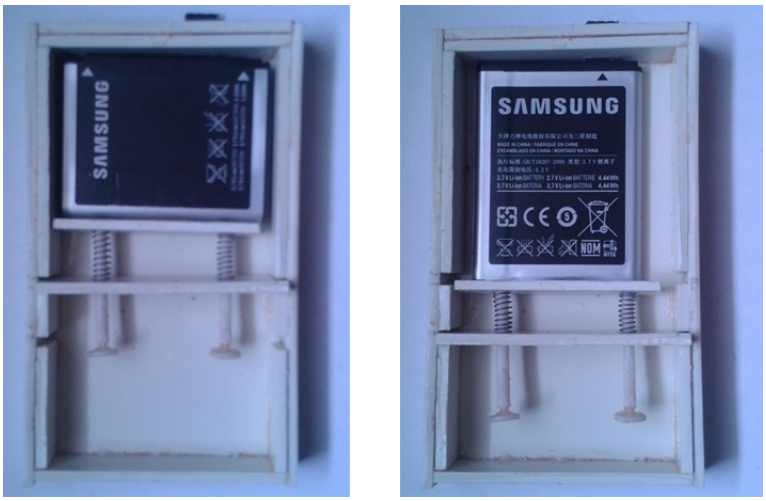


Figura B.4: Conexión del conector en 2 diferentes tamaños de batería - Herramienta hardware.

**Las conexiones del conector son las siguientes y se pueden observar en la figura B.11:**

1. CONECTOR PARA BATERÍA $(-)$: Conector negativo del conector para batería.
2. CONECTOR PARA BATERÍA NEUTRO: Conector central del conector para batería.
3. CONECTOR PARA BATERÍA (+): Conector positivo del conector para batería.

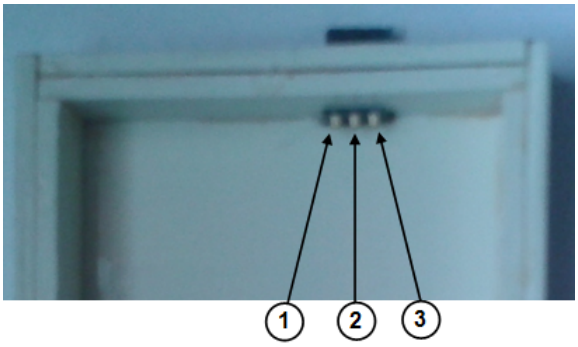


Figura B.5: conexiones del conector para batería - Herramienta hardware.

**Nota:** Dependiendo la batería, los conectores $(-)$ y (+) pueden intercambiarse, favor verificar esto.

**Conexión a Tarjeta PCB:** El conector para batería tiene la siguiente conexión con la tarjeta PCB:

- CONECTOR PARA BATERÍA $(-)$ con BATERIA $(-)$.
- CONECTOR PARA BATERÍA NEUTRO con BATERIA NEUTRO.
- CONECTOR PARA BATERÍA (+) con BATERIA (+).

## B.2. MEDIDOR BATERÍA 1.0 (APLICACIÓN CLIENTE)

Descripción de la aplicación: Medidor batería 1.0 es una aplicación de escritorio que permite obtener una medida práctica y eficiente sobre el consumo de un dispositivo, específicamente en cuanto a su corriente, voltaje y potencia. Los datos son desplegados en un tabla y en dos gráficas durante la toma de las mediciones. Además permite guardar los datos y las gráficas en un archivo .csv y en archivos de imagen .png, respectivamente.

### Condiciones Iniciales

Para poder desplegar y hacer uso de la aplicación Medidor Batería 1.0 , se presenta las condiciones iniciales que se deben tener en cuenta.

- Soporte Java para el sistema operativo `http://www.java.com/es/`.
- Soporte JRE de Java para Windows X86 `http://www.oracle.com/technetwork/java/javase/downloads/jre7u9-downloads-1859586.html` .
- Herramienta Hardware para la medición de batería, conectada al dispositivo y al ordenador.
- Driver de instalación dispositivo Microship PIC18f4550 USB `http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010300#developmentTools`.

### Funciones de la aplicación

En la figura B.6 se observan las funcionalidades que posee la aplicación las cuales se describen a continuación.

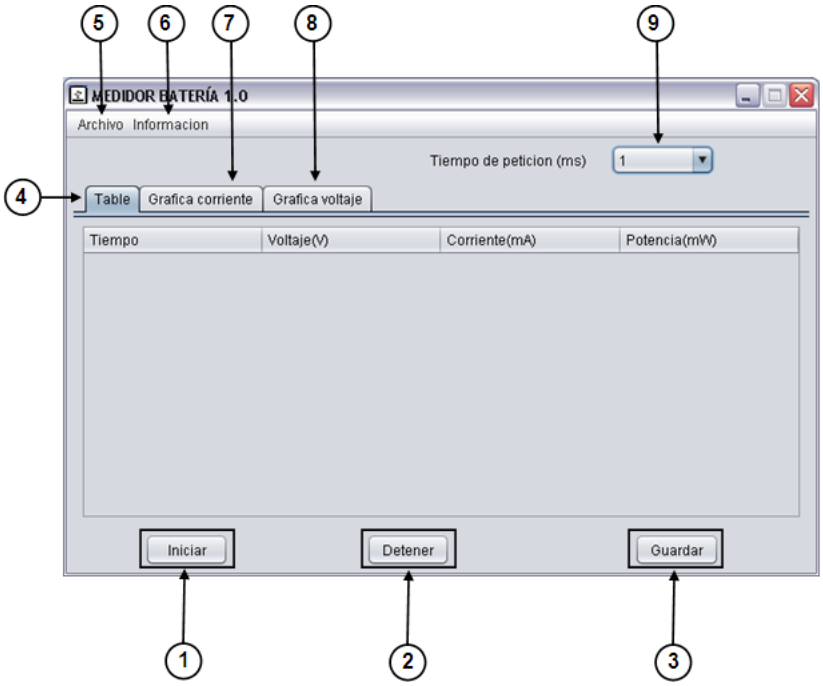


Figura B.6: Interfaz principal - Partes.

1. Botón Iniciar: Este botón permite inicializar la adquisición de medidas del dispositivo hardware.

2. Botón Detener: Permite detener la adquisición de medidas.

3. Botón Guardar: Guarda los datos obtenidos de la medición hasta el momento en una carpeta nueva con el nombre que se ingrese, esta carpeta quedará localizada en la carpeta "Mediciones".La figura B.7 muestra la opción de ingresar el nombre.

