FACULTAD DE INGENIERÍA Y CIENCIAS AGROPECUARIAS

ANÁLISIS DE ESTRATEGIAS DE EFICIENCIA ENERGÉTICA PARA DISPOSITIVOS MÓVILES, BASADO EN EL CONCEPTO DE GREEN IT

Trabajo de titulación presentado en conformidad a los requisitos para obtener el título de Ingeniero en Redes y Telecomunicaciones

Profesor Guía

Mdhd. Héctor Fernando Chinchero Villacís

Autor

Carlos Alberto Fernández Sangucho

Año

2016

DECLARACIÓN PROFESOR GUÍA

“Declaro haber dirigido este trabajo a través de reuniones periódicas con el estudiante, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación.

_____

Héctor Fernando Chinchero Villacís

Master en Domótica

CI: 1715451330

DECLARACIÓN DE AUTORÍA DEL ESTUDIANTE

Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes.

_____

Carlos Alberto Fernández Sangucho

CI: 1711961696

AGRADECIMIENTOS

A Dios por todas las bendiciones que he recibido en mi vida. A mis padres por su esfuerzo y sacrificio, muchos de mis logros se los debo a ustedes.

A mis queridos hermanos y tíos, por su apoyo incondicional, amor y cariño.

DEDICATORIA

A mis Padres, por creer en mí y ser un ejemplo de vida, gracias por su apoyo y por enseñarme a enfrentar con valentía los problemas por más duros que estos sean.

A mi hijo que con sus palabras de aliento me motivan y comprometen a ser mejor cada día.

RESUMEN

Las avanzadas características y funcionalidades que tienen los *Smartphones* han venido a desplazar en gran parte al teléfono móvil convencional, es por eso, que existe una creciente demanda de teléfonos inteligentes en todas partes del mundo. Sin embargo uno de los problemas que enfrentan estos dispositivos es el poder garantizar una autonomía en la duración de su batería, independientemente de su sistema operativo y de las aplicaciones que se utilicen.

Con referencia a lo anterior, hay diferentes baterías en el mercado por lo que su comparación es compleja respecto a su duración, no obstante varios de estos dispositivos manejan distintas configuraciones para el ahorro de energía, sin embargo sigue siendo insuficiente, por eso se requiere establecer otros mecanismos que permitan optimar su uso.

Sobre la base de las consideraciones descritas, se busca identificar los componentes de hardware que descargan la batería de un *Smartphone.* Para conseguir este propósito, se plantea una metodología, que permita medir el consumo de energía en un dispositivo móvil con sistema operativo Android, tomando en cuenta las prestaciones que ofrece el código abierto y su entorno de desarrollo de aplicaciones en Android Studio.

Este análisis se realiza a los componentes de Bluetooth y GPS que permitirán conocer cuál de estos componentes consumen mayor energía y formular estrategias de eficiencia energética.

ABSTRACT

The smartphones advanced features and functionalities have come to largely displace conventional mobile phone. For this reason, there is a growing demand for smart phones all over the world. However, one of the problems faced by these devices is to ensure autonomy in battery life, regardless of their operating system and applications used.

With reference to the previous, there are different batteries on the market, so the comparison is complex regarding its duration. Nevertheless, many of these devices handle different configurations for energy saving, but it is still insufficient. It is necessary to establish other mechanisms to optimize their use.

Based on the considerations described, it seeks to identify the hardware components that drain the battery of a Smartphone. To achieve this purpose, It is a methodology to measure power consumption on a mobile device with Android operating system, taking into account the benefits offered by the open source and application development environment arises.

This analysis is performed to Bluetooth and GPS components that allow you to know which of these components consume more energy and make energy efficiency strategies.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En un mundo moderno las tecnologías de información y comunicación tienen un rol trascendental en el desarrollo de las empresas y las personas. Tanto es así, que con la llegada de los dispositivos móviles (*Smartphones*), existe una gran necesidad de mantenernos comunicados unos con otros. En ese contexto, las grandes empresas de telefonía están innovando y haciendo esfuerzos por brindar equipos y servicios acorde a la demanda actual. Como consecuencia de esto, la mayoría de estos dispositivos inteligentes se usan con frecuencia en actividades de consulta de información, entretenimiento o también como oficinas móviles en ambientes corporativos gracias a que son multitarea.

Con referencia a lo anterior, el informe de Cisco VNI Mobile 2015-2010, señala que “el incremento de dispositivos inteligentes, vídeo móvil y redes 4G multiplicará por 8 el tráfico global de datos móviles en los próximos cinco años” (Cisco, 2016). De igual manera menciona que los usuarios de dispositivos móviles, incluidas las "*phablets*" (un híbrido entre teléfono y tableta), llegarán en ese año a los 5.400 millones, un 70% de la población estimada para ese año.

Es evidente entonces, que estas cifras son indicadores de que la penetración de *Smartphones* y dispositivos móviles está creciendo exponencialmente, por lo tanto, el consumo de energía aumentará de igual forma en estos dispositivos. Si bien hay mejoras en la tecnología de la batería, las mismas se han retrasado respecto al avance continuo de los *Smartphones*.

En las mismas circunstancias el hardware, aplicaciones y diferentes sistemas operativos crecerá en gran medida, consecuentemente, los fabricantes de dispositivos móviles tienen un gran desafío, en formular alternativas de solución para reducir el consumo de energía y prolongar la vida útil de las baterías, en beneficio de la sociedad y la protección del medio ambiente.

Actualmente los *Smartphones* llevan consigo una serie de componentes integrados como por ejemplo: la placa base, la antena Wi-Fi, el Bluetooth, la

pantalla, el teclado. Asimismo funcionan con todas las tecnologías y redes actuales como GSM, 3G, 4G y LTE; súmese a esto, el sinnúmero de funciones y aplicaciones para diversión o para trabajo, todo esto sin duda demanda un mayor uso de energía, provocando que la batería se descargue rápidamente.

Hay varios artículos, blogs, páginas de fabricantes, aplicaciones propias, dedicados a brindar consejos para optimizar la batería de los dispositivos móviles, por citar algunos de ellos, recomiendan disminuir el brillo de la pantalla, desactivar el Bluetooth, GPS, el uso de fondos de pantalla, entre otras opciones.

Sin embargo no lo es todo, estos dispositivos siempre están ejecutando actividades en segundo plano que consumen igual o más energía, por eso es necesario que a través del mismo dispositivo se reduzcan los niveles de consumo.

Respecto a las mejoras en la administración de energía, el sistema operativo Android es uno de ellos que tiene un rol más activo respecto a este tema. Este sistema que es basado en Linux, tiene un *Middleware* que es “un software que permite conectar componentes softwares o aplicaciones.

El mismo consiste en un conjunto de servicios que permiten que múltiples procesos corriendo en una o varias máquinas interactúen de un lado a otro de la red” (Varela, 2007).

De acuerdo a los razonamientos que se han venido realizando, el actual trabajo está enfocado en analizar los componentes de Bluetooth y GPS bajo una metodología de medición a través del desarrollo de una aplicación en Android Studio, que permita medir el consumo y la potencia de energía consumida por estos componentes en un dispositivo móvil.

Los resultados que se obtengan permitirán conocer cuál de estos componentes consumen mayor energía y formular estrategias de eficiencia energética.

Objetivos

General

Analizar el consumo de energía en los principales componentes de hardware de un dispositivo móvil basados en Android, bajo escenarios de uso reales, a través del desarrollo de una aplicación y una metodología de medición de potencia para la formulación de un modelo de consumo energético.

Específicos

- Proponer una metodología de medición de potencia de consumo de dos de los principales componentes (Bluetooth, GPS) de un dispositivo móvil, mediante el desarrollo de una aplicación en Android Studio.
- Realizar mediciones de corriente y voltaje de las potencias consumidas en la batería, considerando los estados que tienen Bluetooth y GPS.
- Elaborar un modelo de consumo energético global para el dispositivo móvil, partiendo de los resultados obtenidos en la metodología empleada en los niveles de software respectivamente.
- Evaluar el modelo de consumo energético con el propósito de desarrollar métodos de administración y de ahorro de energía.

Alcance

El presente proyecto parte del estudio del consumo de energía de los principales componentes de Bluetooth y GPS para dispositivos móviles basados en Android, seguido del desarrollo de una metodología de medición de potencia y corriente a través del desarrollo de una aplicación monitoreo de consumo de energía; para ello se elaborará un modelo de consumo energético mediante el empleo software de Android Studio con el fin de analizar las muestras adquiridas en las mediciones de potencia y corriente de un Smartphone, finalmente se evaluará dicho modelo con el propósito de desarrollar métodos de administración y de ahorro de energía.

1. CAPÍTULO I. MARCO TEÓRICO

1.1. Dispositivos móviles

Una definición exacta “de que es y no es” un dispositivo móvil, resulta complejo debido a las características, prestaciones y al gran despliegue que han tenido en los últimos años. Sin embargo, para Ana y Gader (2011) un dispositivo móvil son aparatos electrónicos que cumplen con características específicas como, capacidad de conectarse a una red, disponer de memoria y procesamiento, poseer mecanismos de entrada y salida para interactuar con el usuario y el uso de batería como fuente de energía.

1.1.1. Clasificación

Según los niveles de funcionalidad existen varias clasificaciones para los dispositivos móviles. La Tabla 1 describe de manera general estas funcionalidades.

Tabla 1. Clasificación de los dispositivos móviles

Dispositivos	Funcionalidad	Ejemplos
Comunicación	Ofrece principalmente comunicación telefónica, envío de mensajes SMS (*Short Message Service*), MMS (*Multimedia Application Protocol*), o acceso WAP (Wireless *Application Protocol*), etcétera.	- Teléfonos móviles, tradicionales, - *Smartphones*, - Pantalla táctil, - Conexión internet, - Instalación y ejecución de programas.
Computación	Brinda un mayor procesamiento de datos, su pantalla y teclado son similares a una computadora de escritorio.	- PDA, - *Notebook*, - *Netbook*,
Multimedia	Son de menor tamaño y se utiliza como reproductor de audio, video e imágenes en diferentes formatos.	- MP3, MP4, - DVD, - Portátiles, - *eBooks*.
Grabador de multimedia	Tienen como función la grabación de datos en determinado formato digital, principalmente audio y video.	- Cámaras fotográficas digitales, - Cámaras de video digital
Consola portátil	Facilita al usuario una plataforma de juego.	- PSP (PlayStation Portable)

Adaptado de (Ana y Gader, 2011, pp. 18-20)

1.1.2. Tipos de dispositivos

Considerando la gran cantidad de dispositivos móviles que existe en el mercado y las diferentes funcionalidades que ofrecen, las empresas T38 y DuPont Global Mobility Innovation Team (2005) establecen un tipo de clasificación y proponen los siguientes estándares para dispositivos móviles.

1. **Dispositivo móvil de datos limitados (*Limited data mobile device*)**.- Se relaciona con los teléfonos móviles clásicos, se caracteriza por tener la pantalla pequeña y por ofrecer servicios básicos como el SMS y acceso WAP (Martinez, 2011).

2. **Dispositivo móvil de datos básicos (*Basic data mobile device*)**.- Este estándar tiene relación con teléfonos inteligentes "*Smartphone*", debido a las pantallas medianas que poseen y la facilidad de visualizar e-mail, la lista de direcciones (Martinez, 2011).

3. **Dispositivo móvil de datos mejorados (*Enhanced data mobile device*)**.- Brinda las mismas características del segundo estándar, adicional ofrece al usuario aplicaciones nativas y corporativas como Microsoft Office Mobile (Martinez, 2011).

Como se puede apreciar en la Figura 1, se presenta la clasificación de los dispositivos móviles conocidos también como "*Handheld*", según su tamaño, capacidad, conexión a la red, memoria entre otras de sus funcionalidades.

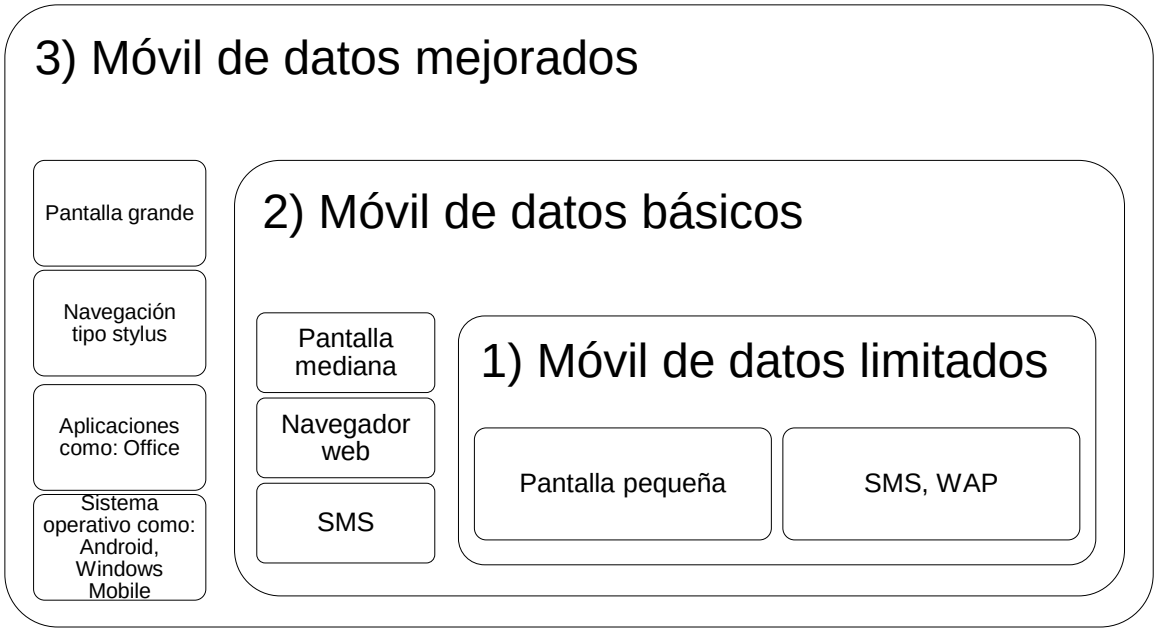


Figura 1. Clasificación de dispositivos móviles

Adaptado de (Martínez, 2011, p. 7)

1.1.3. Smartphone

Ramírez (2013) señala un *Smartphone* es un dispositivo electrónico que tiene el funcionamiento de un celular o teléfono móvil con características de un computador personal. Así mismo el hardware y software tienen como fin realizar tareas que exigen mayores características como por ejemplo procesamiento, memoria y capacidad.

Estas características son definidas a través del sistema operativo, el cual se encarga de administrar los recursos del equipo, proveer seguridad, optimizar las funcionalidades y conectividad hacia Internet, utilizando varias tecnologías y estándares de comunicaciones inalámbricas como, GPRS, Wi-Fi, Infrarrojo, Bluetooth, WAP, posicionamiento global GPS, etcétera; entre otras características administra el correo electrónico, los contactos de la agenda, se sincroniza con otros equipos, permite la instalación de aplicaciones y juegos, además graba videos y toma fotos (Ramirez, 2013).

1.1.4. Sistemas operativos para dispositivos móviles

Los sistemas operativos para móviles, son más simples y menos robustos que los desarrollados para equipos de escritorio; están orientados a la movilidad, conectividad inalámbrica, procesamiento, administración y almacenamiento.

Una de las características es la administración del hardware y software, sin embargo se ven limitados por la autonomía de carga y tiempo de uso de la batería.

Empresas como Samsung, Apple, LG, HTC, Huawei, Sony, Microsoft, entre otras, desarrollan sistemas operativos acorde a la demanda del mercado móvil; entre los más conocidos tenemos Android, iOS, Windows Phone y BlackBerry OS, siendo los dos primeros sistemas operativos los que lideran el mercado de la telefonía móvil. La Tabla 2 indica los sistemas operativos y sus respectivas empresas.

Tabla 2. Cuadro de sistemas operativos para móviles

Sistema operativo	Empresa	Observación
Android	Google	Usado por HTC, LG, Motorola, Samsung
iOS	Apple	Utilizan los iPhone, iPad
Windows Phone	Microsoft	Usado por Smartphone de gama alta de Nokia
BlackBerry OS	RIM	Sistema propio de BlackBerry
Symbian	Nokia	Es de propiedad de Nokia (Están por dar de baja)
Ubuntu OS	Canonical	Sistema operativo futuro
Firefox OS	Mozilla	Sistema operativo futuro
Tizen	Samsung	Sistema operativo futuro

Adaptado de (UNAD, s.f.)

1.1.5. Rankin de sistemas operativos

La demanda del mercado móvil, posiciona como líder al sistema operativo Android de Google, desplazando al segundo lugar al iOS de Apple su principal competidor. La Figura 2 muestra que Android ocupa el primer lugar con un 60.99% versus el 31,76% de iOS. El 7,25% se distribuye entre los nueve sistemas operativos restantes.

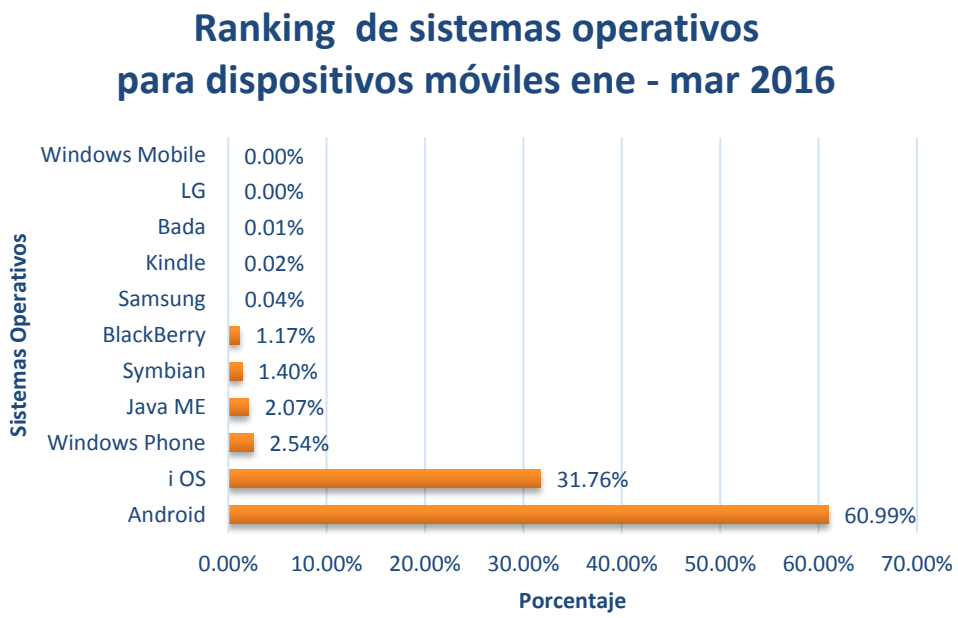


Figura 2. Ranking de sistemas operativos más usados

Adaptado de (Mobile/Tablet System Market Share, 2016)

Los sistemas móviles actuales tienen varias características, entre las que se destacan (UNAD, s.f.).

- Posee un Kernel Unificado
- Construido por capas
- Es multiproceso y multitarea.
- Soporta diferentes tipos de pantallas y múltiples lenguajes
- Permite conectividad inalámbrica
- Administración del Hardware y aplicaciones
- Navegación Web
- Multiusuario

1.1.6. Aplicaciones que miden el consumo de energía

En Google Play Store, existe una serie de aplicaciones gratuitas o pagadas que detallan de forma general el consumo de batería en dispositivos inteligentes, algunos de ellos son configurables y fáciles de usar. Sin embargo la mayoría de aplicaciones gratuitas entregan información de todo el dispositivo móvil sin discriminar cuales de los componentes consumen más energía, siendo esto una limitante a la hora de realizar un análisis en su operación.