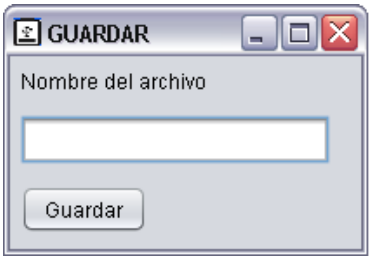


Figura B.7: Ingresar nombre - Opción guardar.

4. Pestaña Tabla: Permite visualizar los datos de la medición en forma de tabla

5. Menú archivo: Despliega el menu de la figura B.8.

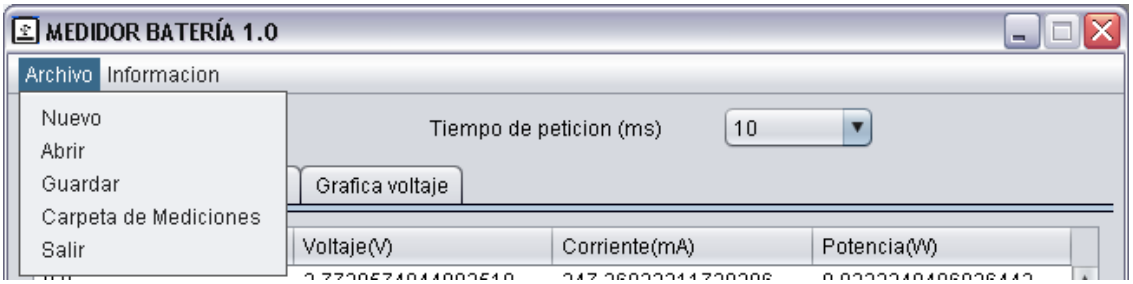


Figura B.8: Menú archivo - interfaz principal.

Este menú contiene las siguientes opciones:

- Nuevo: Permite reiniciar la aplicación para iniciar una nueva medición.
- Abrir: Permite abrir una medición localizada en la carpeta mediciones con extensión .csv.
- Guardar:Esta la misma función del botón guardar de la figuraB.6.
- Carpeta de mediciones: Permite visualizar la carpeta “Mediciones”.
- Salir: Cierra la aplicación.

6. Menú información: despliega el menú de la figura B.9.

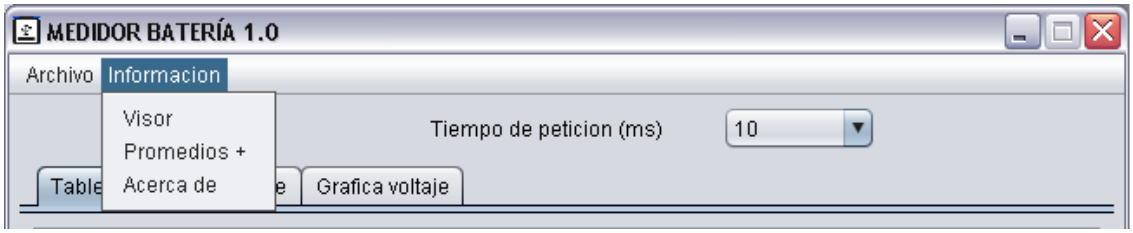


Figura B.9: Menú información - interfaz principal.

Este menú contiene las siguientes opciones:

- Visor: Despliega la ventana medidas la cual muestra los valores instantáneos de voltaje y corriente, como se puede observar en la figura B.10.
- Promedios: Despliega la ventana calculo de promedios, como se puede observar en la figura B.11.
- Acerca de: Despliega la información acerca de la aplicación, como se puede observar en la figura B.12.


Figura B.10: Ventana calculo de promedios.

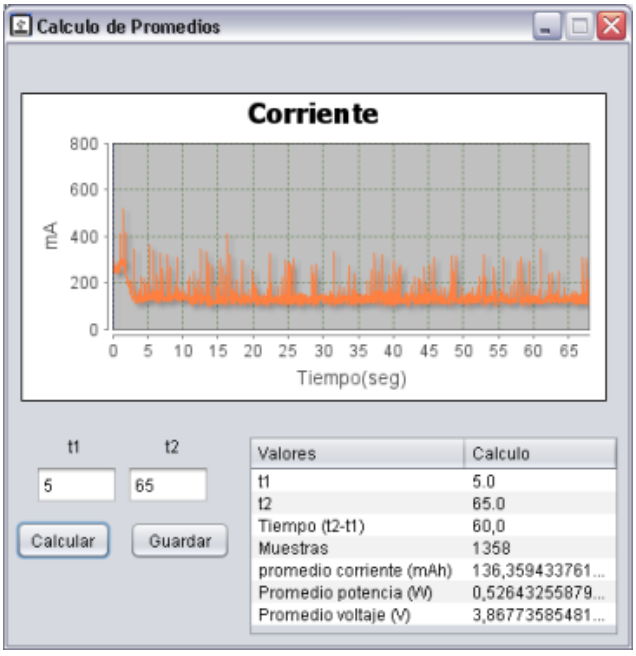


Figura B.11: Ventana calculo de promedios.


Figura B.12: Ventana de información de la aplicación.

7. Pestaña Gráfica corriente: Permite visualizar los datos correspondientes a la corriente en forma gráfica.
8. Pestaña Gráfica voltaje:Permite visualizar los datos correspondientes al voltaje en forma gráfica.
9. Selector Tiempo de petición: Permite seleccionar el retardo entre las peticiones de medición.

# Anexo C

# Guía – Prueba de Comprobación de la Herramienta para la Medición del Consumo de Energía

Esta guía está diseñada con el fin de realizar una adecuada comprobación del correcto funcionamiento de la herramienta, enfocándose en parámetros como:

- Precisión
- Frecuencia de medición
- Graficación en tiempo real
- Persistencia de la información

## C.1. PRECISIÓN

A continuación se establecen unos pasos para verificar la precisión:

1. Conectar en los terminales 1 y 3 del conector JP3 una resistencia fija, esto equivale a tener una carga fija remplazando el dispositivo móvil.