Por ejemplo la Figura 3, muestra la pantalla principal de la aplicación *Battery Monitor*. Esta herramienta monitorea la batería y emite un reporte a nivel de todo el dispositivo móvil.

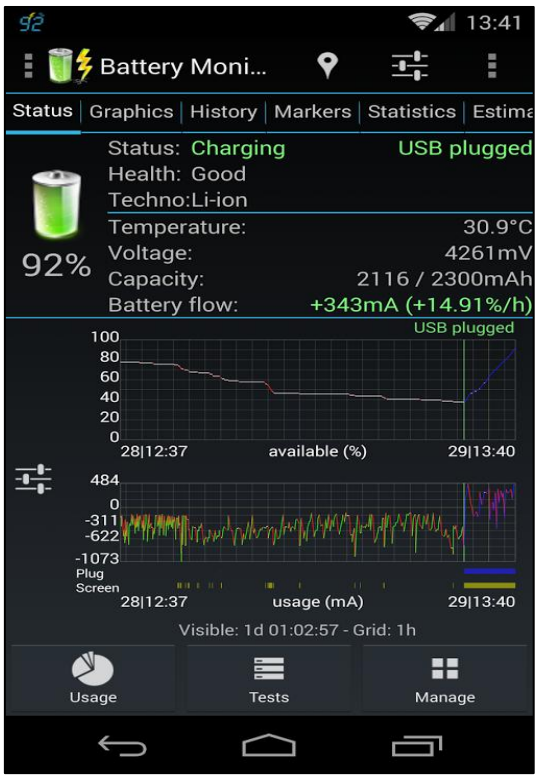


Figura 3. Consumo de energía de la aplicación *Battery Monitor*

Tomado de (Google Play Battery Monitor, 2016)

Considerando el número de plataformas y aplicaciones con funcionalidades similares, una de las preocupaciones que se tiene es la autonomía de los dispositivos móviles, en ese contexto, varios investigadores han realizado

trabajos relacionados con el consumo de energía desde el punto de vista de la eficiencia energética.

Los investigadores Metri, G., Shi, W., y Brockmeyer, M. (2015) en un estudio comparativo sobre la eficiencia energética en dispositivos y aplicaciones móviles señala: es una tarea difícil debido a la falta de herramientas y tecnologías apropiadas y las diferentes categorías de aplicaciones en las plataformas más populares, esto sin duda arrojará posibles errores de medición.

Otra de las aplicaciones es *Trepn profiler* (ver Figura 4), permite medir el rendimiento y el consumo de energía de aplicaciones Android en dispositivos con procesadores *SnapDragon* desarrollados por la empresa Qualcomm Metri, (Metri, G., Shi, W., y Brockmeyer, M. 2015)

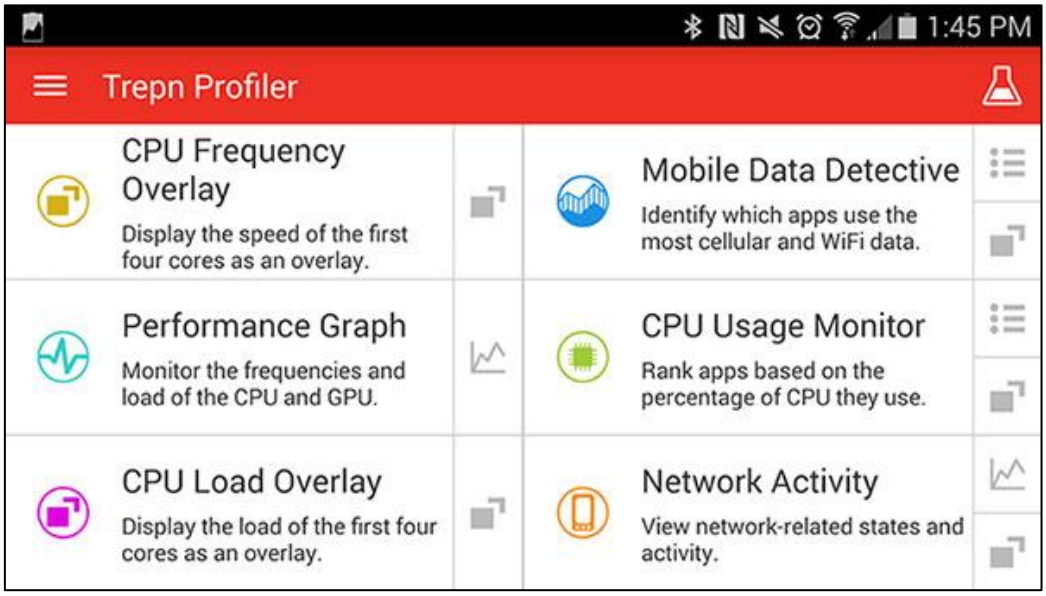


Figura 4. Aplicación Trepn profiler

Tomado de (Google Play, 2016)

Android ofrece al usuario una herramienta de monitoreo, sin embargo es limitada a la hora de obtener el consumo de potencia por componente.

Figura 5. Herramienta de monitorización de la batería en Android

Tomado de (*Smartphone* Samsung Gran Dúos, 2016)

1.2. ¿Qué es Android?

Es la plataforma líder de software de Google, diseñado para dispositivos móviles como *tablets*, *netbooks*, reproductores de música, entre otros; esta plataforma se basa en Linux, incluye el sistema operativo con un middleware, librerías orientadas al desarrollo y una serie de aplicaciones móviles.

La plataforma en mención tiene una base fundamental robusta que está conformada por el *Android Open Source Project* (AOSP), *Google Mobile Services* (Google Maps, Google Play, Google Now), *Google Apps* (Google Drive, Gmail, Google Keep) y el Android Studio, como nuevo IDE para desarrollar aplicaciones móviles (Pastor, 2014).

Varios analistas de tecnología relacionados con los dispositivos móviles, entre ellos *Gartner*, en su último informe sobre el estado del mercado de los teléfonos móviles del año 2015, señala: que los dispositivos Android son los más dominantes del mercado, ocupando un 82.2% de la cuota de mercado de

dispositivos móviles, seguido por Apple con un 14.6% y finalmente Microsoft con el 2.5% (Cisco, 2016).

Tabla 3. Mercado de teléfonos móviles según

Cuotas de mercado de los sistemas operativos para *Smartphones*				
Sistema operativo	**2015 Unidades**	**2015 Cuota de mercado (%)**	**2015 Unidades**	**2014 Cuota de mercado (%)**
Android	271.000	82,2%	243.484	83,8%
iOS	48.086	14,6%	35.345	12,2%
Windows	8.198	2,5%	8.095	2,8%
BlackBerry	1.153	0,3%	2.044	0,7%
Otros	1.229	0,4%	1.416,8	0,5%
Total	329.666	100%	290.384,8	100,0%

Adaptado de (Informe de *Gartner*, 2015)

1.2.1. Características técnicas

A continuación se detallan las características principales de la plataforma Android (Sistema Android, 2012).

- **Framework de aplicaciones**: Es un conjunto de herramientas para el desarrollo de aplicaciones para dispositivos móviles.
- **Navegador integrado**: Utiliza el motor de navegación web de código abierto open *Source Webkit*.
- **SQlite**: Es el motor de base de datos para dispositivos móviles, y de almacenamiento estructurado para la información.
- **Multimedia**: Soporta formatos de imágenes, audio y video como por ejemplo: H.264, MPEG4, MP3, JPG, entre otros.
- **Dalvik**: Es la máquina virtual especializada para Java**,** diseñada para dispositivos móviles con procesos y memoria limitados.
- **Conectividad**: Esta plataforma soporta tecnologías como GSM/EDGE, 3G, Wi-Fi, Bluetooth, HSPA+, LTE.
- **Soporte hardware adicional**: Cámara de fotos y video, GPS, sensores de luz, brújula, acelerómetro, pantalla táctil.

- **Entorno de desarrollo**: Se integra con Eclipse en sus diferentes versiones, posee herramientas de análisis para el rendimiento del software y depuración de la memoria.

1.2.2. Arquitectura

Android es el sistema operativo creado para ser independiente de cualquier tipo de arquitectura de hardware en los dispositivos móviles. Esta característica es importante para los fabricantes y desarrolladores. La arquitectura interna del sistema operativo se conforma por cuatro capas o niveles según la Figura 6.

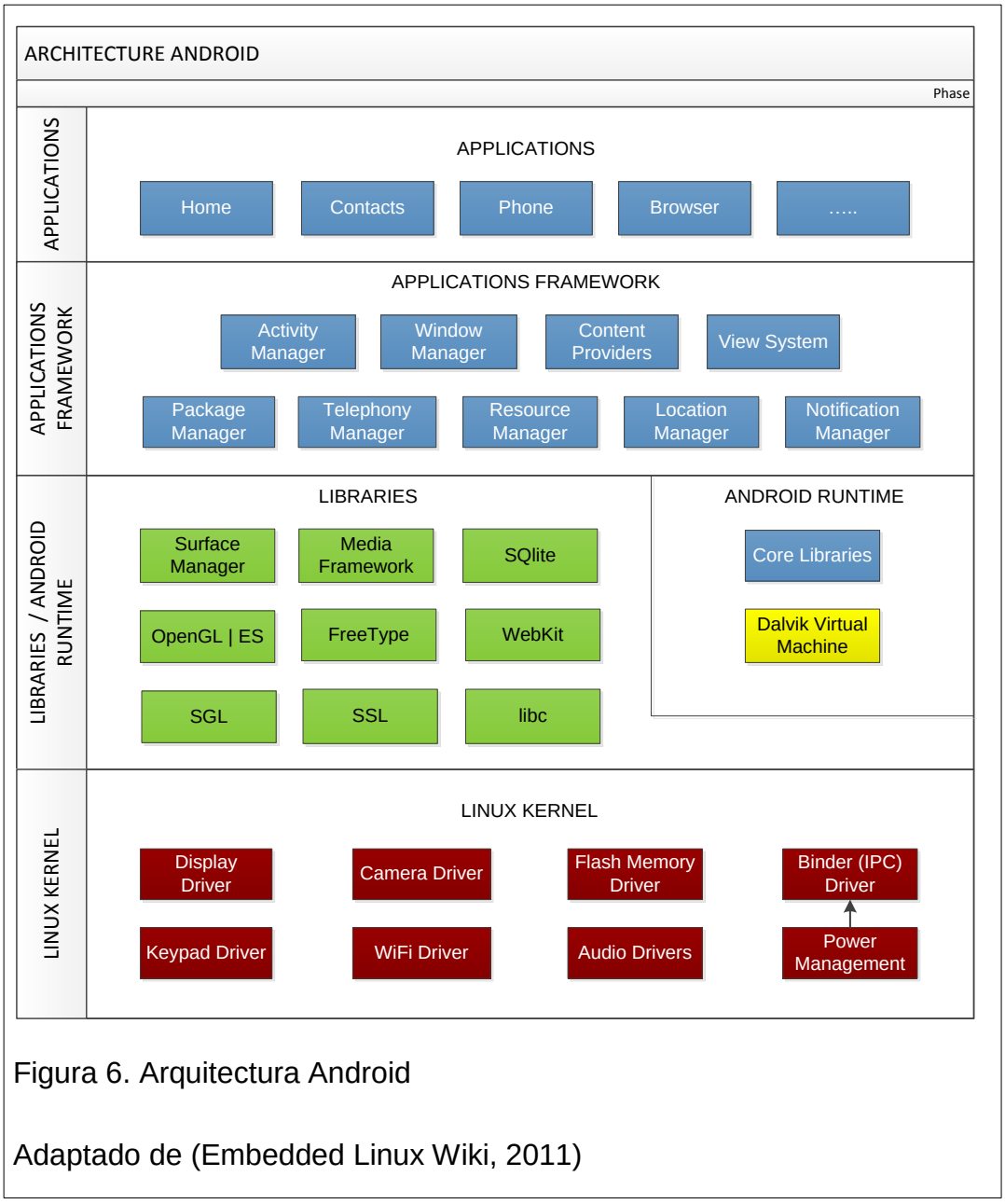


Figura 6. Arquitectura Android

Adaptado de (Embedded Linux Wiki, 2011)

Este tipo de estructura se denomina pila, describe los niveles o capas de la arquitectura empleada por la plataforma Android

- Aplicaciones
- *Framework* de aplicaciones
- Librerías
- *Runtime* de Android
- Núcleo Linux

Según la arquitectura Android descrita en la Figura 6, esta se encuentra construida sobre el Kernel de Linux. Seguido de una capa de Librerías relacionadas con una estructura para la administración durante el tiempo de ejecución.

El siguiente nivel le corresponde al Framework que tiene como función la construcción de aplicaciones y finalmente la capa de Aplicaciones.

A continuación se presenta el rol que tiene de cada una de las capas en dicha arquitectura (Universidad Carlos III de Madrid, 2009, p. 2).

- **Aplicaciones (*Aplications*).-** Todas las aplicaciones de esta capa usan Java como lenguaje de programación, API y librerías de los otros niveles, incluye varias aplicaciones como por ejemplo el navegador, calendario, etcétera.

- **Framework de Aplicaciones (*Applications framework*).-** Se compone de varias herramientas para el desarrollo de aplicaciones, permite al programador acceder al código fuente base, con el fin de reutilizar el código para mejorar y ampliar los nuevos componentes de este nivel.

  Las API más importantes son las siguientes (Universidad Carlos III de Madrid, 2009, p.3).

  - ***Activity Manager**:* Se encarga de gestionar las aplicaciones, así como el ciclo de vida de las API.

- ***Window Manager:*** Trabaja en conjunto con la librería *Surface Manager* para administrar múltiples ventanas y aplicaciones
- ***Telephone Manager:*** Administra las funcionalidades del teléfono como llamadas, mensajes, entre otros, mediante un conjunto de API.
- ***Content Provider:*** Su función principal es compartir información entre aplicaciones del sistema, por ejemplo, lista de contactos, SMS.
- ***View System:*** Permite construir interfaces de usuario (GUI), como botones, "check-boxes", entrada de textos, etc. Maneja una serie de vistas estándar relacionadas con el tamaño y posición.
- ***Location Manager:*** Facilita a las aplicaciones para obtener información de actualizaciones periódicas de posicionamiento y localización del dispositivo.
- ***Notification Manager:*** Gestiona las diferentes comunicaciones con el usuario; estas pueden tomar formas diferentes como el parpadeo de los LEDs en el dispositivo ante un mensaje recibido o mediante un sonido o vibración de una llamada, etcétera.
- ***XMPP Service:*** Protocolo de envío y recepción para mensajería instantánea.

- **Librerías (*Libraries*).-** Android incluye en su base de datos un set de librerías C/C++, que son expuestas a todos los desarrolladores a través del framework de las aplicaciones Android System C library, librerías de medios, librerías de gráficos, 3D, Sqlite, etc., (Universidad Carlos III de Madrid, s.f.). Entre las librerías más importantes se encuentran las siguientes:
  - ***Surface Manager****:* Permite que las imágenes 2D y 3D se muestren en pantalla. Gestiona los diversos efectos como transparencias y animaciones.
  - ***OpenGL/SL y SGL****:* Ambas librerías son el motor gráfico para 2D y 3D, siendo este último el que utiliza API para el desarrollo de aplicaciones en sus distintas versiones.

- ***Librería Media Libraries****:* Para el contenido multimedia (audio, video e imágenes animadas/estáticas) esta librería provee de los códecs correspondientes.
- ***FreeType:*** Contiene una biblioteca de las distintas fuentes tipográficas y su distribución es mediante licencias de código abierto.
- ***SSL:*** Protocolo criptográfico que ofrece comunicaciones seguras.
- ***SQLite:*** Motor de base de datos para dispositivos móviles, para almacenamiento estructurado de información.
- ***WebKit:*** Proporciona un motor e interactúa con las aplicaciones del navegador de Android.
- ***libc****:* Contiene las bibliotecas estándar más importantes del sistema, entre ellas la biblioteca matemática y C los programas que permite el correcto funcionamiento.

- **Runtime de Android (Android runtime). -** Se basa en la máquina virtual *Dalvik* considerando las limitaciones en procesador y memoria que tiene la máquina virtual de Java de los dispositivos de Android. *Dalvik* optimiza los recursos y faculta al núcleo de Linux varias funciones como por ejemplo el manejo de la memoria de bajo nivel.

  Dalvik en su funcionamiento no realiza ningún cambio durante el proceso de compilación, al final entrega un ejecutable el cual contiene los archivos **.class de java**. Este proceso se encuentra descrito en la Figura 7; se realiza en primera instancia a través de la generación del código fuente (archivos .java), el cual es traducido por el Java Compiler (javac), para luego obtener un fichero tipo byte code (archivos .class). Al final la aplicación es ejecutada por la máquina virtual de Java (JVM) (Revelo, 2014).

  Dalvik realiza esta misma actividad, sin embargo, incluye un proceso a través del compilador Dex, que consiste en traducir “los byte codes de java a un estilo de byte codes nativos que serán convertidos a un ejecutable .dex. Finalmente este archivo es ejecutado por una instancia de Dalvik VM” (Revelo, 2014).

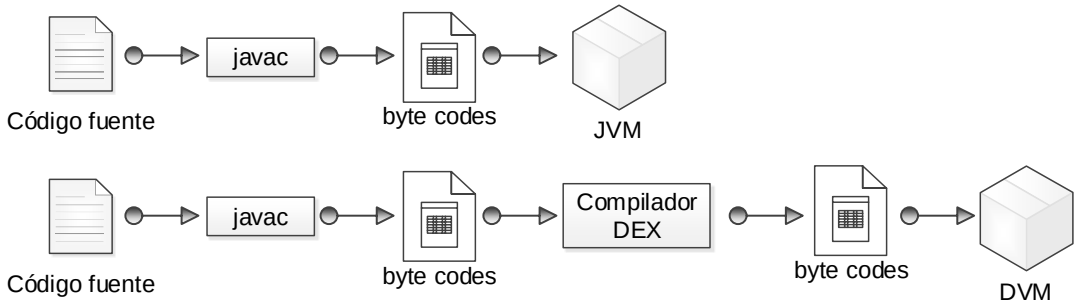


Figura 7. Comparativo de los procesos en Dalvik

Adaptado de (Revelo, 2014)

- **Gradle**

Es una herramienta de gran utilidad para los programadores de Android, facilita el compilado, empaquetado, testeo y liberación de aplicaciones que se basen en la JVM, mediante el lenguaje "Domain Specific Language" (DSL); utiliza "Groovy" para escribir el código fuente así como para la construcción de del proyecto (Revelo, 2014).

- **Núcleo Linux (Linux Kernel).-** Es la base principal del sistema de Android, donde se encuentran los drivers para diferentes componentes de hardware; gestionar los servicios de memoria, de seguridad, energía entre otros.

1.2.3. Administración de energía

En la actualidad el rendimiento de un sistema operativo se ve reflejado por la eficiencia en el consumo de energía y la capacidad de procesamiento de sus tareas. Entendiéndose todas aquellas que se encuentra funcionando en segundo plano; Android basa su administración y control de energía a través del "*PowerManager"* del Núcleo de Linux, mediante "*Wakelocks*" independientemente de que la pantalla está bloqueada o el dispositivo no esté en uso.

Un "*Wakelocks*" es una "orden" que le da una aplicación al procesador del dispositivo Android para que éste se mantenga despierto, activo o en funcionamiento (Wakelock de CPU). Las aplicaciones también pueden generar

este tipo de orden a la pantalla (Wakelock de pantalla) para que se mantenga encendida (Sei, 2013).

López (2012) menciona que el mecanismo de administración de energía de Android, se basa en *Wakelocks*, para controlar el consumo de energía. Un *Wakelock* es una función del servicio "*PowerManager"* para el control de energía en el dispositivo.

Este tipo de administración permite que los recursos asignados a los componentes y aplicaciones se encuentren activos, por lo tanto, es obligatorio crear y adquirir "*Wakelocks*"; si este no está activo, pasa a un estado de bajo consumo al apagarse la CPU, optimizando de esta manera la energía del dispositivo cuando no se use (López, 2012).

En relación a lo anterior, la Figura 7 presenta tres estados "*SLEEP*", "*NOTIFICATION*", y "*AWAKE*" para la administración de energía.

> Cuando una aplicación obtiene un wakelock completo o se produce un evento por actividad de la pantalla o el teclado, el dispositivo se mantiene o se mueve al estado "AWAKE". Si hay un timeout o se presiona el botón de apagado/encendido, se produce la transición al estado "NOTIFICATION". Mientras se adquiera un wakelock parcial el dispositivo se mantiene en el estado "NOTIFICATION". Cuando todos los wakelocks parciales se liberan, se pasa al estado "SLEEP". En este estado, si todos los recursos se activan, ocurre la transición al estado "AWAKE" (López, 2012, p. 33).

Con respecto a la Figura 8, si el dispositivo móvil no tiene un Wakelock ejecutándose, el sistema operativo Android inmediatamente suspende la CPU y pasa al estado "SLEEP", donde el consumo de energía es relativamente bajo, desde luego la CPU sigue realizando tareas. Es importante mencionar que si la pantalla se apaga, no pasa al modo antes mencionado, la CPU no está activa, pero trabaja con bajo consumo de energía por el nivel de actividad que realice.

Figura 8: Estados de la administración de energía en *Android*

Adaptado de (Advanced Android Power Management and Implementation of Wakelocks, 2008)

1.2.4. Arquitectura de administración de energía

La administración de energía en un dispositivo móvil se realiza mediante el módulo "*PowerManagement"*; ejecuta un marco de trabajo para la administración y control de energía en el dispositivo y determina el estado de consumo del dispositivo móvil (ver Figura 9).

Asimismo controla los periféricos mediante los drivers de bajo nivel y el uso de los *Wakelocks*; asimismo se extiende este control al teclado, a los botones y la iluminación/visualización de la pantalla.

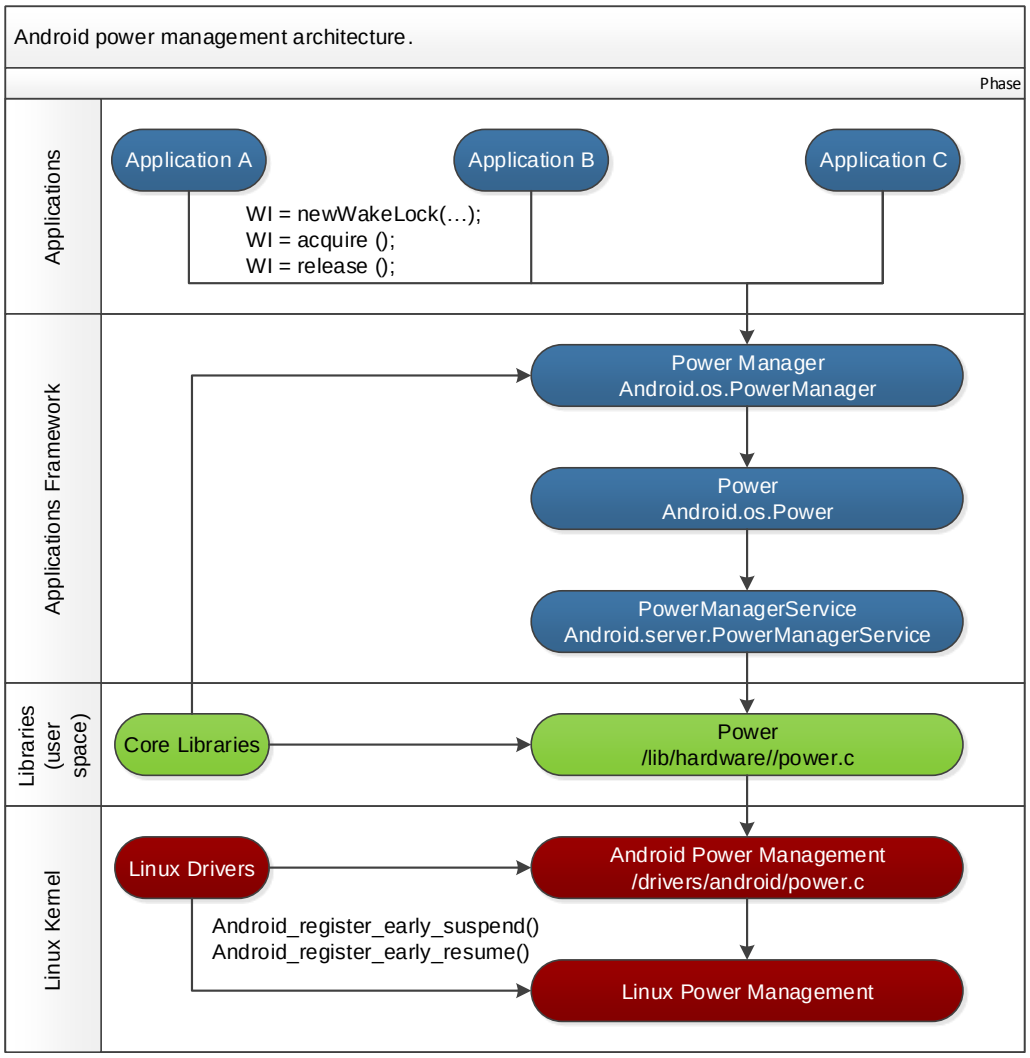


Figura 9. Arquitectura de administración de energía

Adaptado de (López ,2012, p. 35)

En resumen, el consumo de cada periférico se controla mediante el uso de los *Wakelocks* y de las API de Android cuando una aplicación necesita que uno de los periféricos permanezca encendido (López, 2012, pp. 34-36).

1.2.5. Tipos de *Wakelocks*

Existe cuatro (4) principales *Wakelocks* en Android; si uno de ellos no está activo, el dispositivo se suspende para ahorrar energía. La Tabla 4 detalla los Wakelocks para el CPU, pantalla y teclado en un dispositivo móvil.

Tabla 4. Lista de *Wakelocks*

Flag Value	Descripción	CPU	SCREEN	KEYBOARD
PARTIAL_WAKE_LOCK	Este modo asegura que la CPU se mantenga activa a pesar de que la pantalla se apague.	ON	OFF	OFF
SCREEN_DIM_WAKE_LOCK	La pantalla se mantiene encendida a pesar de bajar el brillo y el teclado puede apagar su iluminación.	ON	DIM	OFF
SCREEN_BRIGHT_WAKE_LOCK	Permite el máximo brillo cuando la pantalla está encendida, mientras el teclado puede apagar su iluminación.	ON	BRIGHT	OFF
FULL_WAKE_LOCK	Tanto la pantalla como el teclado se mantienen encendidos con el máximo brillo.	ON	BRIGHT	BRIGHT

Adaptado de (Motlhabi, 2008, p.4)

1.2.6. Arquitectura del mecanismo de gestión de energía de Android mediante *Wakelocks*

Mediante los *Wakelocks* se gestiona los recursos de los diferentes servicios y aplicaciones que requiere un dispositivo móvil; por ejemplo "*FULL_WAKE_LOCK"* impide que se apague la pantalla y el teclado y permanezca activa la CPU.

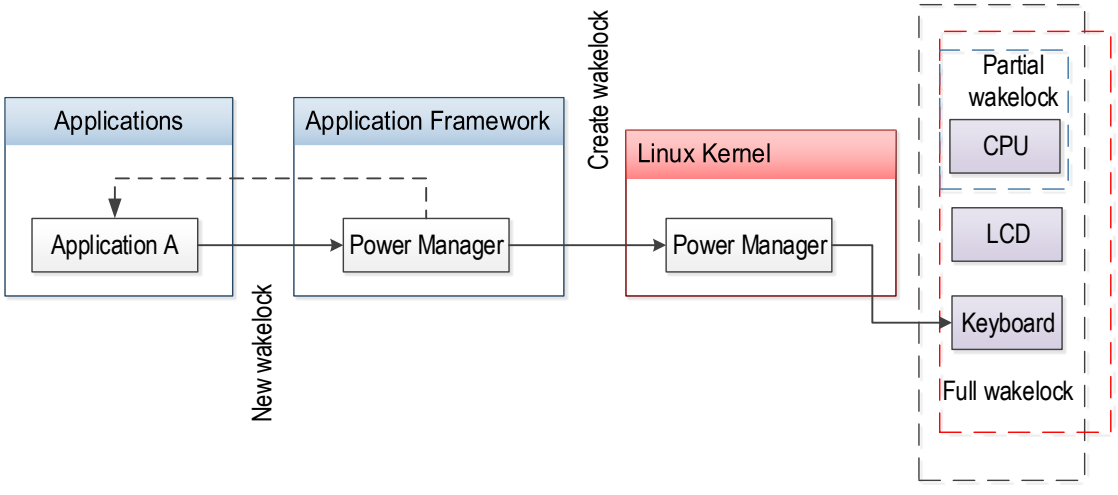


Figura 10. Administración de energía mediante *Wakelocks*

Adaptado de (López , 2012, p. 37)

1.2.7. Flujo de un *Wakelocks*

El monitoreo de la batería y el estado del dispositivo móvil es una función que realiza el *PowerManager*. La Figura 11, representa el flujo de un *Wakelocks*, cuando se requiere economizar energía; si un *Wakelock* no está activo, pasa a un estado de bajo consumo en el dispositivo, este proceso prolonga y optimiza la batería.

Figura 11. Flujo de vida de un *Wakelocks*

Adaptado de (López, 2012, p.38)

1.2.8. Entorno de trabajo y desarrollo en Android

El impulso de aplicaciones sencillas y accesibles a la mayoría de sistemas y plataformas móviles es el objetivo de la comunidad de programadores para el desarrollo de Android. Utilizan las herramientas del SDK junto con el lenguaje de programación Java para desarrollar aplicaciones móviles.

1.2.9. Flujo de ejecución

El flujo de un script que se muestra en la Figura 12, lo define el autor de la siguiente manera:

> En su nivel más bajo, SL4A (Scripting Layer for Android) es un entorno de ejecución de scripts; su función principal es crear y ejecutar scripts de distintos leguajes de programación. Su estructura se organiza a través de un árbol jerárquico para cada tipo de lenguaje. Para transferir este código fuente a la plataforma Android, se realiza una compilación cruzada para la arquitectura ARM usando el NDK (Android Native Development Kit). El módulo Android se cargará como una biblioteca cuando SL4A lanza un intérprete específico y a partir de ese punto, los scripts serán interpretados línea a línea (Uriol, 2011, p. 61).

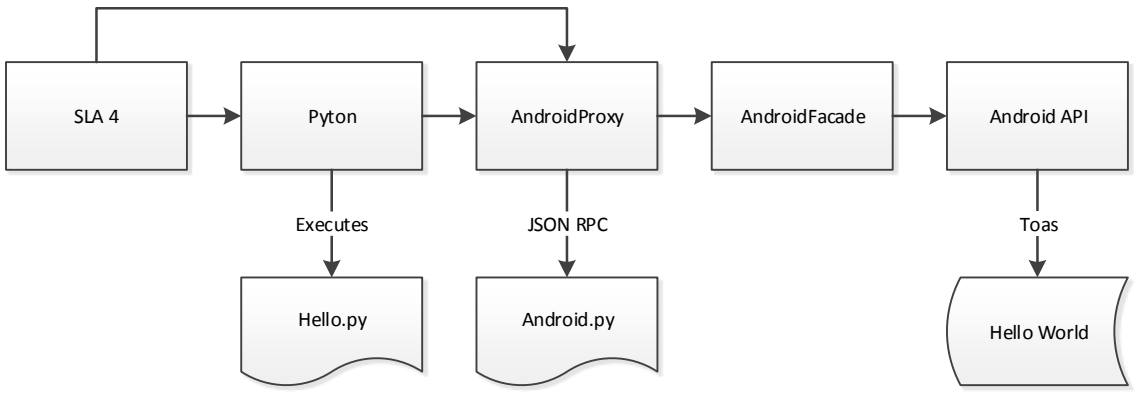


Figura 12. Flujo de un script – SL4A

Adaptado de (Uriol, 2011, p. 62)

1.2.10. Comunicación en la arquitectura SL4A

Para la comunicación entre el SL4A y el sistema operativo Android, el autor según la Figura 13 indica:

> Se basa, en el protocolo de llamada a procedimiento remoto, RPC (*Remote Procedure Call*) y en el formato de datos JSON (*JavaScript Object Notation*). El mecanismo RPC se utiliza normalmente en las arquitecturas distribuidas en las que hay una transferencia de información entre cliente y servidor. En el caso de SL4A, el SO Android es el servidor y el fichero SL4A el cliente. El servidor RPC (implementado como una aplicación Java de Android) tendrá acceso directo a la API de Android y se comportará como un *proxy* remoto mediante el patrón de diseño '*Java Android API* Facade' que encapsula y proporciona el acceso a las APIs seleccionadas de Android (Uriol, 2011, p. 62).

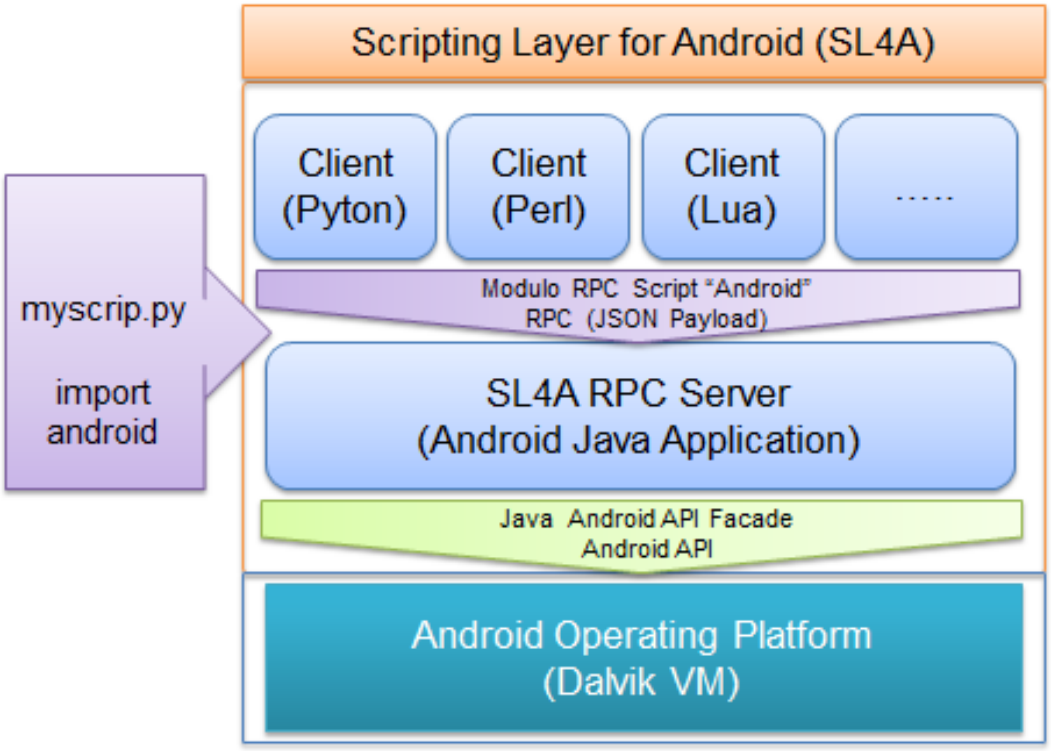


Figura 13. Arquitectura Scripting Layer

Adaptado de (Uriol, 2011, p. 62)

1.2.11. Arquitectura de una aplicación móvil

Esta compuesta por la capa presentación, la cual sirve de interprete entre el usuario y los componentes gráficos de la GUI (graphical user interface).

Seguidamente se tiene la Capa de Negocios, que establece las reglas y el flujo de trabajo, a traves de las diferentes operaciones y procesos para el acceso a los datos.

Finalmente esta la Capa de Datos, que establece los diferentes tipos de datos a utilizar y almacenar en la Base de datos de la aplicación (Márquéz, 2015).

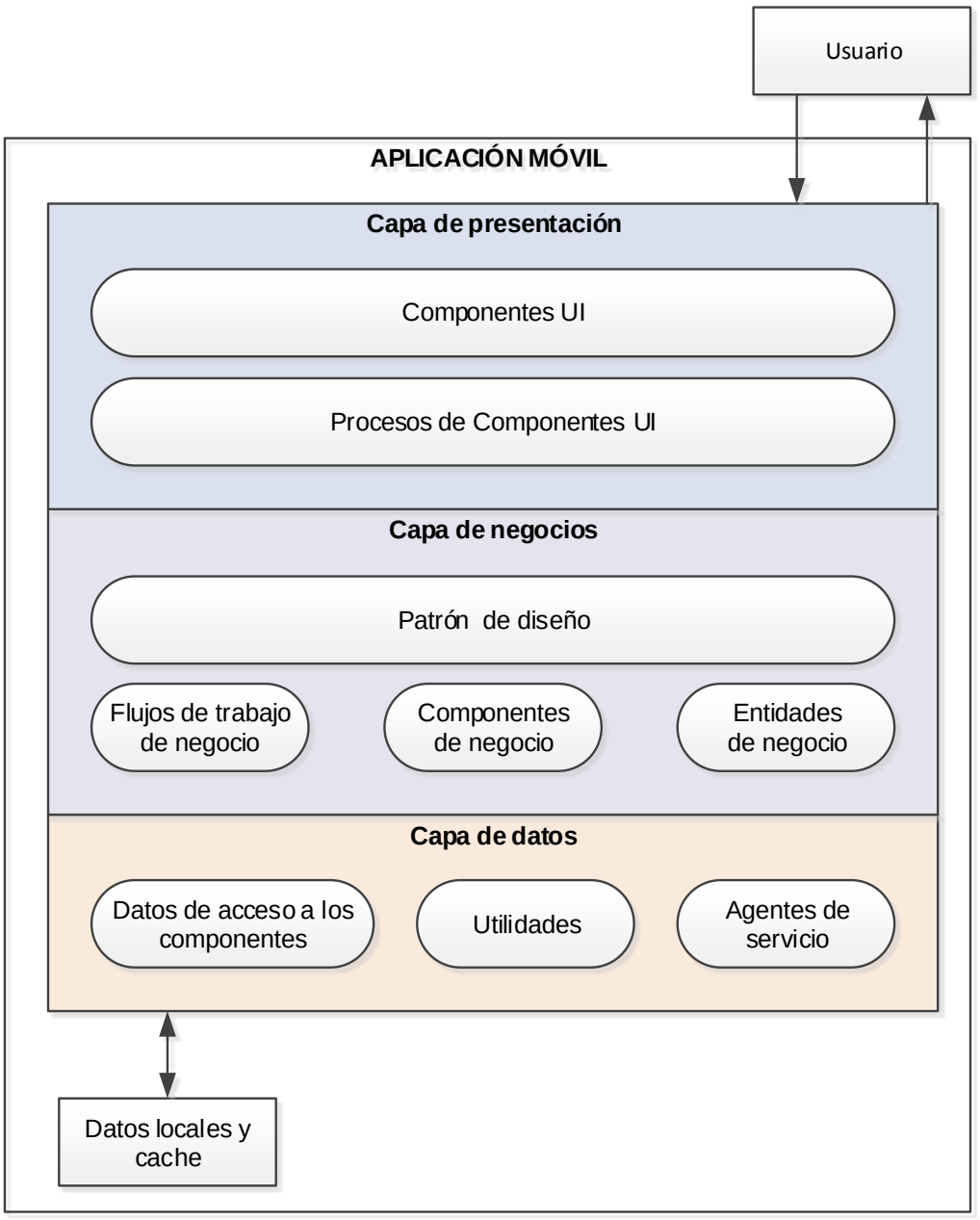


Figura 14. Arquitectura de una aplicación móvil

Tomado de (Márquez, 2015, p. 44)

En resumen las capas de Presentación, Negocios y Datos se interrelacionan entre sí, cumpliendo diferentes funciones. Para ilustrar esto la Figura 15 expone algunas características.