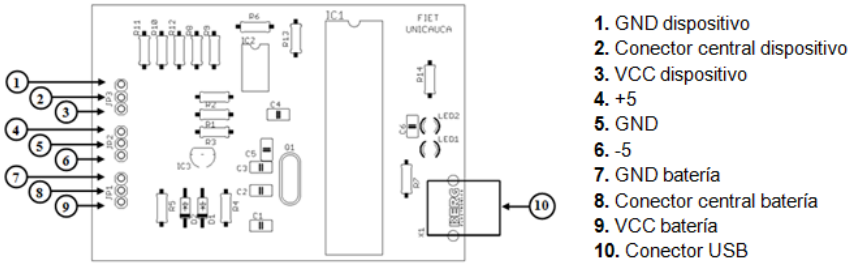


Figura C.1: Conectores de la Tarjeta PCB.

2. Medición del voltaje:

   *a*) Medir con el multímetro el voltaje en los terminales 7 y 9.

*b*) Ejecutar la aplicación cliente y posteriormente observar el voltaje obtenido. Esto se realiza seleccionando la opción Información/Visor, como se observar en la figura C.2.

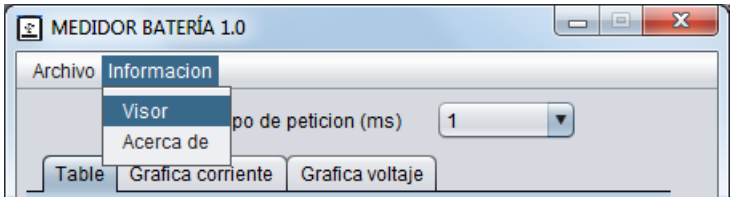


Figura C.2: Opción Información/Visor de la Aplicación Cliente.

El visor se muestra en la figura C.3.

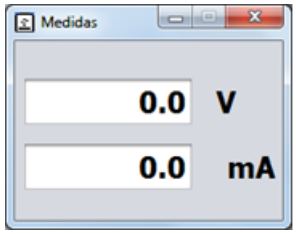


Figura C.3: Visor de la Aplicación Cliente.

*c*) Anotar los valores en la tabla C.1, aclarando que los voltajes en la batería de un dispositivo móvil generalmente se encuentra entre el rango de 3.6V hasta 4.2V:

| Voltaje Alimentación(V) | Voltaje Multímetro(V) | Voltaje Aplicación(V) | Concepto / Observación |
|---|---|---|---|
| 4.2 | 4.19 | 4.196 | Diferencia muy pequeña |
| 4.15 | 4.15 | 4.156 | Diferencia muy pequeña |
| 4.1 | 4.1 | 4.107 | Diferencia muy pequeña |
| 4.05 | 4.05 | 4.058 | Diferencia muy pequeña |
| 4 | 4 | 3.999 | Diferencia muy pequeña |
| 3.95 | 3.95 | 3.950 | Sin diferencia |
| 3.9 | 3.9 | 3.900 | Sin diferencia |
| 3.85 | 3.85 | 3.851 | Diferencia muy pequeña |
| 3.8 | 3.8 | 3.802 | Diferencia muy pequeña |
| 3.75 | 3.75 | 3.743 | Diferencia muy pequeña |
| 3.7 | 3.7 | 3.694 | Diferencia muy pequeña |
| 3.65 | 3.658 | 3.654 | Diferencia muy pequeña |
| 3.6 | 3.599 | 3.605 | Diferencia muy pequeña |
| 3.55 | 3.551 | 3.546 | Diferencia muy pequeña |
| 3.5 | 3.5 | 3.500 | Sin Diferencia |

Tabla C.1: Comparación de los voltajes de: Alimentación, Multímetro y la Aplicación

3. Medición de corriente:

*a*) Conectar el multímetro entre el VCC de la batería y el terminal 9, para así medir la corriente en el multímetro.

*b*) Ejecutar la aplicación cliente y posteriormente observar el corriente obtenido. Realizar el despliegue del visor como se indicó en el numeral 2*b*.

*c*) Variar la resistencia conectada en los terminales 1 y 3, y anotar los resultados en la tabla C.2

| Resistencia (Ω) | Corriente Multímetro(mA) | Corriente Aplicación(mA) | Concepto / Observación |
|---|---|---|---|
| 48 | 80.1 | 80.04 | Diferencia muy pequeña |
| 33 | 105.5 | 105.4 | Diferencia muy pequeña |
| 27 | 130.8 | 130.9 | Diferencia muy pequeña |
| 22 | 174 | 174,9 | Diferencia muy pequeña |
| 10 | 339 | 339,8 | Diferencia muy pequeña |
| 5,7 | 633 | 633,2 | Diferencia muy pequeña |

Tabla C.2: Corriente en el Multimetro y en la Aplicación, dependiendo de una Resistencia específica.

**Nota:** Los anteriores datos fueron tomados con un voltaje de la batería de 3.786V

## C.2. FRECUENCIA DE MEDICIÓN:

Para corroborar que se cuenta con una frecuencia de medición buena, se establecen los siguientes pasos:

1. Conectar una resistencia fija entre los terminales 1 y 3.
2. Ejecutar la aplicación cliente y observar los datos de la tabla que esta despliega.
3. Repetir el anterior paso, variando el tiempo de petición de la aplicación, estos tiempos pueden ser: 1, 2, 5, 10, 15, 20 y 50 ms.
4. Llenar la tabla C.3

| Tiempo de Petición Seleccionado (ms) | Tiempo entre peticiones real (ms) | Retardo (ms) | Observaciones en cuanto al retardo |
|---|---|---|---|
| 1 | 15 ms – 16 ms | 14 ms – 15 ms | Varía entre estos 2 tiempos |
| 2 | 15 ms – 16 ms | 14 ms – 15 ms | Varía entre estos 2 tiempos |
| 5 | 15 ms – 16 ms | 14 ms – 15 ms | Varía entre estos 2 tiempos |
| 10 | 15 ms – 16 ms | 14 ms – 15 ms | Varía entre estos 2 tiempos |
| 15 | 15 ms – 16 ms | 0 ms - 1 ms | Retardo de 1 ms como máximo para estas peticiones |
| 20 | 20 ms – 21 ms | 0 ms - 1 ms | Retardo de 1 ms como máximo para estas peticiones |
| 50 | 50 ms – 51 ms | 0 ms - 1 ms | Retardo de 1 ms como máximo para estas peticiones |

Tabla C.3: Retardo entre el tiempo de petición seleccionado y el tiempo de petición real, con una resistencia fija como carga.