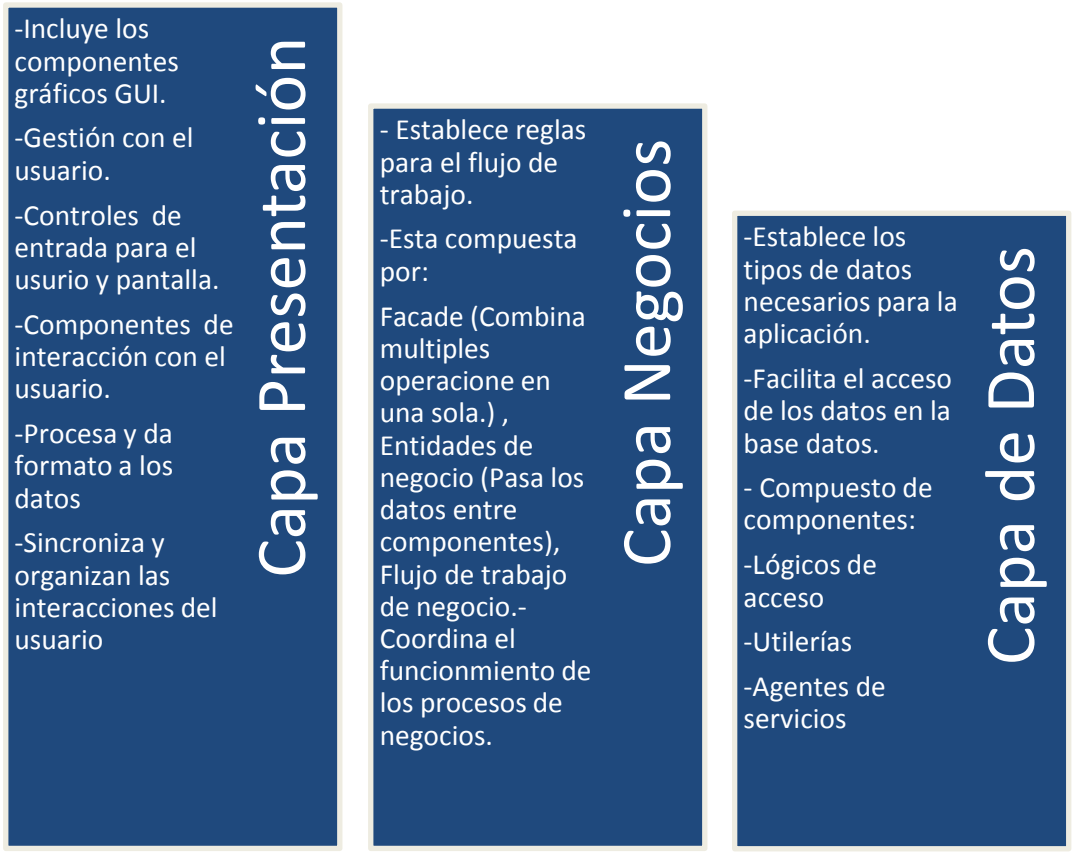


Figura 15. Características de las capas

Adaptado de (Márquez, 2015, pp. 44-45)

1.3. Medición de energía

Actividades como ver videos, conectarse a redes Wi-Fi, enviar mensajes de texto, acceder a redes sociales, etcétera., requieren de un dispositivo móvil con autonomía en su batería.

Al momento no se tiene soluciones que prolongue la duración de la batería más allá del promedio de un día. Sin embargo, se tiene software que permite conocer el consumo de energía del dispositivo móvil en conjunto y no por componente.

Según López (2012) “Cuando nos referimos al consumo de un dispositivo, se habla tanto del consumo de energía como de potencia. Con energía nos

referimos a la capacidad para realizar un trabajo que se mide en Julios según el Sistema Internacional de Unidades" (p.40).

Con referencia a lo anterior "la potencia es la energía consumida durante una unidad de tiempo. La energía puede ser medida por la potencia durante el tiempo de ejecución de una tarea"; la unidad de medida es el joule (J) y de la potencia el watt (W) (Marquéz , 2015, p. 52).

A continuación la Tabla 5 presenta las ecuaciones de energía y potencia:

Tabla 5. Ecuaciones de energía y potencia

Ecuación	**Descripción**	**Fórmula**
Energía	$E_J = Representa\ la\ enegía\ en\ Joules$ $V\ = Es\ el\ voltaje$ $T = Tiempo\ que\ dura\ la\ tarea$	$E_J = V.I.T$
Potencia	$P_W = Representa\ la\ Potencia\ en\ Watts$ $E_J = Representa\ la\ enegía\ en\ Joules$ $T = Tiempo\ que\ dura\ la\ tarea$	$P_w = \frac{E_J}{T}$

Adaptado de (Márquez, 2015, p.55)

1.3.1. Administración de energía en un dispositivo móvil

Entender el funcionamiento de la arquitectura de un dispositivo móvil, así como sus políticas de gestión de energía, permitirá al desarrollador obtener información de los componentes y de los controladores del sistema que se encuentran relacionados con el *kernel* del sistema operativo.

Esta interrelación se genera a través del estándar de la Interfaz Avanzada de Configuración y Energía (ACPI) a nivel de hardware; cuya función es entender los mecanismos avanzados para la gestión, medición y ahorro de la energía.

Por otra parte el Kernel de Linux permite analizar y visualizar los registros a gestionarse a través de la interfaz *Traceview* según se expone en la Figura 16.

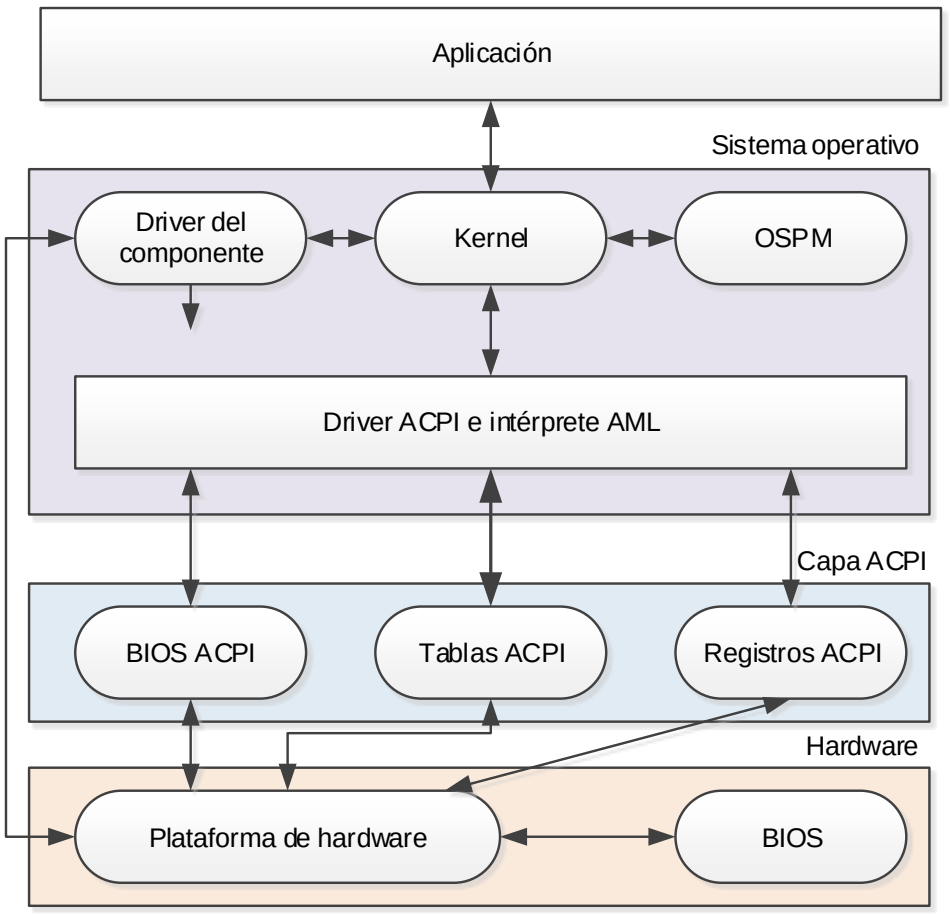


Figura 16. Interfaz de energía y configuración avanzada

Tomado de (Márquez, 2015, p. 53)

1.3.2. Estados para el consumo de energía en los dispositivos móviles

Los estados de un dispositivo móvil son: activo, inactivo y suspendido; estos estados permite obtener información para valorar la potencia consumida en cada componente.

La Figura 17 describe las diferentes características que tienen los estados antes citados.

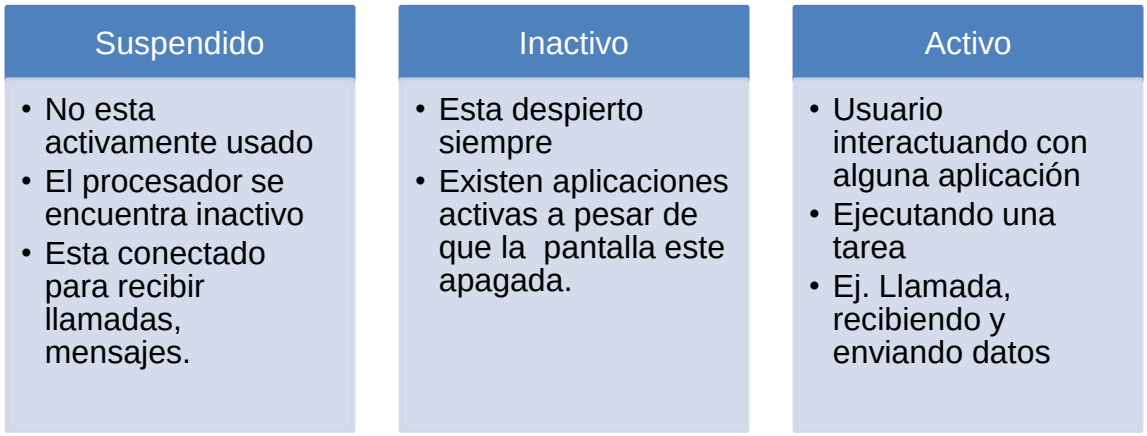


Figura 17. Descripción de los estados de un dispositivo móvil

Adaptado de (López , 2012, pp. 42-43)

1.3.3. Medida de potencia

Medir la potencia y evaluar el consumo de energía en un dispositivo móvil tiene su complejidad y depende de los componentes que se deseen medir y de las herramientas que se usen para este propósito.

En ese contexto, el análisis de la administración de energía requiere conocer en primera instancia, la distribución de los diferentes componentes en un dispositivo móvil.

Como se puede ver la Figura 18 representa los componentes principales de un dispositivo móvil:

- El CPU
- Bluetooth
- El GSM/GPRS
- RAM
- Códec de voz
- Gestor de energía
- Wi-Fi
- GPS

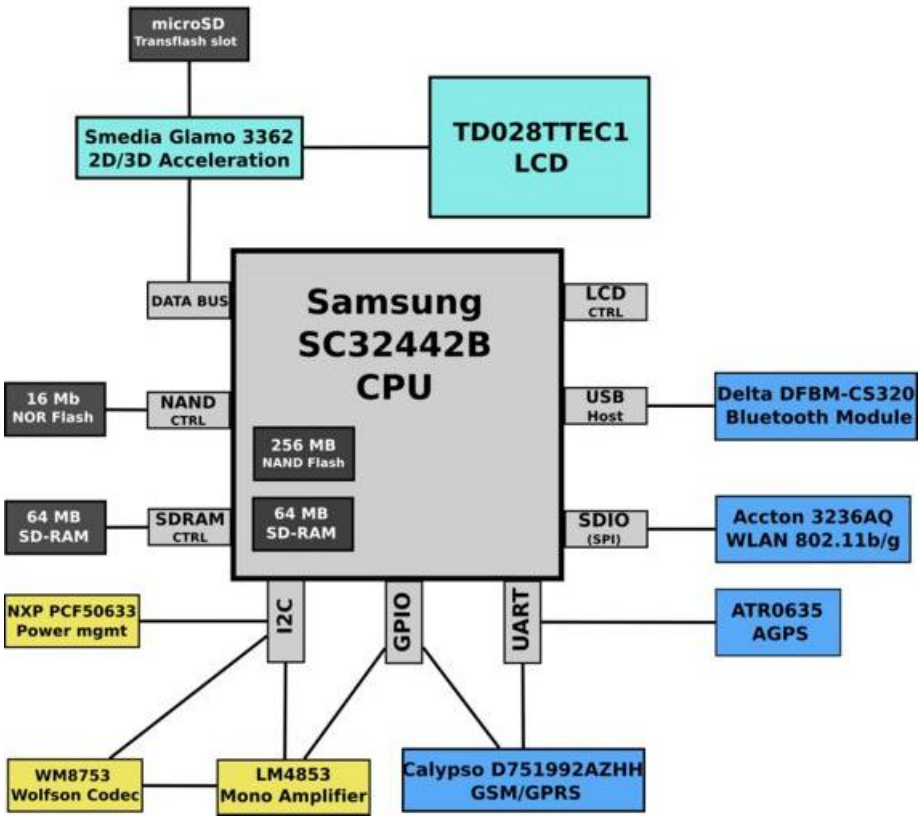


Figura 18. Componentes de un Smartphone

Tomado de (OPENMOKOWIKI, 2008)

2. CAPÍTULO II. METODOLOGÍA

Establecer una metodología de medición de energía para un componente especifico en un dispositivo móvil, requiere seguir algunas directrices, sean estas a través del uso de hardware o de software, las mismas no son fáciles, debido a la complejidad de los circuitos que poseen ciertos componentes y por el manejo de diferentes intensidades y voltajes.

2.1. Descripción de la metodología

La metodología que se empleó para medir el consumo de energía de los componentes de Bluetooth y GPS se presenta en la Figura 19.

Es necesario aclarar al lector que en este documento se hace referencia a la palabra "componentes" de Bluetooth y GPS, esto no representa el hardware como tal, al contrario, se relaciona con la funcionalidad que tiene el Bluetooth y el GPS.

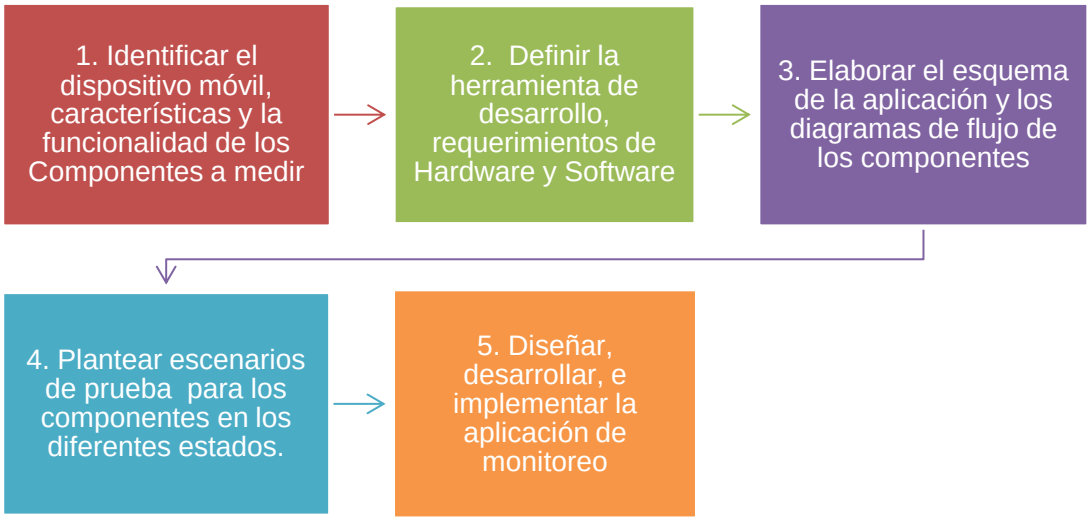


Figura 19. Metodología de medición

Adaptado de (Márquez, 2015, p. 29)

Esta metodología se conforma de cinco actividades. Inicia con la selección del dispositivo móvil de estudio, la descripción de las especificaciones técnicas de

este dispositivo, así como de la funcionalidad de los componentes Bluetooth y GPS.

Luego se define la herramienta de desarrollo a utilizar, considerando los requerimientos mínimos de hardware y software, para para la instalación en un equipo con Sistema Operativo Windows 7.

Posterior se elabora el esquema de la aplicación para su funcionamiento junto con los diagramas de flujos por cada componente.

Seguidamente se plantean escenarios de medición y pruebas por componente; dichos escenarios arrojan resultados (muestras) que son parte del análisis a realizar, respecto al consumo potencia y energía del dispositivo móvil, tomando en cuenta los niveles de rendimiento, es decir, el tiempo normal que se demora cada componente en la ejecución de las diferentes pruebas de medición.

Finalmente se diseña, desarrolla y se implementa en el dispositivo móvil, una aplicación de consumo de energía para los componentes de Bluetooth y GPS. Para esto se instala la herramienta de desarrollo de Android Studio, los recursos que demande esta herramienta , las configuraciones, los archivos de construcción de *Gradle* y el entorno de desarrollo integrado que posee dicha herramienta.

2.1.1. Identificación del dispositivo móvil

El dispositivo móvil a utilizar es el Smartphone Samsung Galaxy J7 LTE, Modelo SM-J700M/DS (ver Figura 20) cuyo sistema operativo es Android con la versión 5.1.1 (Lollipop); posee un procesador de 1.5 GHz que permite ejecutar juegos y aplicaciones.

A nivel de hardware este dispositivo soporta dos tarjetas SIM, con su respectivo gestor (dual-SIM) para utilizar dos operadores móviles. Además incluye Bluetooth Versión 4.0 con A2DP, Wi-Fi 802.11 a/b/g/n, (ver Tabla 6).

Tabla 6. Componentes principales Samsung Galaxy J7 LTE

Modelo y características físicas	
Marca	Samsung
Modelo	SM-J700M/DS
Antena	Antena interna
Sistema Operativo (SO)	
Sistema Operativo	Android 5.1.1 Lollipop
Hardware y memoria	
CPU / Procesador	Qualcomm MSM8939 Snapdragon 615 octa-core de 1.5 GHz
Memoria RAM	1.5GB LPDDR3
Memoria interna	16 GB
Memoria expandible	microSD hasta 128GB
Red móvil y de datos	
2G GSM	GSM850, GSM900, DCS1800, PCS1900
3G	B1(2100), B2(1900), B5(850), B8(900) GPRS, EDGE, UMTS, HSDPA, HSUPA,
4G FDD LTE	B1(2100), B3(1800), B5(850), B7(2600), B8(900), B20(800)
4G TDD LTE	B40(2300)
Pantalla y el sistema gráfico	
Tecnología	Súper AMOLED
Resolución de la Pantalla	720 x 1280 píxeles
Conectividad	
USB	USB 2.0 Micro-B (Micro-USB)
Bluetooth	Versión 4.1, con A2DP
Wi-Fi	802.11 a/b/g/n
GPS	A-GPS, GeoTagging y GLONASS
Batería y Autonomía	
Tipo de batería	Litio - Extraíble (desmontable)
Amperios de la batería	3000 mAh

Adaptado de (Samsung, 2016)

Smartphone Samsung Galaxy J7

Figura 20. Smartphone Samsung Galaxy J7

Tomado de (Citylar, 2016)

2.1.2. Componentes a medir

Tal como se ha venido mencionando en los apartados anteriores, los componentes a medir son el Bluetooth y el GPS. En esta ocasión se expone las características y funcionalidades de dichos componentes.

2.1.2.1. Bluetooth

Bluetooth constituye un protocolo de comunicación inalámbrica de corto alcance para redes inalámbricas de área personal (Wireless Personal Area Network, WPAN) y de bajo costo. Se caracteriza por tener baja potencia de transmisión a través de un enlace de radio frecuencia. Entre uno de los propósitos de este protocolo, está el sustituir el cable en la interconexión de varios dispositivos electrónicos como Smartphones, Laptops, etcétera, (Becvar, Mach y Pravda, 2013, p. 84).

La velocidad de transmisión de Bluetooth en su versión inicial es de 1 Mbps, sin embargo y considerando las diferentes versiones que existen en la actualidad, estas velocidades varían. Por ejemplo la versión 2.0 que se muestra en la Tabla 7, es de 3 Mbps

Tabla 7. Clasificación del Bluetooth según la capacidad de canal

Versión	Ancho de banda	Características
1.2	1 Mbps	Conexión rápida, fue ratificado como estándar IEE 802.15.1
2.0 + EDR (Enhanced Data Rates)	3 Mbps	Velocidad de datos mejorada
3.0 + HS (High Speed)	24 Mbps	Usa MAC/PHY como transporte de alta velocidad
4.0 + LE (Low Energy)	32 Mbps	Se fundamenta en la reducción del consumo, disminuye la potencia de transmisión de la señal radio utilizada y el radio de cobertura

Adaptado de (Becvar, Mach y Pravda, 2013, p. 84)

Desde el punto tecnológico de transmisión, Bluetooth utiliza **FHSS** (*Frequency Hopping Spread Spectrum*) con el fin de solucionar dificultades de interferencia y desvanecimiento que se producen a lo largo de una conexión.

Cualquier dispositivo Bluetooth utiliza el salto de frecuencia, cuya función está encaminada dividir en varios canales de salto, empleando frecuencias de portadora en la transmisión de un paquete hacia un canal u otro de forma pseudo aleatoria; en relación a las bandas de frecuencia estas se encuentran alrededor de los 2,4 GHz y un ancho de banda de 83,5 MHz (Becvar, Mach y Pravda, 2013, p. 85).

2.1.2.1.1. Grupos de trabajo

La tecnología WPAN establece cuatro grupos de trabajo, su clasificación se da en función de las características y especificaciones de comunicación en la transmisión y recepción de datos según se detalla en la Tabla 8.

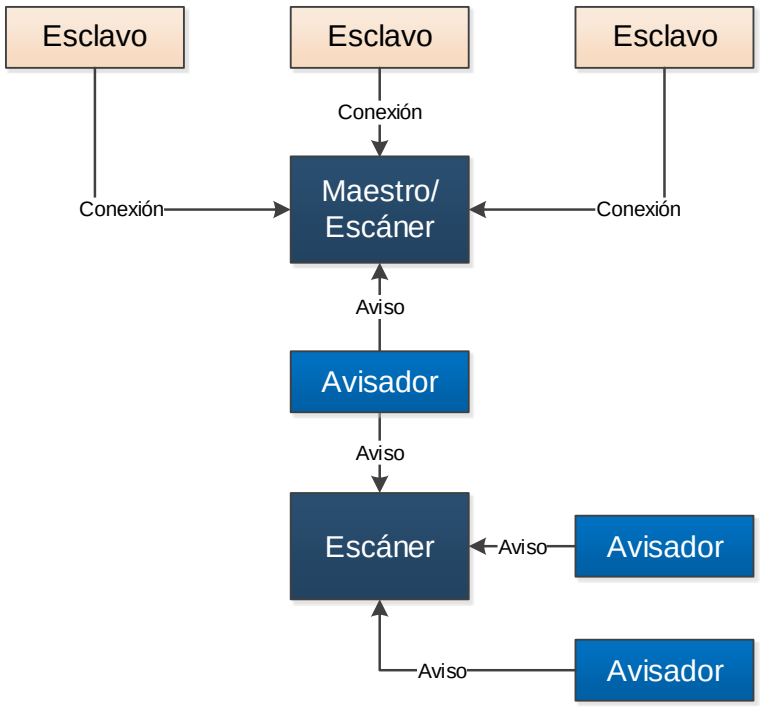
Tabla 8. Grupos de trabajo para el estándar IEEE 802.15

Grupo de trabajo	Características	Descripción
IEEE 802.15.1	Aprobado por IEEE Standard Association (IEEE-SA)	Permite la transmisión de datos y voz, además interconectar, teléfonos celulares, computadoras, etc.
IEEE 802.15.2	Establece la coexistencia de sistemas inalámbricos, trabajando en bandas de frecuencia que no requieren licencia	Permite cuantificar los efectos por interferencia, además establece mecanismos de coexistencia entre dispositivos WPAN y WLAN.
IEEE 802.15.3	Surge de la necesidad de poder trasmitir datos de forma rápida y eficiente en una WPAN.	Trabaja en la banda de los 2.4 GHz, posee menor interferencia debido a que ocupa un ancho de banda (15 MHz) menor y trasmite con menor potencia (8dBm) para una distancia de 30 a 50 metros. Los rangos de velocidad son: 11, 22, 33, 44 y 55 Mbps.
IEEE 802.15.4	Destaca la flexibilidad en la red, el bajo costo en su implementación y bajo consumo de energía.	Orientado para aplicaciones en el ámbito de la domótica y la industria, gracias a su facilidad de implementación

Adaptado de (Acosta, M., 2006, p. 11-14)

2.1.2.1.2. Generic Acces Profile (GAP)

GAP es considerado el perfil frecuente de Bluetooth para todas sus versiones, entre uno de los métodos que maneja este perfil tenemos los procedimientos de escaneo y asociación de los dispositivos. Define los roles de maestro, esclavo, avisador y escáner de comunicación para los dispositivos integrantes de una red (Rica, 2013).



Maestro: El dispositivo se puede comunicar con uno o varios esclavos.
Esclavo: Solo se puede comunicar con un maestro a la vez.
Avisador: Anuncia paquetes con información pero no puede recibirlos.
Escáner: Recibe los paquetes de los avisadores y puede establecer comunicación con ellos

Figura 21. Topología de red en Bluetooth

Adaptado de (Rica, 2013, p.25-26)

2.1.2.2. GPS

El GPS (Global Positioning System Sistema Global de Posicionamiento) fue desarrollado por el Departamento de Defensa de los Estados Unidos de América. Es un sistema global de navegación por satélite (GNSS) que permite determinar con precisión hasta de centímetros, la posición en todo el mundo de un objeto, persona, etcétera (Becvar, Mach y Pravda, 2013, p.94).

En efecto provee servicios de posicionamiento, navegación y temporización, además consta de los segmentos de espacio, control y usuario que se muestra en la Tabla 9.

Tabla 9. Segmentos de GPS

Espacial	Control	Usuario
• Consta de 24 satélites. • Vuelan alrededor de 20.200 km por encima de la tierrac, con un período orbital de 11 horas y 58 minutos. • Organizados en 6 órbitas. • Transmiten señales con información de posición del usuario.	• Consta de instalaciones en tierra, para el seguimiento, control y gestión del segmento espacial. • Es responsable de la gestión de todos los procedimiento de control. • Recibe y analiza la información de los satélites y de navegación.	• Esta representado por un receptor GPS, el cual procesa la información recibida de los satélites. • Posee un chip GPS con una componente de radio, procesador de señal digital, memoria, parte de control y una interfaz de unidad central.

Adaptado de (Becvar, Mach y Pravda, 2013, p.94)

2.1.2.2.1. Integración con dispositivos móviles

Actualmente la mayoría de los dispositivos móviles se pueden vincular a un receptor de GPS. Esta vinculación se da a través de los módulos que poseen los dispositivos móviles para comunicarse vía inalámbrica mediante la entrega de los datos de posicionamiento del usuario (Martinez, 2011).

2.1.2.2.2. Comunicación con el receptor

La comunicación con el receptor GPS se la realiza a través del puerto COM y el protocolo normalizado de comunicación entre equipos y dispositivos de navegación NMEA (National Marine Electronic Asociation). NMEA se encarga de enviar mensajes de la posición y otros datos respecto al posicionamiento de un receptor GPS.

El diagrama de la Figura 22 describe la secuencia que realiza el GPS para la búsqueda del puerto COM en un dispositivo móvil.

El dispositivo móvil establece el procedimiento para autodetectar el puerto COM. Para esto el GPS consulta si el puerto COM está disponible en el dispositivo móvil, si la respuesta es afirmativa selecciona el puerto, caso contrario envía un mensaje al dispositivo móvil para que se repita el procedimiento inicial.

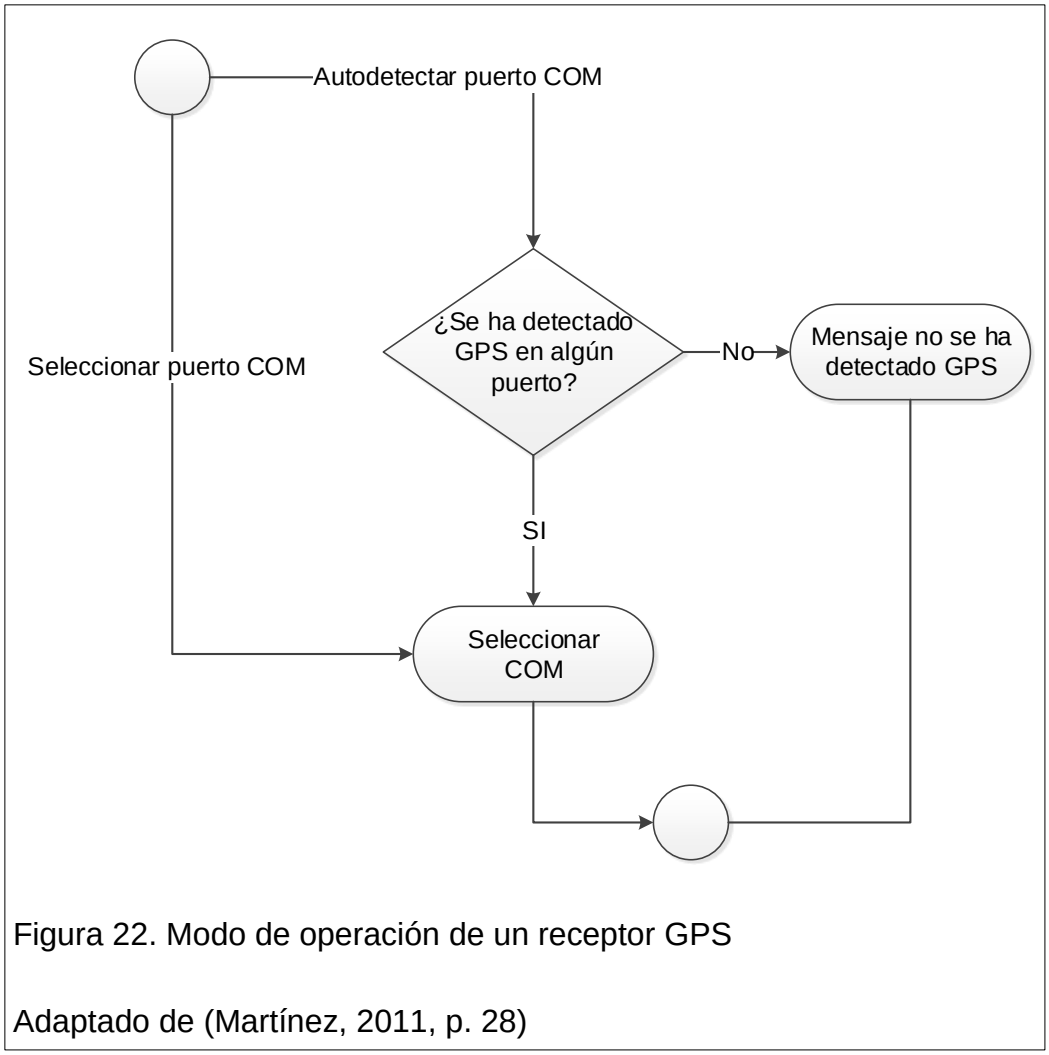


Figura 22. Modo de operación de un receptor GPS

Adaptado de (Martínez, 2011, p. 28)

2.1.3. Herramienta para desarrollo Android Studio

Para realizar la medición de los componentes de Bluetooth y GPS que consumen mayor energía en el dispositivo móvil, se eligió Android Studio, por su entorno de desarrollo y por brindar herramientas para el desarrollo de aplicaciones en un sistema operativo Android.

Entre las características de Android Studio resaltan las herramientas de empaquetado y etiquetado de código, cuyo fin busca organizar grandes cantidades de código.

Figura 23. Herramienta de desarrollo Android Studio

Tomado de (Android libre, 2014)

2.1.3.1. Estructura de un proyecto

Android Studio representa un proyecto “Project” a través de una estructura organizada por carpetas, la misma que está conformada por otras carpetas como por ejemplo: app, build y gradle. En el apartado siguiente se explica las características que tiene esta estructura.

2.1.3.2. Directorios

Con relación a lo anterior, la Figura 24 presenta la distribución organizada de las principales carpetas que utiliza Android Studio en el desarrollo de cualquier aplicación.

En primer lugar tenemos la carpeta **app**, la cual contiene todos los archivos de la aplicación para que sea empaquetada; contiene otras carpetas como: **bluid, libs, src**, entre otras.

Dentro de la carpeta **src** (Source), se tiene la carpeta **main** con los archivos fuentes de **Java**. De igual forma se tiene la carpeta **res** (Resources) con los recursos del proyecto como por ejemplo íconos, diseños, etc., y el archivo de configuración **Android Manifest**, de nuestra aplicación (Revelo, 2014).

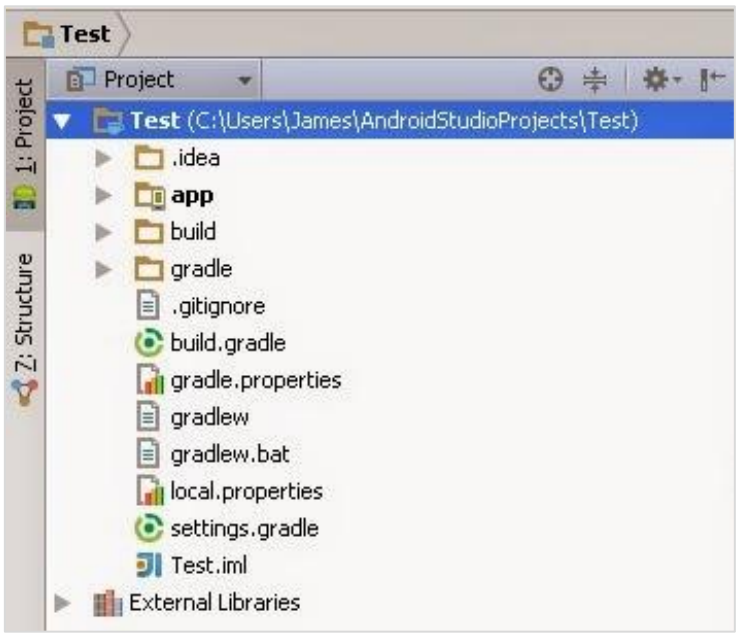


Figura 24. Estructura de un proyecto

Tomado de (Revelo, 2014)

2.1.4. Requisitos

En el desarrollo de la aplicación móvil para los componentes de Bluetooth y GPS se define los requisitos mínimos de hardware y software con el fin de optimizar el tiempo y esfuerzo sin descuidar la calidad del producto final.

2.1.4.1. Hardware

Considerando los requisitos mínimos que requiere Android Studio para ejecutar las herramientas de desarrollo y su correspondiente emulador, la Tabla 10 presenta las características del equipo de cuarta generación que se utiliza en este desarrollo.

Tabla 10. Características DELL LATITUDE E6440

DELL LATITUDE E6440	Descripción
Procesador	Cuarta generación del procesador Intel® Core™ i5-4310M (3MB Caché, hasta 3.40 GHz)
Sistema Operativo	Windows® 7 Professional, 64-bit, Español
Memoria	4GB de Memoria DDR3L a 1600MHz, 1 DIMM
Disco Duro	Disco Duro Híbrido SATA de Estado Sólido de 500GB
Conexiones inalámbricas	Tarjeta Inalámbrica Intel® 6235 802.11n, Bluetooth 4.0, WiFi
Conectividad	Ethernet Gigabit 10/100/1000
Pantalla	Pantalla LED iluminada con antirreflejo de 14.0" de alta definición (HD) (1366 X 768)

Tomado de (DELL, 2015)

2.1.4.2. Software

Se requiere un entorno basado en ciertas tecnologías integradas para desarrollar en Android. Para esto se utiliza el SDK (Software Delevelopment Kit) de Android Studio, el cual nos permite crear en primera instancia una aplicación de monitoreo sobre la base de un emulador para los componentes de Bluetooth y GPS y luego instalar en el dispositivo móvil.

- Android Studio Versión 2.1 para Windows 7
  - JDK 1.8 (Java Development Kit)
  - Android Tools,
  - Android Platform-tools,
  - Android SDK Build-tools (versión actual),
  - Una o más versiones de la plataforma Android,
  - Android Support Repository (extras),
  - Google Repository (extras),
  - Google Play Services (extras).

2.1.5. Esquema de la aplicación

Para el desarrollo de la aplicación de monitoreo se analiza el funcionamiento general de cada componente por separado, y se plantea tres diagramas de flujos. Cabe señalar que previo a realizar cualquier medición, se obtiene información del estado de la batería, respecto al porcentaje de nivel de carga, la temperatura que se expresa en grados centígrados y el voltaje de la misma.

Estos datos iniciales de la batería son proporcionados por el Sistema Operativo Android y sirve de base para las pruebas en los diferentes escenarios que se explican más adelante.

2.1.5.1. Diagrama de flujo Bluetooth

La Figura 25, representa el diagrama de flujo del procedimiento a seguir para obtener información del componente Bluetooth, cuando tenga el estado de activo a través de la aplicación de monitoreo.