5. Cambiar la resistencia fija por el dispositivo móvil. Conectarlo en los terminales 1, 2 y 3.
6. Llenar la tabla C.4.

| Tiempo de Petición Seleccionada (ms) | Tiempo entre Peticiones real (ms) | Retardo (ms) | Observaciones en cuanto al retardo |
|---|---|---|---|
| 1 | 15 ms – 16 ms | 14 ms – 15 ms | Varía entre estos 2 tiempos |
| 2 | 15 ms – 16 ms | 14 ms – 15 ms | Varía entre estos 2 tiempos |
| 5 | 15 ms – 16 ms | 14 ms – 15 ms | Varía entre estos 2 tiempos |
| 10 | 15 ms – 16 ms | 14 ms – 15 ms | Varía entre estos 2 tiempos |
| 15 | 15 ms – 16 ms | 0 ms - 1 ms | Retardo de 1 ms como máximo para estas peticiones |
| 20 | 20 ms – 21 ms | 0 ms - 1 ms | Retardo de 1 ms como máximo para estas peticiones |
| 50 | 50 ms – 51 ms | 0 ms - 1 ms | Retardo de 1 ms como máximo para estas peticiones |

Tabla C.4: Retardo entre el tiempo de petición seleccionado y el tiempo de petición real, con el dispositivo conectado.

***Resultado:*** Al analizar los datos obtenidos se puede establecer que el tiempo de petición de 15 ms con una resistencia fija y de 15 ms con el dispositivo, son los mínimos tiempos en donde se encuentra como máximo un retardo de 1 ms, así se tenga un tiempo de petición mayor, y en donde no se pierden los datos. Por lo tanto, se escoge como tiempo mínimo de 15 ms para cada petición, por lo que la frecuencia de medición de 66.667 veces por segundo (66.667 Hz).

## C.3. GRAFICACIÓN EN TIEMPO REAL:

Para esta prueba se necesita remplazar la resistencia por el dispositivo, este se conectará en los terminales 1, 2 y 3. Posteriormente se ejecuta la Aplicación cliente. En el dispositivo móvil se ejecuta una aplicación llamada "Batería - Brillo Pantalla", la cual fue creada previamente, que permite establecer el brillo de la pantalla en máximo o en mínimo. Al ejecutar la aplicación sobre el dispositivo se comprueba en la Aplicación Cliente si las gráficas muestran en tiempo real las marcas que fueron creadas en la aplicación móvil. Se tiene como resultado la figura C.4.

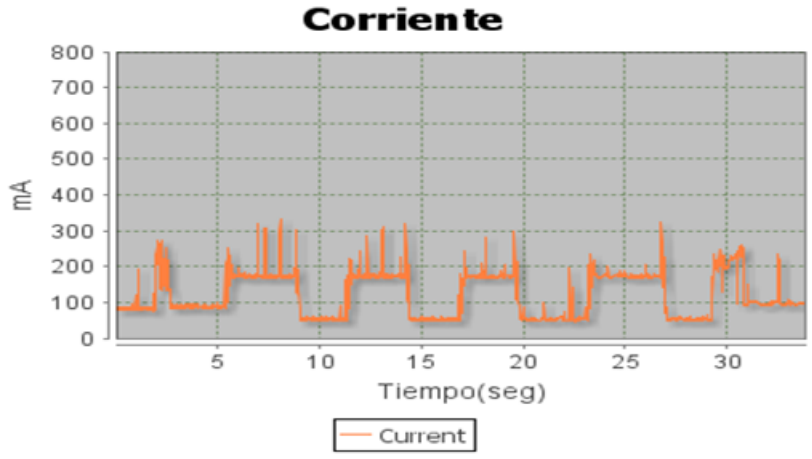


Figura C.4: Graficación en tiempo real de la corriente.

***Resultado:*** En la gráfica se pueden ver las diferentes marcas utilizadas para indicar las acciones que está

realizando la aplicación móvil.

- El primer incremento que se puede observar, aproximadamente en el 2sg, es el cual muestra el instante donde se selecciona y se ejecuta la aplicación en el Dispositivo.
- Entre el 6seg y el 8seg, aproximadamente, se utiliza la función establecer el brillo de la pantalla al máximo.
- Entre el 8seg y el 12seg, aproximadamente, se utiliza la función establecer el brillo de la pantalla al máximo.
- Estas dos funciones se repiten intercaladamente hasta el instante de 29seg, en donde se cierra la aplicación y el brillo de la pantalla se establece de nuevo a un nivel medio.

**Nota:** La prueba se realiza en el dispositivo Samsung Galaxy ACE, el cual se encontraba en estado Modo Avión y con el brillo de la pantalla en un nivel medio antes de iniciar la prueba.

## C.4. PERSISTENCIA DE LA INFORMACIÓN:

Para comprobar si existe una correcta persistencia de la información de la herramienta, se realiza la siguiente prueba: Al finalizar una medición en la Aplicación Cliente se selecciona la opción "Guardar", indicando el nombre de la carpeta y archivos a guardar. Posteriormente se analizan si los datos guardados corresponden a la medición que se obtuvo.

***Resultado:***Se guarda la información de una ejecución previa sobre el dispositivo, se guardan con el nombre de "prueba", cómo se puede observar en la figura C.5.

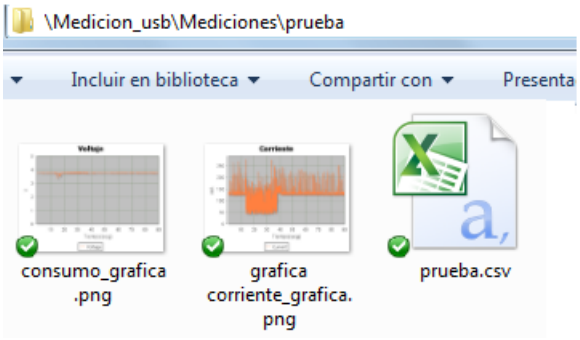


Figura C.5: Archivos guardados sobre carpeta contenedora.