Se inicia activando el componente del dispositivo móvil, este a su vez intercambia información (escaneo) con los dispositivos habilitados para el componente Bluetooth dentro del radio de cobertura (no más de 10 metros), es decir ambos dispositivos se ponen de acuerdo para transferir información una vez que se encuentren emparejados.

Una vez conectados inicia el intercambio información desde cualquiera de los dispositivos (Por ejemplo: manos libres, tablet, etc.).

En este procedimiento intervienen los estados apagado, encendido, encendido y en paridad y transfiriendo del componente Bluetooth. La aplicación de monitoreo obtiene información de la potencia, voltaje y de la corriente que se consuma en el intervalo de tiempo y en los diferentes estados del Bluetooth.

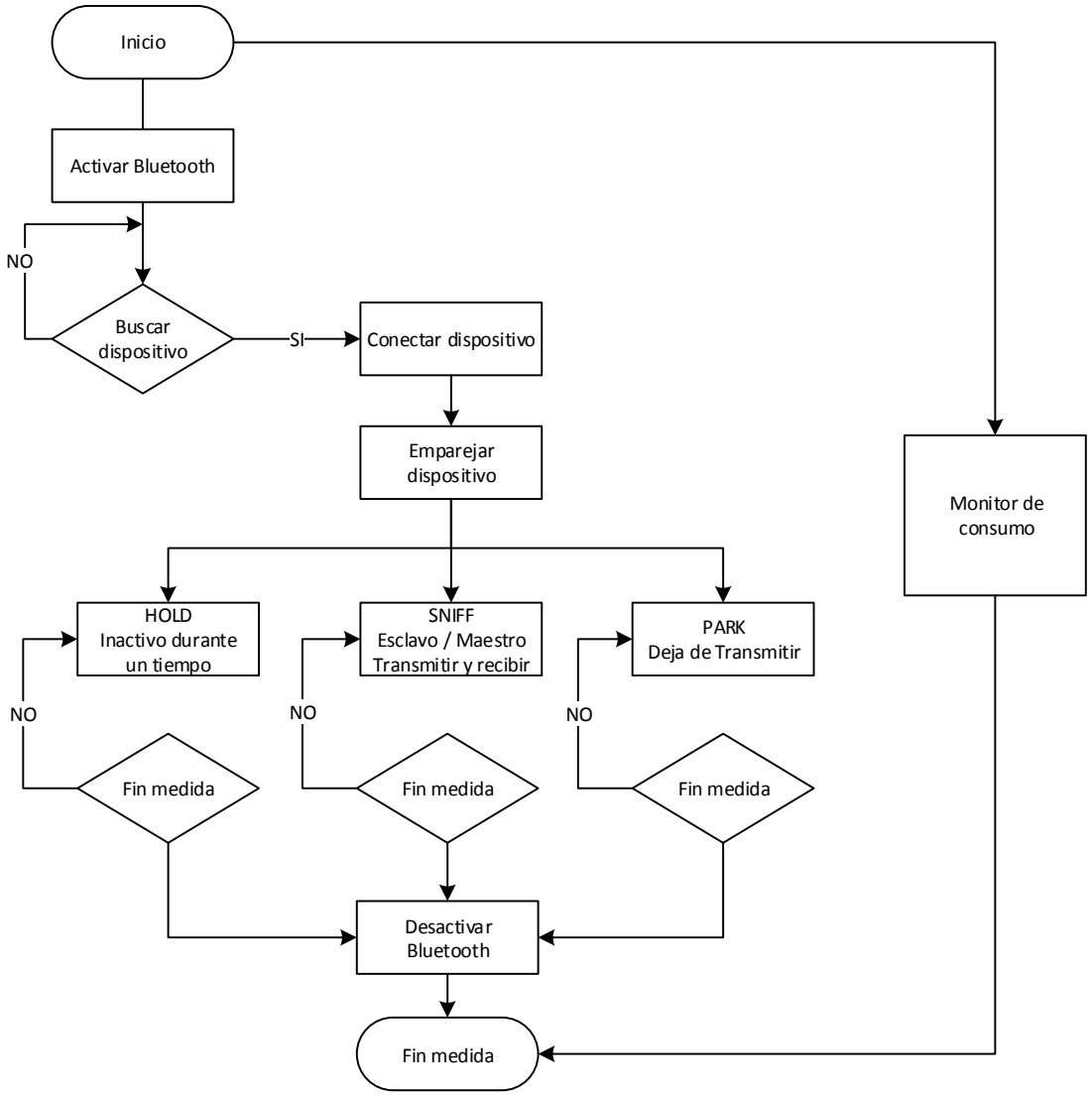


Figura 25. Diagrama de flujo consumo de energía – componente Bluetooth

Adaptado de (López, 2012, p. 67)

Para esto, la aplicación consulta al sistema operativo en la capa más baja. ¿Cuánto está consumiendo la batería, refiriéndose al nivel de carga, temperatura y voltaje?, seguido le vuelve a consultar, ¿estas encendido Bluetooth?, ¿Si o No?, si esta encendido, arroja una corriente mínima de 0,1 mW y una máxima de 10 mW cuando el dispositivo se emparejo y esta transfiriendo información con el otro.

Finalmente la aplicación de monitoreo muestra en tiempo real el consumo de este componente a través de una gráfica.

2.1.5.2. Diagrama de flujo GPS

La Figura 26 corresponde al diagrama de flujo del componente GPS. Al momento de dar click en el boton de encendido, se habilita la conexión en el dispositivo e inicia la transferencia de datos de la ubicación en la que se encuentra el usuario.

Para esto, la aplicación vuelve a consultar al sistema operativo. ¿Cuánto está consumiendo la batería refiriéndose al nivel de carga, temperatura y el voltaje?; de igual forma le consulta ¿estas encendido GPS Si o No?, si esta encendido, tiene una intensidad de corriente por un valor aproximado de 50 mA, caso contrario sería 0 mA.

El resultado de la multiplicación del voltaje de la batería (esta puede ir variando entre 3,8 y 4, dependiendo del celular) y la intensidad de corriente del GPS, es la Potencia consumida por este componente.

Una vez que se establece la conexión antes mencionada, la aplicación de monitoreo recepta los datos de potencia y corriente emitida por el GPS y muestra en tiempo real el consumo de este componente a través de una gráfica.

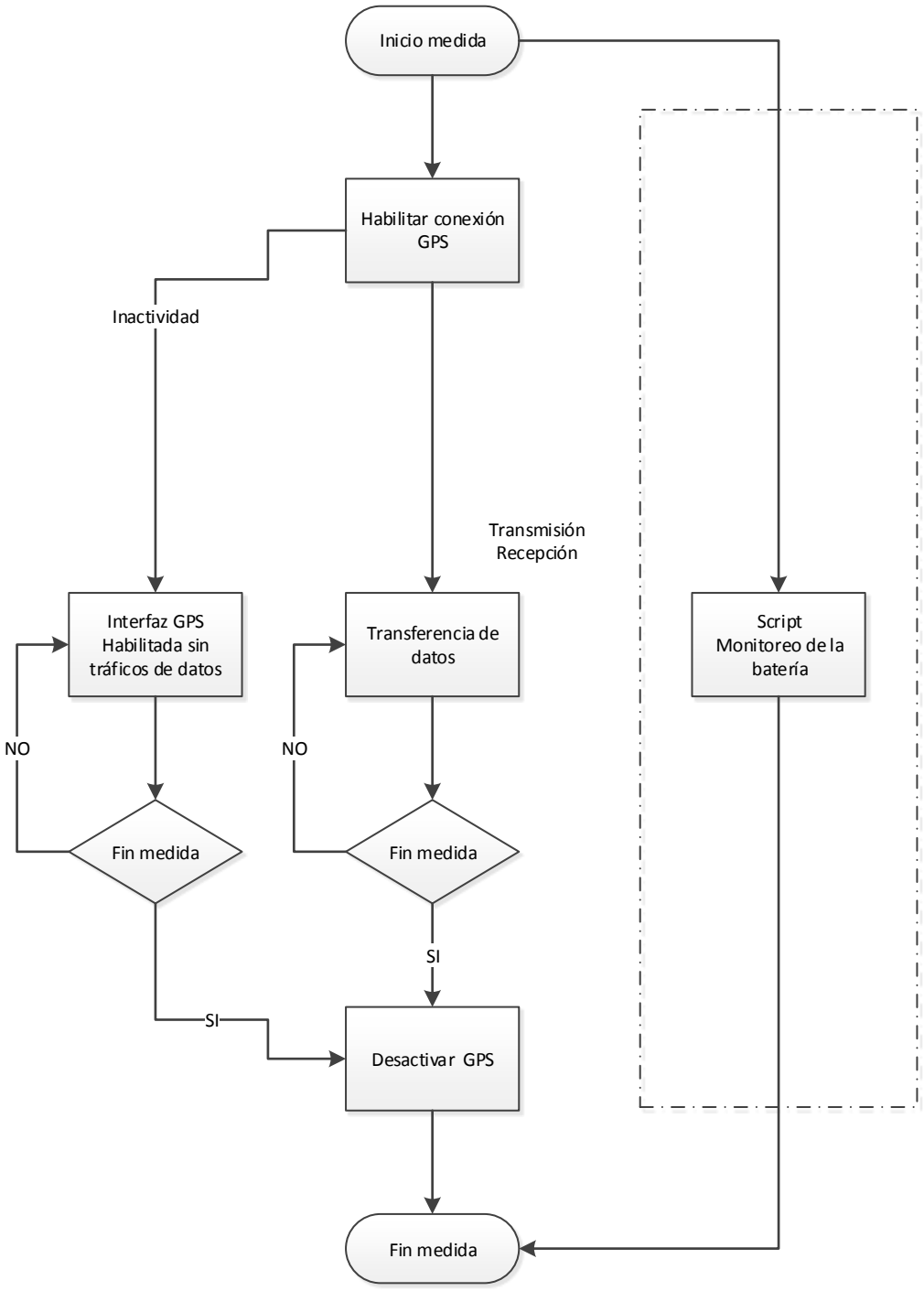


Figura 26. Diagrama de flujo consumo de energía – componente GPS

Adaptado de (López, 2012, p.67)

2.1.5.3. Diagrama flujo de Bluetooth y GPS

La Figura 27 corresponde al funcionamiento de la aplicación de medición de los componentes de Bluetooth y GPS. En primer lugar presenta una pantalla con un menú de opciones relacionados a: Monitor de batería, medición de los componentes, consumo total y finalmente información general de la aplicación.

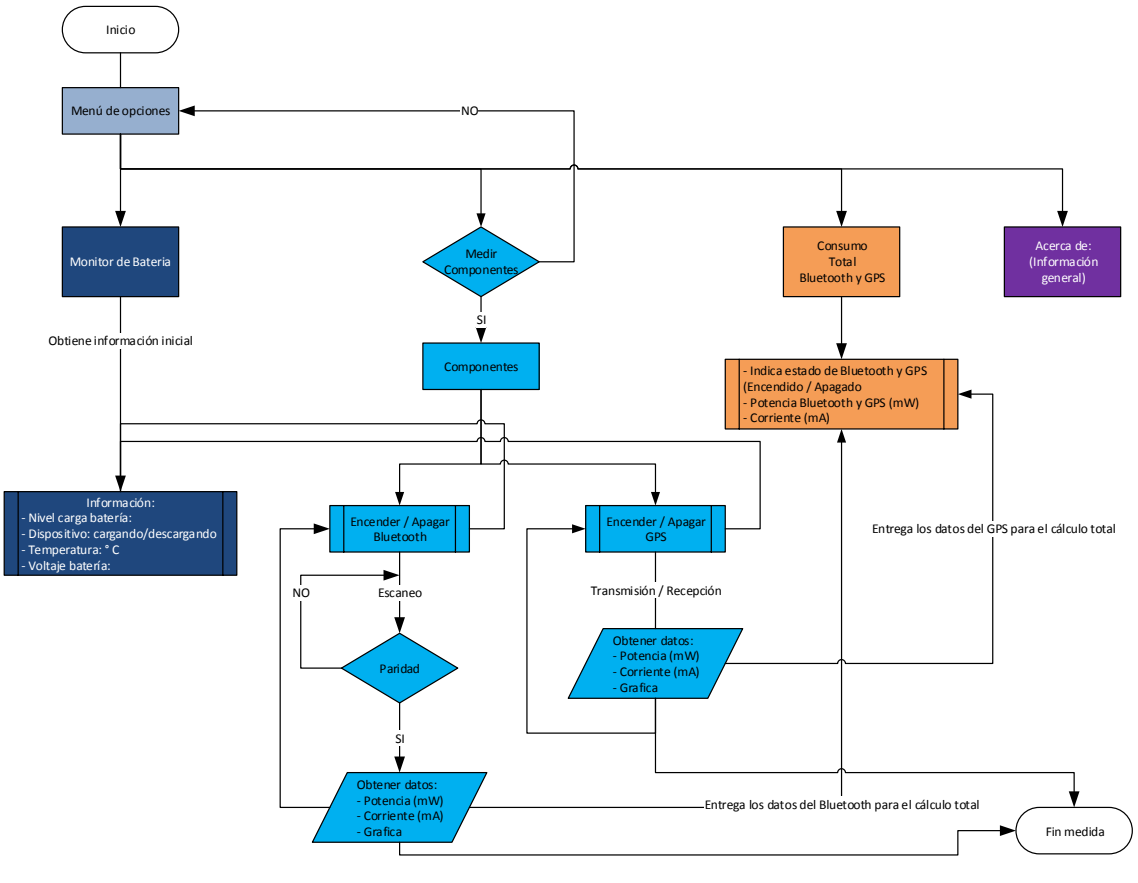


Figura 27. Diagrama de flujo consumo de potencia para Bluetooth y GPS.

2.1.6. Definición de escenarios de mediciones

El proceso de medición para los componentes de Bluetooth y GPS requiere de la definición de escenarios de prueba; para esto partimos con datos iniciales del estado de la batería, que son proporcionados por la aplicación de monitoreo.

2.1.6.1. Estado batería

Inicia con el porcentaje de nivel de carga en un rango de cero a cien por ciento (0% – 100%), y el valor del voltaje de cuatro voltios [4V], siendo este último valor el ideal en la mayoría de dispositivos móviles.

Existen páginas en el Internet que precisan ciertos valores de configuración en el código fuente del Sistema Operativo para desarrollar los módulos (Pantalla, Wi-Fi, Bluetooth, Radio, GPS) en Android. La Figura 28 representa al repositorio de Google Git; esta define valores constantes para la intensidad de corriente relacionado con el consumo del Bluetooth y GPS en miliamperios (mA).

Por ejemplo cuando el Bluetooth está activo (Línea 33) y transfiriendo información se tiene una intensidad corriente de 10 mA y si está conectado o desconectado consume 0,1 mA. En el caso del GPS cuando esta encendido, se tiene un valor de 50 mA.

```
20  <device name="Android">
21    <!-- Most values are the incremental current used by a feature,
22         in mA (measured at nominal voltage).
23         The default values are deliberately incorrect dummy values.
24         OEM's must measure and provide actual values before
25         shipping a device.
26         Example real-world values are given in comments, but they
27         are totally dependent on the platform and can vary
28         significantly, so should be measured on the shipping platform
29         with a power meter. -->
30    <item name="none">0</item>
31    <item name="screen.on">0.1</item>  <!-- ~200mA -->
32    <item name="screen.full">0.1</item>  <!-- ~300mA -->
33    <item name="bluetooth.active">0.1</item> <!-- Bluetooth data transfer, ~10mA -->
34    <item name="bluetooth.on">0.1</item>  <!-- Bluetooth on & connectable, but not connected, ~0.1mA -->
35    <item name="wifi.on">0.1</item>  <!-- ~3mA -->
36    <item name="wifi.active">0.1</item>  <!-- WIFI data transfer, ~200mA -->
37    <item name="wifi.scan">0.1</item>  <!-- WIFI network scanning, ~100mA -->
38    <item name="dsp.audio">0.1</item> <!-- ~10mA -->
39    <item name="dsp.video">0.1</item> <!-- ~50mA -->
40    <item name="camera.flashlight">0.1</item> <!-- Avg. power for camera flash, ~160mA -->
41    <item name="camera.avg">0.1</item> <!-- Avg. power use of camera in standard usecases, ~550mA -->
42    <item name="radio.active">0.1</item> <!-- ~200mA -->
43    <item name="radio.scanning">0.1</item> <!-- cellular radio scanning for signal, ~10mA -->
44    <item name="gps.on">0.1</item> <!-- ~50mA -->
45    <!-- Current consumed by the radio at different signal strengths, when paging -->
46    <array name="radio.on"> <!-- Strength 0 to BINS-1 -->
47        <value>0.2</value> <!-- ~2mA -->
```

Figura 28. Valores para módulos de Bluetooth y GPS

Tomado de (Google Kit, 2013)

2.1.6.2. Escenarios Bluetooth

Según se indica en el apartado anterior, el Bluetooth parte con valores iniciales de intensidad de corriente de 0,1 mA cuando esta encendido y 10 mA cuando esta emparejado con otro dispositivo. Así mismo, se tiene que la potencia máxima es de 40mW cuando está transfiriendo información y de 0.4mW cuando esta emparejado.

Con estas consideraciones la Figura 29 muestra un escenario general, el cual sirve para los cuatro escenarios de pruebas a realizarse, siendo el cuarto escenario el que permite obtener mediciones desde la aplicación y determina el consumo de energía de la batería, al enviar desde el dispositivo móvil un archivo de audio, video hacia un equipo portátil.

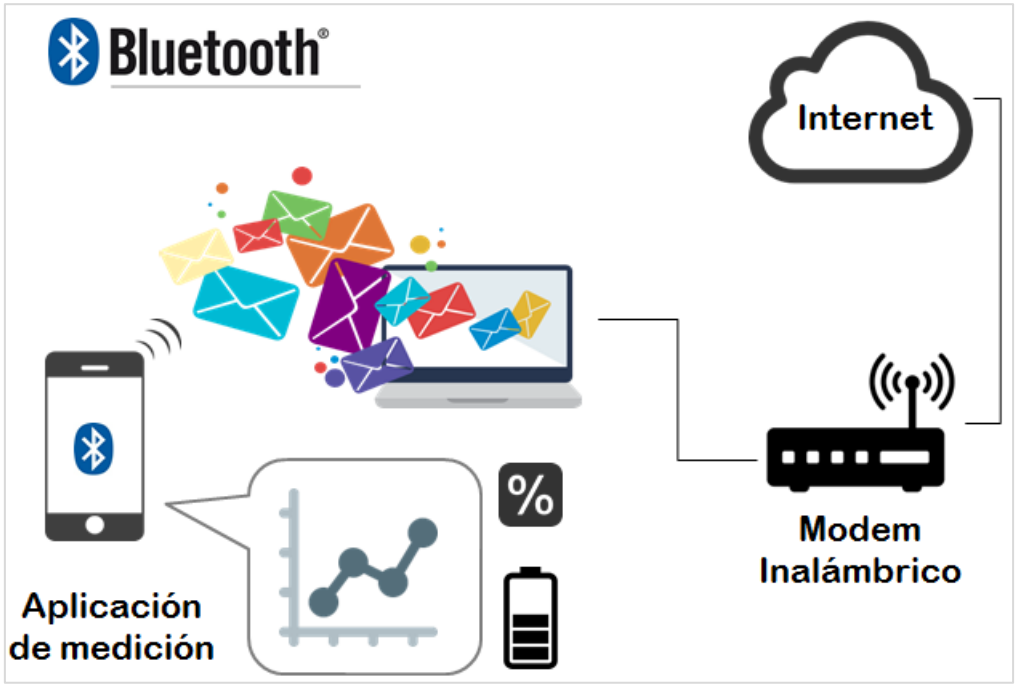


Figura 29. Escenario general para pruebas de Bluetooth

Más adelante se presenta los diferentes escenarios, datos y gráficas para el componente Bluetooth.

2.1.6.2.1. Escenario 1: Apagado

Para este escenario, los valores de Intensidad de corriente (mA) es cero (0) y de voltaje es de 4V, por lo tanto, la potencia consumida es igual a 0mW. Cabe señalar que el valor del voltaje se toma de los datos que arroja el sistema operativo respecto a la batería, el mismo que tendrá variaciones producto del desgaste y uso que tenga la batería.

Tabla 11. Bluetooth apagado

Descripción	
Componente	Estado
Bluetooth	Apagado
Intensidad (I)	0 mA
Voltaje (V)	4 V
Potencia (P)	0 mW

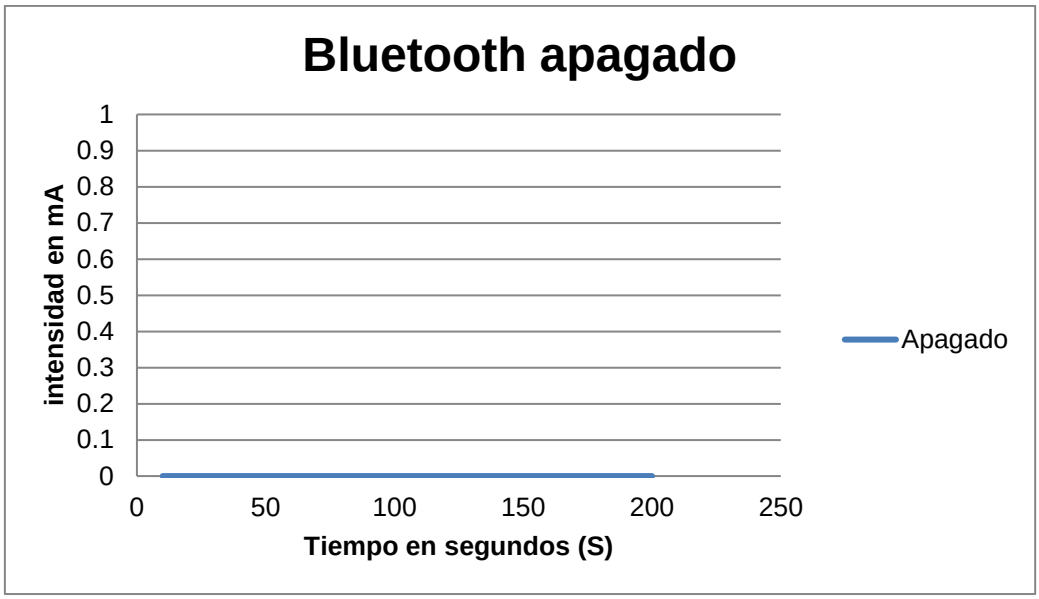


Figura 30. Bluetooth apagado

Según se observa la Figura 30, el Bluetooth en el estado apagado no consume batería. Los resultados obtenidos de potencia consumida se los expondrá a través de la aplicación de medición más adelante.

2.1.6.2.2. Escenario 2: Encendido

En esta ocasión el componente se encuentra encendido y la intensidad de corriente arroja el valor de 0,1 mA, independientemente de que este o no conectado (emparejado) con otro dispositivo. Se mantiene el valor del voltaje de la batería y la potencia consumida en este estado es de 0,4 mW. Este último valor es el que se mostrará en los gráficos de la aplicación de medición.

Tabla 12. Bluetooth encendido

Descripción	
Componente	Estado
Bluetooth	Encendido
Intensidad (I)	0,1 mA
Voltaje (V)	4 V
Potencia (P)	0,4 mW

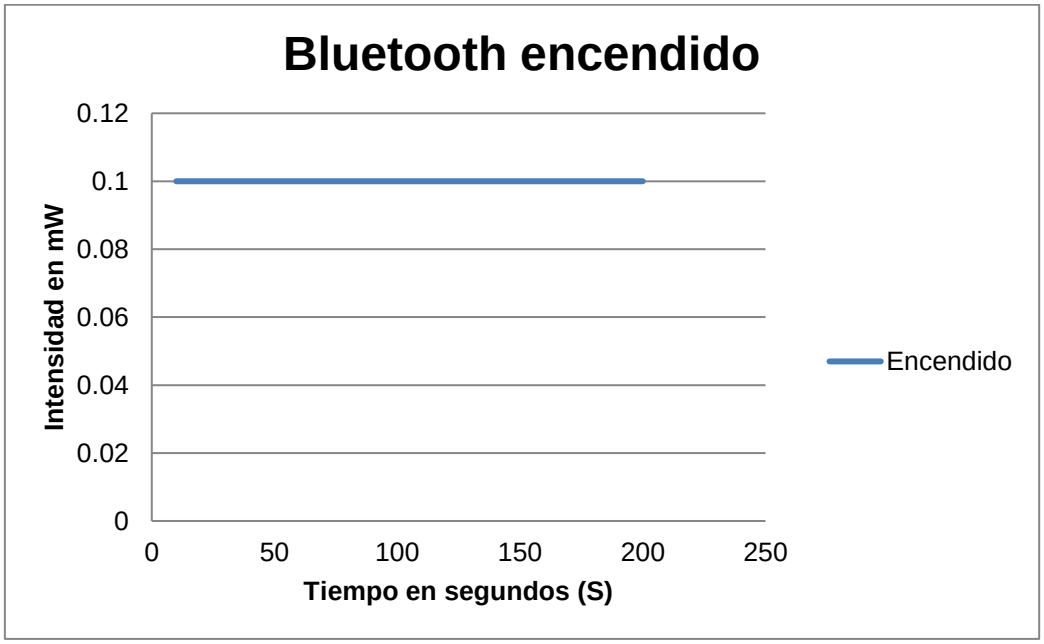


Figura 31. Bluetooth encendido

La Figura 31, muestra en el tiempo una intensidad de corriente de 0,1mA, este valor seguirá siendo constante mientras no cambie al estado de transferencia.

2.1.6.2.3. Escenario 3: Encendido y en paridad

Este escenario recoge los estados anteriores, partiendo con un valor mínimo de 0,1 mA de intensidad de corriente, luego llega al valor de 10 mA al emparejarse con un dispositivo y finalmente cuando ya no está emparejado regresa al valor inicial de 0,1 mA.

Tabla 13. Bluetooth encendido y en paridad

Descripción	
Componente	Estado
Bluetooth	Encendido y en paridad
Intensidad (I)	0,1 mA; 10 mA; 0,1 mA
Voltaje (V)	4 V
Potencia (P)	40 mW

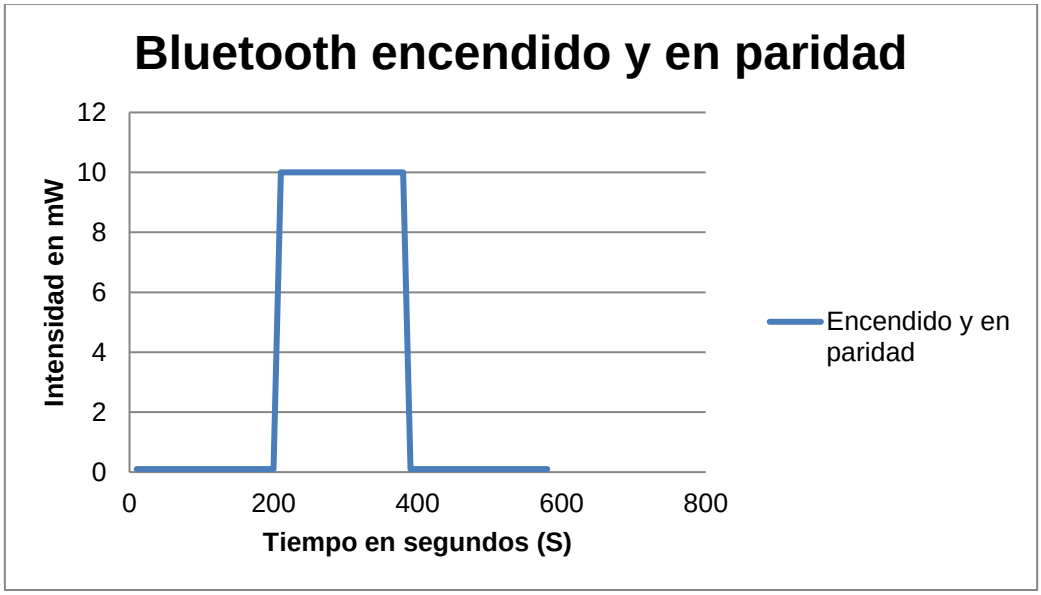


Figura 32. Bluetooth encendido y en paridad

En esta prueba el Bluetooth realizó el proceso de emparejarse y desemparejarse, una vez que transfiere información.

2.1.6.2.4. Escenario: 4 Encendido, en paridad y trasfiriendo

Para este escenario de prueba, la batería tiene un comportamiento diferente y empieza a descargarse en función del tiempo que el Bluetooth se encuentre

transfiriendo información. La potencia máxima consumida en este escenario es la suma de las demás potencias iniciales previo al envío de información y durante el tiempo que dure esta actividad.

Tabla 14. Bluetooth encendido, en paridad y transfiriendo

Descripción	
Componente	Estado
Bluetooth	Encendido, en paridad y transfiriendo
Intensidad (I)	0,1 mA; 10 mA;
Voltaje (V)	4 V
Potencia (P)	0,4 mA; 40 mW

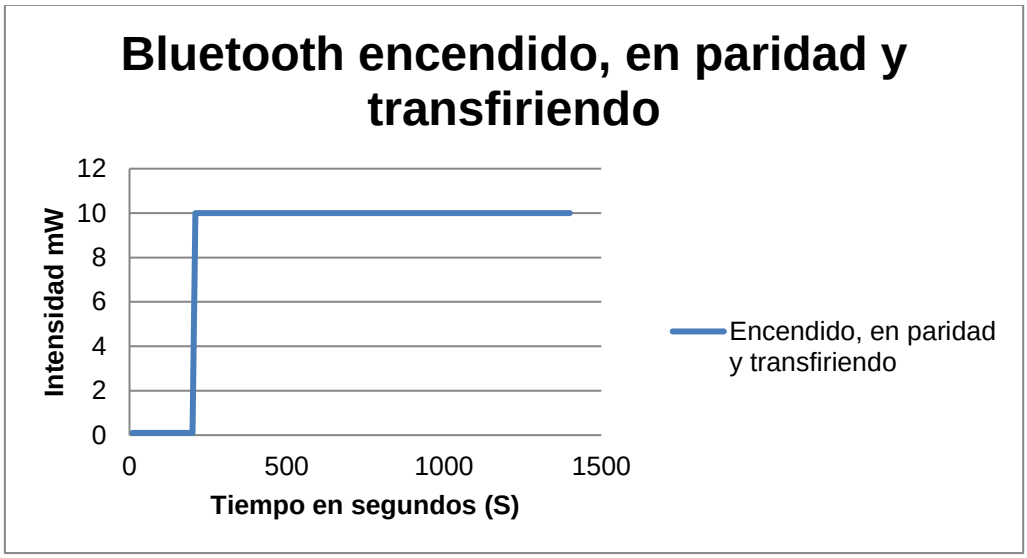


Figura 33. Bluetooth encendido, en paridad y transfiriendo

En esta prueba se observa que la intensidad de corriente se mantiene constante entre 10 y 11 mW durante el tiempo que dure la misma.

Finalmente el resultado de esta medición se suma al resultado del componente del GPS y se obtiene el consumo total de energía.

2.1.6.3. Escenarios GPS

Del mismo modo que el componente anterior, partimos de los valores por defecto que se mencionó para desarrollo de módulos en Android. Se tiene una intensidad de corriente de 50 mA cuando el GPS está activo y de 4V que corresponde a la batería del dispositivo.

La Figura 34 representa el escenario general para las pruebas de este componente.

Figura 34. Escenario general para pruebas de GPS

- Para el módulo GPS se obtiene el valor ideal de cincuenta mili Amperios [50mA].
- GPS tiene un valor de potencia máximo que es el umbral, siendo este cuando este cargado 100% con un valor de 200mW.

2.1.6.3.1. Escenario 1: Apagado

El valor de Intensidad de corriente (mA) es cero y el voltaje (V) es 4V, por lo tanto, la potencia consumida es igual a 0mW.

Los 4V corresponden al valor que entrega el sistema operativo respecto a la batería.

Tabla 15. GPS apagado

Descripción	
Componente	Estado
GPS	Apagado
Intensidad (I)	0 mA
Voltaje (V)	4 V
Potencia (P)	0 mW

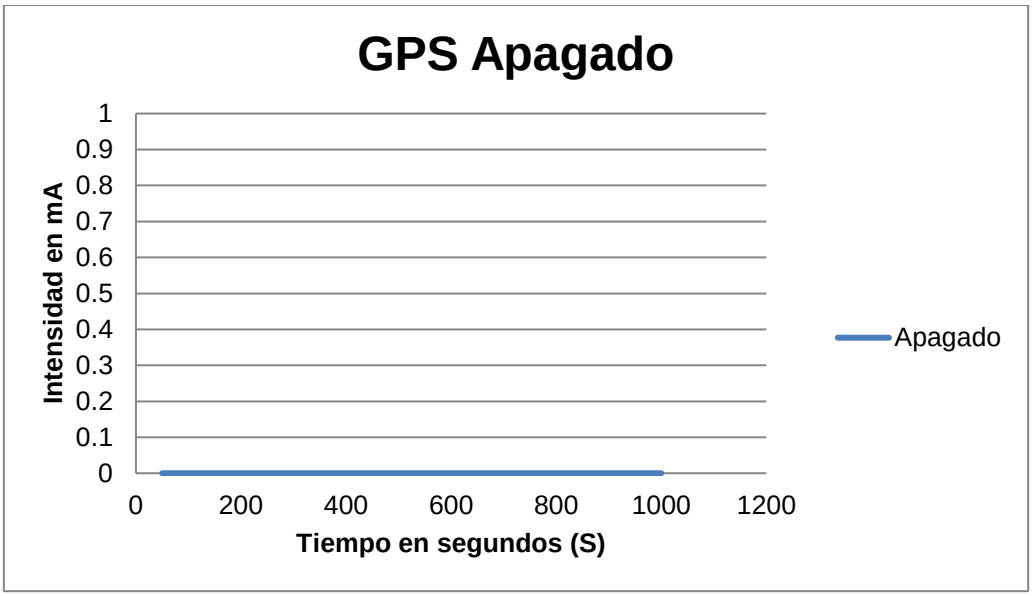


Figura 35. GPS apagado

2.1.6.3.2. Escenario 2: Encendido y cargado al 100%

Para este escenario partimos con una carga del dispositivo al 100%, el valor de intensidad de corriente de 50 mA y el voltaje de 5V.

Una vez que se enciende el componente GPS, se establece la comunicación con el satélite y se observa a través de la gráfica del dispositivo móvil el comportamiento que tiene la batería. Adicional se utiliza una APP (Runkeeper) de Google Play que permite definir una ruta para entrenar.

Tabla 16. GPS encendido y cargando

Descripción	
Componente	Estado
GPS	Encendido y cargado al 100%
Intensidad (I)	50 mA
Voltaje (V)	4 V
Potencia (P)	200 mW

Con esta APP funcionando se procede a realizar medidas, se observa que el valor de potencia consumida está en el rango de los 130 y 200 mW. Sin embargo al llegar al nivel al 50% de carga del dispositivo, dicha potencia llega a la mitad.

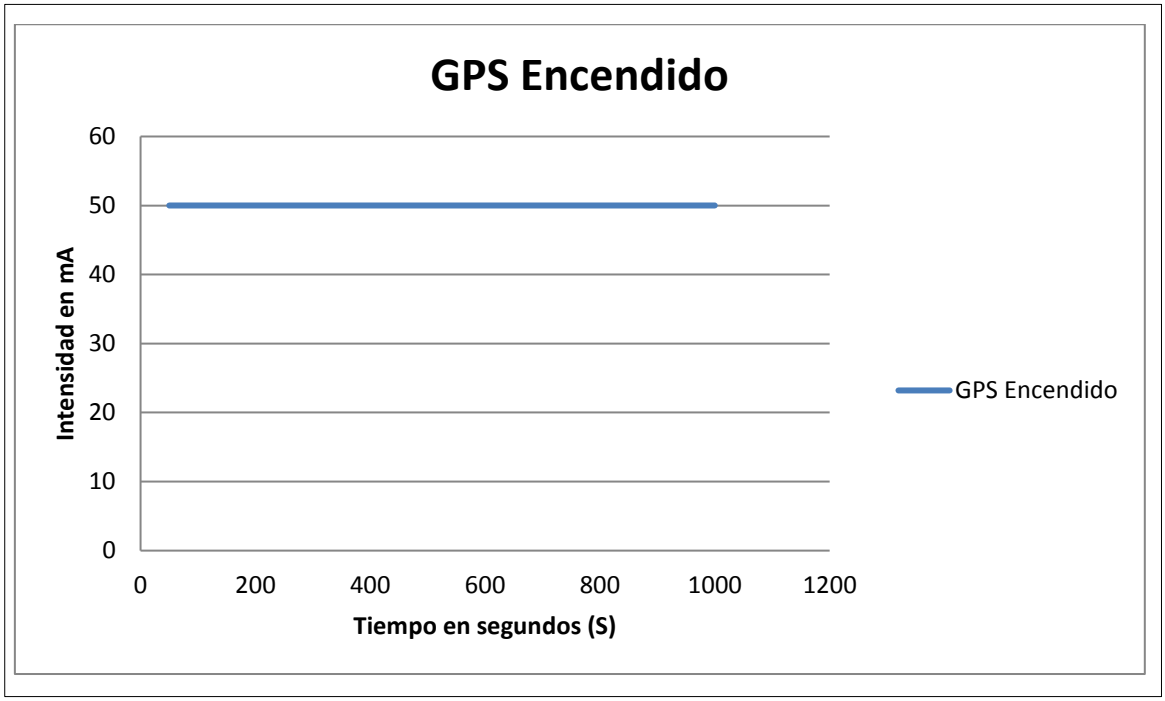


Figura 36. GPS encendido y descargando

2.1.7. Desarrollo de la aplicación

Esta sección presenta el diseño, instalación, configuración, selección de paquetes y objetos para la aplicación de monitor de consumo de energía.

2.1.7.1. Diseño

La aplicación de monitoreo de consumo de energía se conforma de diez (10) módulos. La Figura 37 presenta el modelado para dicha aplicación.