Se abre el archivo "prueba.csv" el cual contienen los datos en una tabla en Excel, luego se corrobora si estos datos son los mismos que se pueden apreciar en las gráficas: "corriente_grafica.png" y "voltaje_grafica.png". Con los datos obtenidos se utiliza la opción de crear una gráfica en Excel, para posteriormente compararlos con las gráficas que ha creado la Aplicación cliente, cómo se puede observar a continuación:

- Voltaje:

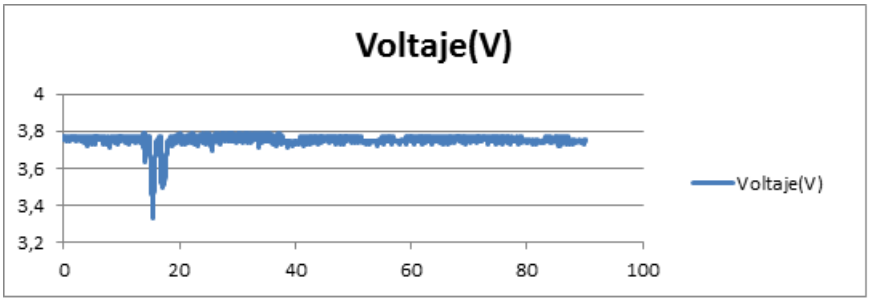


Figura C.6: Figura de los datos de voltaje obtenidos del archivo "prueba.csv".

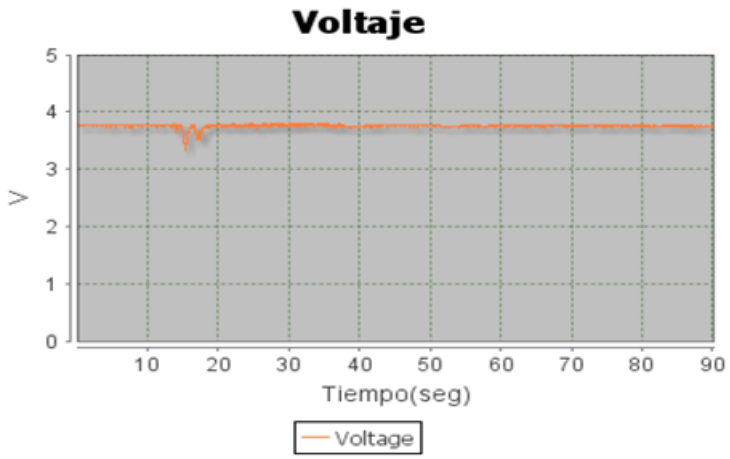


Figura C.7: Figura creada por la Aplicación Cliente del voltaje.

- Corriente:

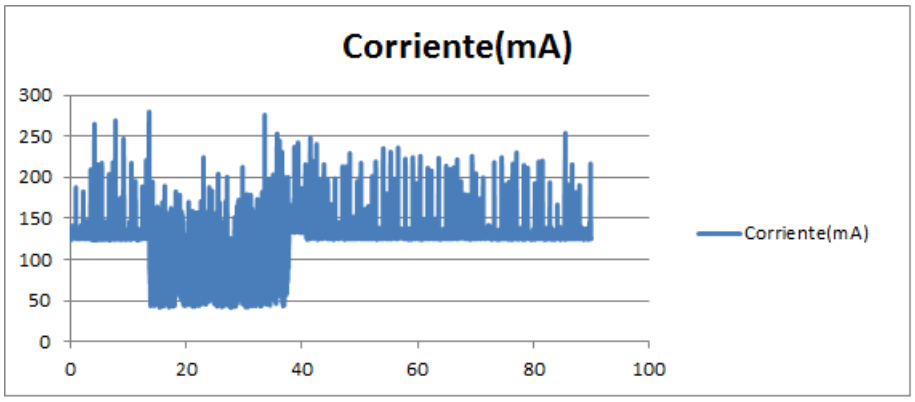


Figura C.8: Figura de los datos de corriente obtenidos del archivo "prueba.csv".

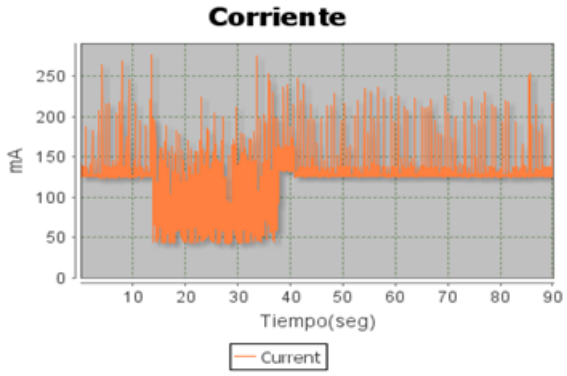


Figura C.9: Figura creada por la Aplicación Cliente de la corriente.

***Resultado:*** Tanto en voltaje como en corriente, se puede observar que los datos guardados en los archivos .csv son los mismo que se pueden observar cuando la aplicación se está ejecutando. Por lo que se tiene una correcta persistencia de la información.

**Dispositivos con los que se realizaron las mediciones:**

- **Osciloscopio:** Rigol DS1102E `http://www.rigolna.com/products/digital-oscilloscopes/ds1000e/ds1102e/`
- **Multímetro:** BEST BT60A
- **Fuente (voltaje alimentación):** GW INSTEK GPS 1850D `http://www.gwinstek.com/en/product/productdetail.aspx?pid=38&mid=53&id=128`.

# Anexo D

# Implementación de los Métodos para la sincronización de eventos

En este anexo se describe la implementación de los picos de consumo el cual es una métodos para la sincronicación de eventos descritos en el numeral 3.2. El anexo consta de 3 partes, en la primera se describe una aplicación donde se implementan los Picos de Consumo, en la segunda se explica la clase .java que se crea a partir de los Picos de Consumo y es utilizada en las aplicaciones que hacen parte de las pruebas, y en la tercera se despliegan las gráficas generadas por los Picos de Consumo en la Herramienta para la Medición de Consumo de Energía.

## D.1. Picos de Consumo

Para la utilización de picos de consumo se implementa una aplicación en Android llamada "Picos de Consumo" en la cual se genere 1 pico de consumo máximo y 1 pico de consumo mínimo. Estos picos se realizan sobre la pantalla, modificando el brillo que está pueda generar. Para lo anterior se hace uso de la librería WindowManager, la cual permite modificar de una forma sencilla y rápida el brillo de la pantalla. A continuación se explica en pasos el código utilizado:

1. Se obtiene el atributo del WindowsManager para el brillo de la pantalla:

   *WindowManager.LayoutParams nivel = getWindow().getAttributes();*

2. Se específica el valor del brillo en la pantalla, el valor debe ser de tipo float en un intervalo de 0.1 y 1.

   *nivel.screenBrightness = (float) 0.1 ;*

3. Se establece el nuevo nivel del brillo de la pantalla del dispositivo:

   *getWindow().setAttributes(nivel);*

En la figura D.1 se puede observar el código de la aplicación "Picos de Consumo".

```java
package com.brillo;

import com.medicionbateriaconbrillo.R;

public class PicosdeConsumoMain extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    protected void onStart() {
        // TODO Auto-generated method stub
        super.onStart();
    }

    @Override
    protected void onDestroy() {
        // TODO Auto-generated method stub
        super.onDestroy();
    }

    public void brillomax(View v) {
        WindowManager.LayoutParams nivel = getWindow().getAttributes(); //Obtención del atributo del brillo
        nivel.screenBrightness = 1; //Se especifica el nivel del brillo a cambiar
        getWindow().setAttributes(nivel); //Se establece el nuevo nivel del brillo
    }

    public void brillomin(View v) {
        WindowManager.LayoutParams nivel = getWindow().getAttributes(); //Obtención del atributo del brillo
        nivel.screenBrightness = (float) 0.1; //Se especifica el nivel del brillo a cambiar
        getWindow().setAttributes(nivel); //Se establece el nuevo nivel del brillo
    }
}
```

Figura D.1: Código de la aplicación "Picos de Consumo".

La figura D.2 contiene la interfaz de la aplicación “Picos de Consumo”.

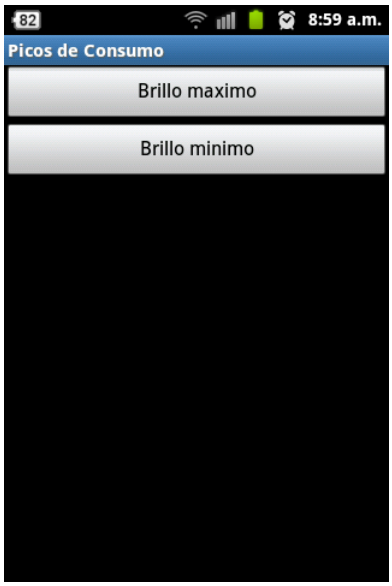


Figura D.2: Interfaz de la aplicación "Picos de Consumo".

## D.2. Clase “Marcas.java”

A partir de de los Picos de consumo nombrados y explicados en la sección anterior, se crea una clase .java encargada de manejar los Picos en todas las aplicaciones necesarias que se crean para las pruebas. En la

figura D.3 se puede observar el código de la clase “Marcas.java”, en donde se implementan los métodos explicados en la sección anterior.

```
import android.os.Handler;
import android.view.WindowManager;

public class Marcas {

    private WindowManager.LayoutParams nivel;
    private AudioMPMain actividad;
    boolean fin = false;

    public Marcas(AudioMPMain actividad) {
        this.actividad = actividad;
    }
    public void Marcainicio() {
        nivel = actividad.getWindow().getAttributes();
        nivel.screenBrightness = (float) 1;
        actividad.getWindow().setAttributes(nivel);

    }
    public void Marcaconsumo() {
        nivel = actividad.getWindow().getAttributes();
        nivel.screenBrightness = (float) 0.1;
        actividad.getWindow().setAttributes(nivel);
    }
    public void Retardo_iniciar() {

        new Handler().postDelayed(new Runnable() {
            public void run() {
                actividad.iniciar();
            }
        }, 3000);
    }
}
```

Figura D.3: Código de la clase "Marcas.java".

### D.2.1. Descripción de la clase “Marcas.java”:

La clase posee 3 métodos, los cuales se describen a continuación:

- Marcainicio(): Este método es el encargado de elevar el brillo de la pantalla al máximo cuando es seleccionado. Por lo general es utilizado al iniciar las aplicaciones.

- Marcaconsumo(): Este método es el encargado de disminuir el brillo de la pantalla al mínimo cuando es seleccionado. Por lo general es utilizado cuando se activan los procesos en las aplicaciones.

- Retardo:iniciar(): Este método utiliza un objeto de tipo Handler, el cual hace uso de la función Post-Delayed() que agrega a la cola de actividad un hilo que se ejecuta después de un retardo. Este retardo, que se ha definido en 3000ms, tiene como objetivo principal permitirle a las aplicaciones que se pueda estabilizar el nivel de consumo si existe un cambio en el nivel del brillo de pantalla.

## D.3. Caracterización de los Picos de Consumo

Al utilizar los Picos de Consumo sobre las aplicaciones a desarrollar en las pruebas se debe caracterizar el consumo que generan estos picos para poder diferenciarlos del consumo de las aplicaciones como tal, por lo que se evalúa la aplicación "Picos de Consumo" sobre la Herramienta para la Medición del Consumo de Energía. La medición sobre la herramienta se explica a continuación:

### D.3.1. Prueba de la aplicación "Picos de Consumo"

Para observar el funcionamiento de los Picos se define un escenario inicial donde se va a probar la aplicación:

**Escenario inicial:**

Para la medición del consumo de los picos de consumo en el dispositivo móvil se presenta el siguiente escenario:

- Modo avión.
- Pantalla con nivel de brillo medio.
- GPS, WIFI, auto sincronización y Rotación de pantalla desactivados.
- Aplicaciones en segundo plano desactivadas.

Posterior a la definición del escenario inicial se procede a hacer una prueba sobre la herramienta:

**Ejecución de la aplicación:**

Se ejecuta la aplicación y al mismo tiempo se observa los gráficos y datos desplegados por la Herramienta para la Medición del Consumo de Energía sobre la aplicación , obteniendo los siguientes resultados:

**Gráfica de Corriente:** En la figura D.4 se puede apreciar los siguientes eventos sobre el consumo:

1. Se selecciona el Botón "Brillo maximo", el cual aumenta el nivel del brillo al máximo.
2. Se estabiliza el nivel del brillo al máximo.
3. Se selecciona el Botón "Brillo minimo", el cual disminuye el nivel del brillo al mínimo.
4. Se estabiliza el nivel del brillo al mínimo.

**Nota:** En la figura D.4 se presentan unos picos externos a la aplicación que son atribuidos al procesamiento del dispositivo.

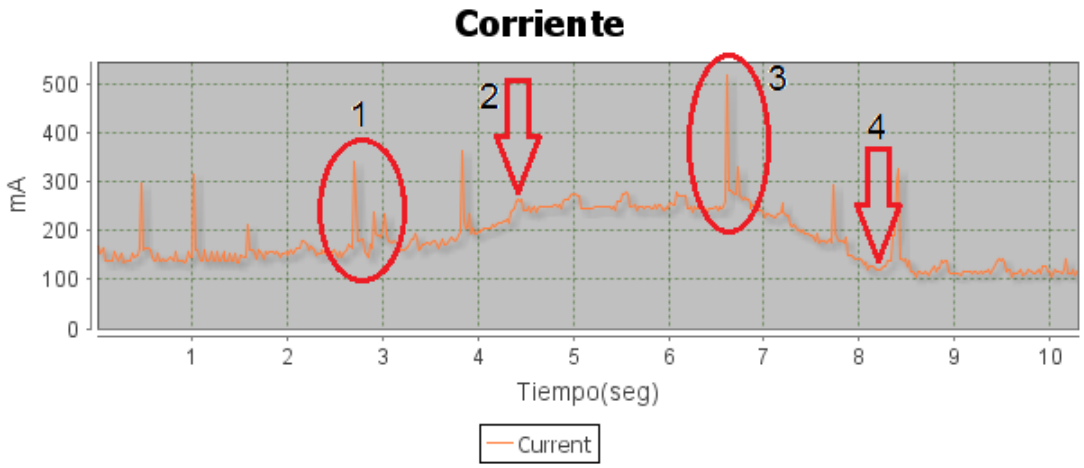


Figura D.4: Gráfica de corriente de la aplicación “Picos de Consumo” - Marcas.java.

# Anexo E

# Pruebas de medición

Este anexo presenta la descripción de las pruebas realizadas sobre los periféricos GPS, sistema de audio (auricular y altavoz), acelerómetro, brújula y teclado; utilizando la plantilla para el diseño de pruebas mostrada en la tabla 5.12.

## E.1. Pruebas GPS

| Prueba nº1 | |
|---|---|
| **Periférico** | GPS |
| **Práctica de programación a evaluar** | Variación de los parámetros minTime y minDistance en el registro de actualizaciones requestLocationUpdates(). |
| **Especificación del escenario** | • Modo avión.<br>• Pantalla con brillo mínimo.<br>• GPS, WIFI, auto sincronización y Rotación de pantalla desactivados.<br>• Aplicaciones en segundo plano desactivadas<br>• Lugar: exteriores<br>• Dispositivo móvil estático y en movimiento (a 60Km/h). |
| **Hipótesis** | • Aumentar los parámetros minTime y minDistance disminuye las actualizaciones y esto a su vez el consumo. |
| **Descripción software o aplicación** | Se modifica la aplicación "Localización" variando los parámetros minTime y minDistance. |

Tabla E.1: Prueba n 1 GPS.

## E.2. Pruebas audio

| Prueba nº1 | |
|---|---|
| **Periférico** | Audio |
| **Práctica de programación a evaluar** | Comparación entre géneros musicales utilizando el algoritmo inicial los géneros son: salsa, rock, reggaeton y clásica. |
| **Especificación del escenario** | • Modo avión.<br>• Pantalla con brillo mínimo.<br>• GPS, WIFI, auto sincronización y Rotación de pantalla desactivados.<br>• Aplicaciones en segundo plano desactivadas<br>• Nivel de sonido multimedia 75%.<br>• Archivo de audio mp3 estabilizado a 95db y duración de las canciones de 3 minutos.<br>• Reproducción sin audífonos.<br>• Canciones: Mosaico del Grupo niche, Back in black de AC/DC, Danza kuduro de Don Omar y Sinfonía nº5 de Beethoven. |
| **Hipótesis** | |
| **Descripción software o aplicación** | Se utiliza la aplicación "Audiomp" agregando 4 botones para cargar cada canción. |

Tabla E.2: Prueba numero 1 audio.

| Prueba nº2 | |
|---|---|
| **Periférico** | Audio |
| **Práctica de programación a evaluar** | Comparación entre una canción grabada en estudio y una en vivo. |
| **Especificación del escenario** | • Modo avión.<br>• Pantalla con brillo mínimo.<br>• GPS, WIFI, auto sincronización y Rotación de pantalla desactivados.<br>• Aplicaciones en segundo plano desactivadas<br>• Nivel de sonido multimedia 75%.<br>• Archivo de audio mp3 con duración de 3 minutos y con volumen que trae por defecto (89dB la estudio y 95,1dB la en vivo).<br>• Reproducción sin audífonos. canción: Back in black de AC/DC en estudio y en vivo. |
| **Hipótesis** | |
| **Descripción software o aplicación** | Se utiliza la anterior aplicación. |

Tabla E.3: Prueba numero 2 audio.

| Prueba nº3 | |
|---|---|
| **Periférico** | Audio |
| **Práctica de programación a evaluar** | Se evalúa si existe un menor consumo al utilizar el control de volumen como practica de programación usando diferentes niveles entre audífonos y altavoz |
| **Especificación del escenario** | • Modo avión.<br>• Pantalla con brillo mínimo.<br>• GPS, WIFI, auto sincronización y Rotación de pantalla desactivados.<br>• Aplicaciones en segundo plano desactivadas<br>• Nivel de sonido multimedia 75%.<br>• Archivo de audio mp3 estabilizado a 95db y duración de la canción de 3 minutos.<br>• Reproducción con y sin audífonos.<br>• Canción: Se selecciona la canción que presente el mayor consumo en la prueba 1. |
| **Hipótesis** | |
| **Descripción software o aplicación** | Se utiliza la aplicación "Audiomp" agregándole la opción de establecer el volumen de reproducción del celular en 70% cuando detecta que se han conectado los audífonos y en 100% si no han sido conectados (se está utilizando el altavoz). |

Tabla E.4: Prueba numero 3 audio.

## E.3. Pruebas acelerómetro

| Prueba nº1 | |
|---|---|
| **Periférico** | Acelerómetro |
| **Práctica de programación a evaluar** | Modificar de el tiempo de retardo entre mediciones del sensor en el registerListener(). |
| **Especificación del escenario** | • Modo avión.<br>• Pantalla con brillo mínimo.<br>• GPS, WIFI, auto sincronización y Rotación de pantalla desactivados.<br>• Aplicaciones en segundo plano desactivadas.<br>• Dispositivo con movimiento.<br>• Tasas de eventos del sensor de 200ms y 500ms. |
| **Hipótesis** | Disminuirá el consumo debido a que presentan menos eventos por segundo. |
| **Descripción software o aplicación** | Se modifica la aplicación "Acelerometro_1", cambiando el valor de el tiempo de retardo del sensor en el registerListener() a 200ms y 500ms. |

Tabla E.5: Prueba numero 1 acelerómetro.

| Prueba n°2 | |
|---|---|
| **Periférico** | Acelerómetro |
| **Práctica de programación a evaluar** | Se realiza la medición del consumo de la interfaz desactivando la referencia del SENSOR_SERVICE. |
| **Especificación del escenario** | • Modo avión.<br>• Pantalla con brillo mínimo.<br>• GPS, WIFI, auto sincronización y Rotación de pantalla desactivados.<br>• Aplicaciones en segundo plano desactivadas.<br>• Dispositivo con movimiento.<br>• Se realiza la prueba con y sin el registro. |
| **Hipótesis** | Obtener una referencia del SENSOR_SERVICE produce un consumo aun sin registrar el SensorEventListener. |
| **Descripción software o aplicación** | Se modifica la aplicación "Acelerometro_1", desactivando el registro del SENSOR " sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE)". |

Tabla E.6: Prueba numero 2 acelerómetro.

| Prueba n°3 | |
|---|---|
| **Periférico** | Acelerómetro |
| **Práctica de programación a evaluar** | Se condiciona el despliegue de los datos en pantalla a un tiempo determinado. |
| **Especificación del escenario** | • Modo avión.<br>• Pantalla con brillo mínimo.<br>• GPS, WIFI, auto sincronización y Rotación de pantalla desactivados.<br>• Aplicaciones en segundo plano desactivadas.<br>• Dispositivo con movimiento.<br>• Tasa de eventos del sensor en el registerListener() en 200ms.<br>• Despliegue de datos: 250ms, 500ms, 1000ms y sin despliegue. |
| **Hipótesis** | Muchos de los picos generados pueden ser debidos a los datos mostrados a gran velocidad en la interfaz |
| **Descripción software o aplicación** | Se modifica la aplicación "Acelerometro_1", dejando el valor del tiempo de peticiones del registerListener() a 200ms, además se utiliza el event.timestamp (que entrega el tiempo en nanosegundos entre un evento y el otro) para establecer un tiempo fijo y así desplegar los datos en la interfaz |

Tabla E.7: Prueba numero 3 acelerómetro.

## E.4. Pruebas brújula

| Prueba nº1 | |
|---|---|
| **Periférico** | Brújula |
| **Práctica de programación a evaluar** | Se realiza la medición del consumo de la interfaz desactivando la referencia del SENSOR_SERVICE. |
| **Especificación del escenario** | • Modo avión.<br>• Pantalla con brillo mínimo.<br>• GPS, WIFI, auto sincronización y Rotación de pantalla desactivados.<br>• Aplicaciones en segundo plano desactivadas.<br>• Dispositivo Galaxy Ace con movimiento.<br>• Se realiza la prueba con y sin el registro. |
| **Hipótesis** | Obtener una referencia del SENSOR_SERVICE produce un consumo aun sin registrar el SensorEventListener. |
| **Descripción software o aplicación** | Se modifica la aplicación "Brujula_inicial", desactivando el registro del SENSOR "sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE)". |

Tabla E.8: Prueba numero 1 brújula.

| Prueba nº2 | |
|---|---|
| **Periférico** | Brújula |
| **Práctica de programación a evaluar** | Modificar el tiempo de despliegue de los datos en pantalla utilizando el algoritmo 3. |
| **Especificación del escenario** | • Modo avión.<br>• Pantalla con brillo mínimo.<br>• GPS, WIFI, auto sincronización y Rotación de pantalla desactivados.<br>• Aplicaciones en segundo plano desactivadas.<br>• Dispositivo Galaxy ACE con movimiento.<br>• Se realiza la prueba con y sin el registro.<br>• Tasa de eventos del sensor en el registerListener() en 200ms.<br>• Despliegue de datos cada 250ms, 500ms y 1000ms y sin despliegue. |
| **Hipótesis** | Muchos de los picos generados pueden ser debidos a los datos mostrados a gran velocidad en la interfaz |
| **Descripción software o aplicación** | Se modifica la aplicación "Brujula_inicial", en el algoritmo 3 estableciendo el valor del tiempo de peticiones del registerListener() en 200ms, además se utiliza el event.timestamp (que entrega el tiempo en nanosegundos entre un evento y el otro) para establecer un tiempo fijo para desplegar los datos en la interfaz. |

Tabla E.9: Prueba numero 2 brújula.

## E.5. Pruebas teclado

| Prueba nº1 | |
|---|---|
| **Periférico** | Teclado |
| **Práctica de programación a evaluar** | Utilizar teclado predictivo permite escribir mas palabras presionando menos veces la pantalla, esto genera un menor consumo. Se escribe la frase: "Prueba sobre el teclado predictivo" |
| **Especificación del escenario** | • Modo avión.<br>• Pantalla con brillo mínimo.<br>• GPS, WIFI, auto sincronización y Rotación de pantalla desactivados.<br>• Aplicaciones en segundo plano desactivadas.<br>• Teclado predictivo activado. |
| **Hipótesis** | Al utilizar un teclado predictivo se generarían menos picos que en el QWERTY, lo que significa que se genera menos consumo. |
| **Descripción software o aplicación** | Se utiliza la misma aplicación "EscenarioTeclado". |

Tabla E.10: Prueba numero 1 teclado.

| Prueba nº2 | |
|---|---|
| **Periférico** | Teclado |
| **Práctica de programación a evaluar** | Se utiliza el teclado SwiftKey, el cual es descargado del google Play, el Swype versión beta (descargado de beta.swype.com). Se realiza la comparación del consumo de estos 2 teclados con el predictivo del celular. Se utiliza la misma frase de la prueba 1. |
| **Especificación del escenario** | • Modo avión.<br>• Pantalla con brillo mínimo.<br>• GPS, WIFI, auto sincronización y Rotación de pantalla desactivados.<br>• Aplicaciones en segundo plano desactivadas.<br>• Teclados SwiftKey y Swype (beta). |
| **Hipótesis** | • El Swype produce un mayor consumo por su método de escritura ya que se mantiene un mayor tiempo en contacto la pantalla para escribir las palabras. es un teclado.<br>• Al ser SwiftKey 3 un teclado con un uso muy grande en los usuarios de android, deben presentar un menor consumo comparado con el teclado predictivo que trae por defecto el dispositivo móvil. |
| **Descripción software o aplicación** | Se utiliza la misma aplicación EscenarioTeclado. |

Tabla E.11: Prueba numero 2 teclado.