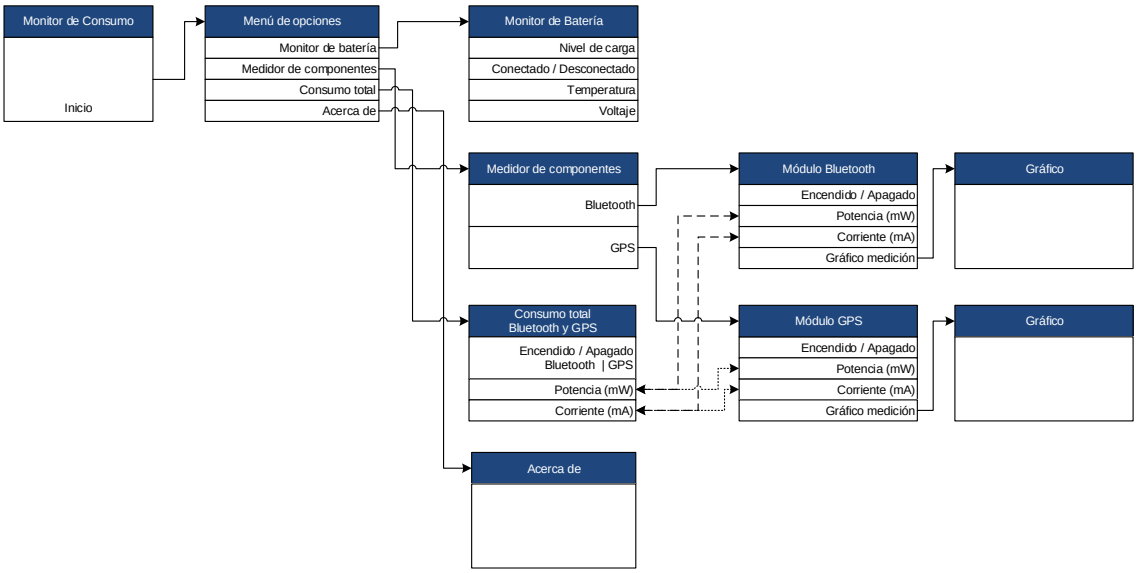


Figura 37. Modelado de la aplicación

2.1.7.2. Instalación y configuración Android Studio

- Descargar e instalar la versión 2.1 de Android Studio y el JDK versión 1.8 (Java Development Kit) para Windows 7 de 64 bits. El proceso de instalación se encuentra descrito en el Anexo1.
- Registrar la variable de entorno para que Android Studio conozca donde se guarda la máquina virtual, ver Anexo 2.
- Crear el proyecto mediante el asistente de Android Studio, los pasos de configuración se encuentran en el Anexo 3
- Ejecutar la herramienta SDK Manager

2.1.7.3. Paquetes y objetos

Para el desarrollo de la aplicación se define varias interfaces que interactúan con los componentes de Bluetooth y GPS.

El Anexo 5, presenta los paquetes y objetos que se utiliza en este desarrollo:

2.1.7.4. Construcción de la aplicación

Una vez finalizado las configuraciones mencionadas en los numerales anteriores, se inicia el programa y se presenta el IDE de desarrollo de Android Studio (ver Figura 38).

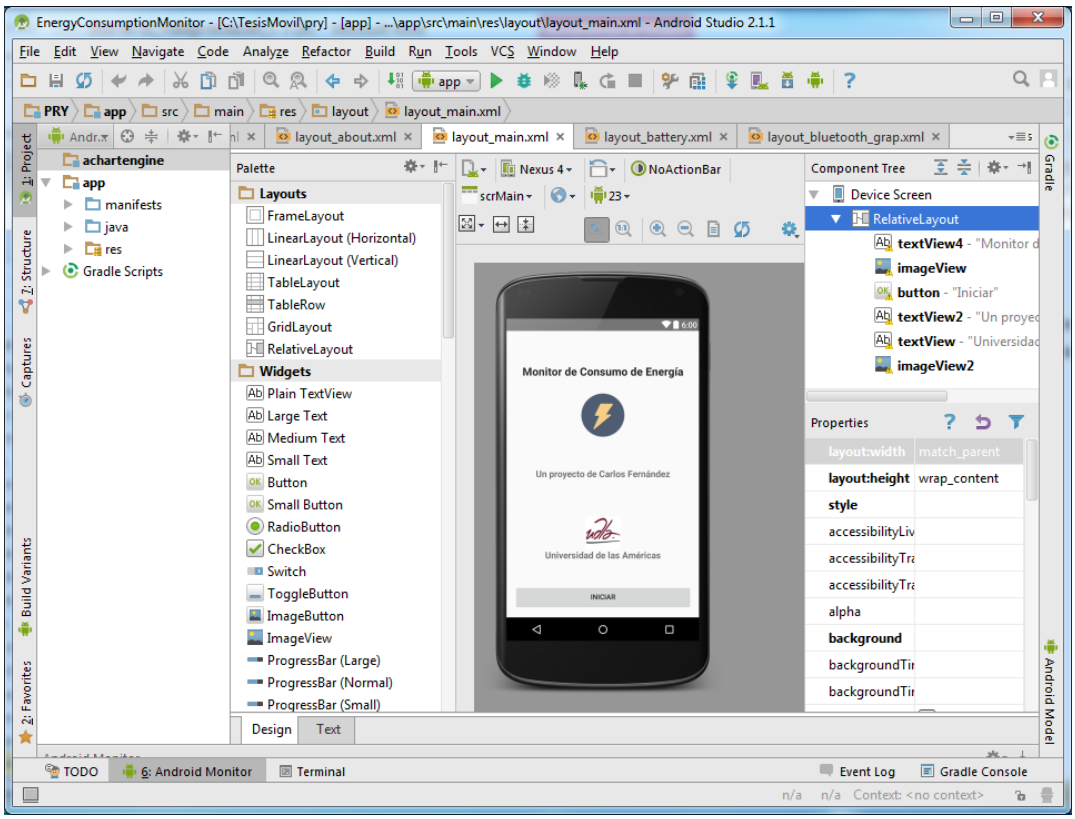


Figura 38. IDE de Android Studio

2.1.7.4.1. Estructura del proyecto

En la ventana de herramientas del panel del izquierdo del IDE se encuentra el proyecto a trabajar (ver Figura 39).

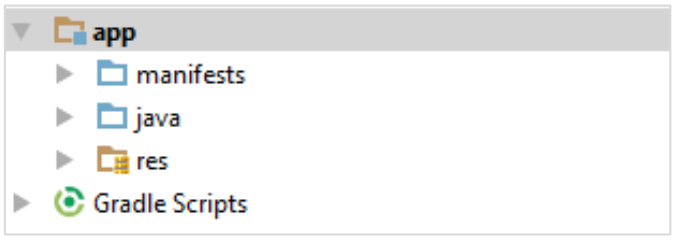


Figura 39. Ventana de herramienta del proyecto

2.1.7.4.2. App

La carpeta **app** está compuesta por la carpeta **manifests** que tiene las configuraciones por defecto, seguido de la carpeta **java** que tiene el código fuente y por último la carpeta **res** que contiene lo que son imágenes, texto, etcétera.

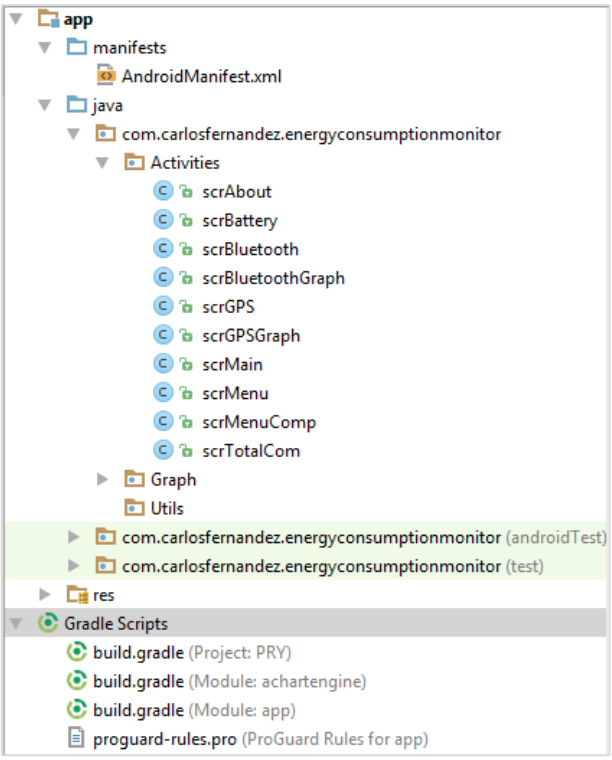


Figura 40. Estructura de la carpeta app

2.1.7.4.3. Manifests

En esta carpeta se encuentra el archivo AndroidManifest.xml, con las configuraciones de los componentes Bluetooth y GPS.

Java maneja un estándar para los nombres, conocido como espacio de nombres, se escribe en minúscula, se antepone las letras “com.”, seguido del nombre de la empresa y por último el nombre del aplicativo.

Como se puede observar en la Figura 41, el nombre del paquete para esta aplicación es "com.carlosfernandez.energyconsumptionmonitor".

```
manifest  receiver
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.carlosfernandez.energyconsumptionmonitor">

    <uses-permission
        android:name="android.permission.BLUETOOTH"
        android:required="false" />
    <uses-permission
        android:name="android.permission.BLUETOOTH_ADMIN"
        android:required="false" />
    <uses-permission
        android:name="android.permission.ACCESS_FINE_LOCATION"
        android:required="false" />
    <uses-permission android:name="android.permission.WRITE_SETTINGS" />

    <receiver android:name=".Activities.scrBattery">
        <intent-filter>
            <action android:name="android.intent.action.ACTION_POWER_CONNECTED" />
            <action android:name="android.intent.action.ACTION_POWER_DISCONNECTED" />
        </intent-filter>
    </receiver>
    <receiver android:name=".Activities.scrBluetooth">
        <intent-filter>
            <action android:name="android.bluetooth.adapter.action.STATE_CHANGED" />
            <action android:name="android.bluetooth.device.action.ACL_CONNECTED" />
            <action android:name="android.bluetooth.device.action.ACL_DISCONNECTED" />
        </intent-filter>
```

Figura 41. Código Manifiests

2.1.7.4.4. Java

Java contiene la carpeta **Activities** que en otras palabras son las Pantallas de la aplicación; se divide en parte lógica que es el código (archivo .java) que esta por detrás de cada pantalla y que les permite interactuar entre ellas. La otra es la gráfica (XML) que contiene los elementos de la pantalla. La Figura 42 señala que se crearon 10 Activities, una de ellas es **srcMenu** que representa al **Menú de opciones** de la aplicación.

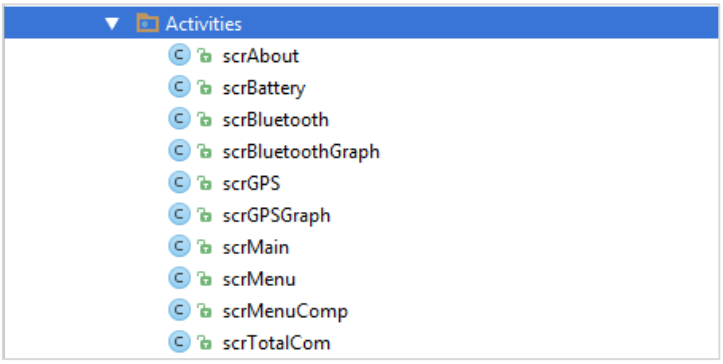


Figura 42. Listado de Activities

2.1.7.4.5. Layout

Desde esta opción se diseñó la interfaz de cada una de las actividades definidas en los apartados que anteceden. La Figura 43 describe la estructura que tiene la carpeta **res** del proyecto. La carpeta **layout** tiene los archivos de diseño de las actividades y guarda las definiciones XML.

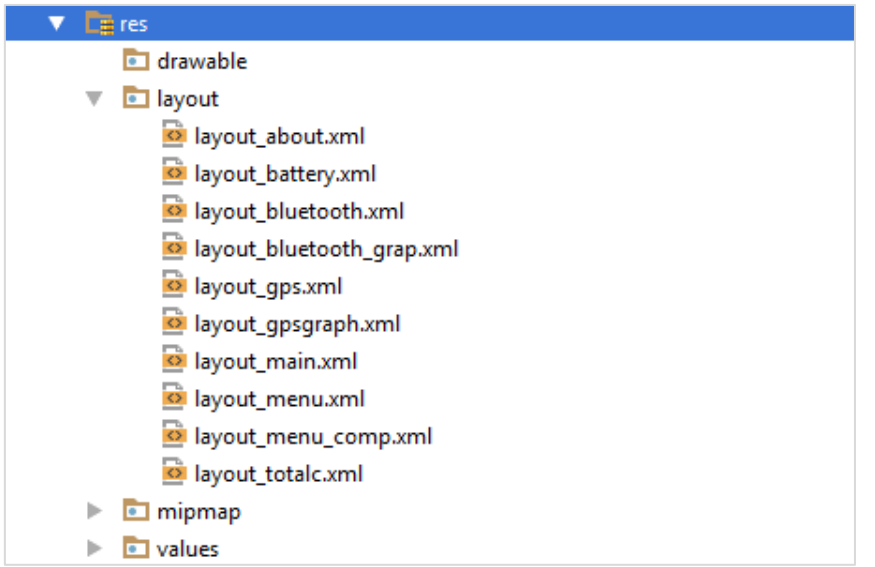


Figura 43. Layouts de la aplicación

2.1.7.4.6. Emulador

Una vez que se termina el desarrollo de la aplicación, se tiene tres formas para ejecutar la misma.

En primer lugar, se compila la aplicación para convertirlo en un ejecutable de tal forma que cargue en la tarjeta de memoria y luego se instale en el dispositivo móvil como cualquier aplicación.

La Figura 44 muestra la instalación del ejecutable cuya extensión es “.apk” en el dispositivo móvil.

Figura 44. Archivo ejecutable instalado

La segunda forma es poner al dispositivo móvil en modo desarrollo, esto hace que se ejecute en el dispositivo móvil. Cabe señalar que depende del sistema operativo tenga los drivers correspondientes considerando que el teléfono es un dispositivo de almacenamiento.

Es importante mencionar que las dos primeras formas no permiten realizar modificaciones en el código fuente.

Finalmente se tiene el emulador de Android Studio (ver Figura 45), que permite ver el diseño final y el funcionamiento de los botones; para esto se carga la imagen del sistema operativo y seguidamente se simula el dispositivo móvil, sin embargo al querer tomar medidas como el estado de la batería, el emulador no da información real.

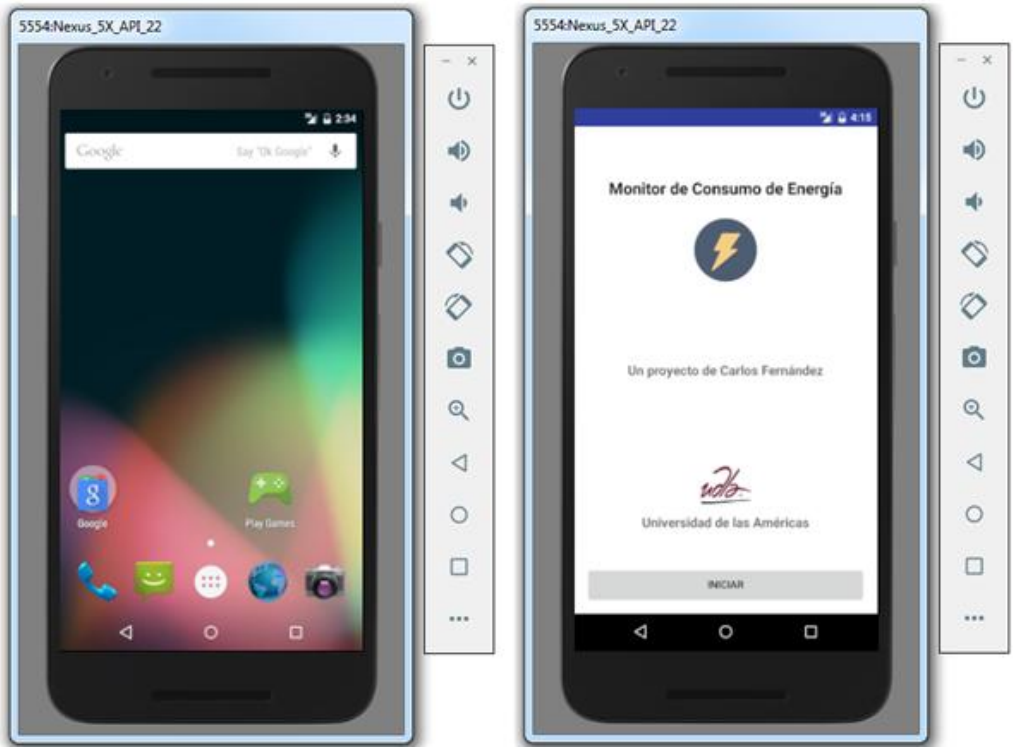


Figura 45. Emulador de la aplicación

2.1.7.5. Módulos

Se presenta una descripción general de la interacción que tienen los diferentes módulos del Monitor de consumo de energía.

2.1.7.5.1. Monitor de Consumo de energía

Representa la pantalla principal del inicio de la aplicación, se compone de una sección informativa y del botón que indexa al resto de módulos.

- Nombre de la aplicación
- Autor
- Universidad
- Botón Iniciar

La Figura 46, muestra el diseño y el código del desarrollo de la misma:

Figura 46. Pantalla - Monitor de consumo de energía

2.1.7.5.2. Menú de opciones

Menú de opciones es la pantalla de entrada principal a la aplicación, consta de módulos: Monitor de la batería, Medidor de componentes, Consumo total y Acerca de, tal como se muestra en la Figura 47.

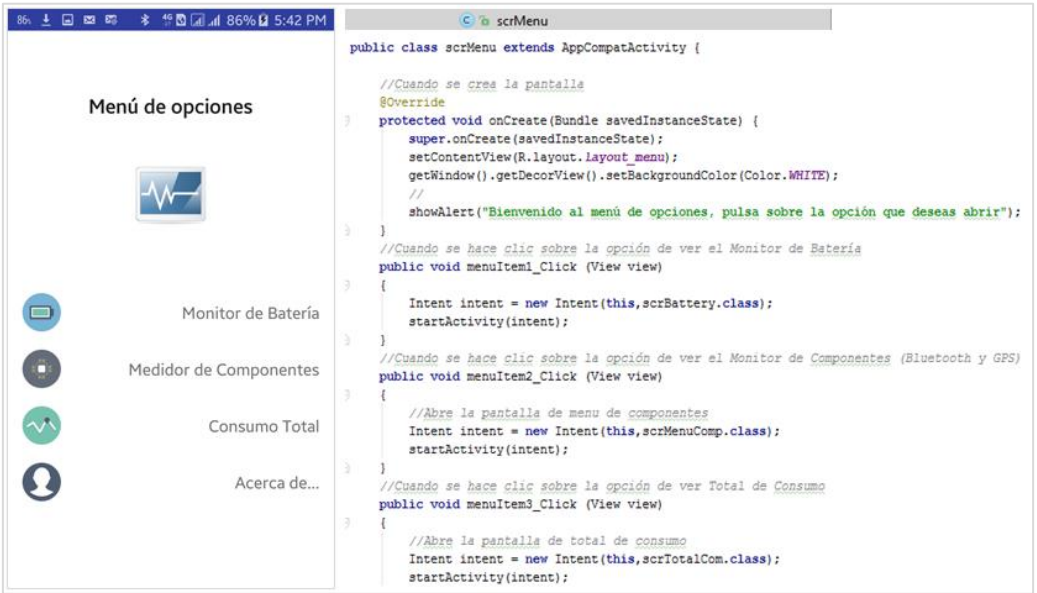


Figura 47. Pantalla – Menú de opciones

2.1.7.5.3. Monitor de batería

Muestra la información del nivel de carga de la batería en porcentaje. Luego indica si el dispositivo se encuentra conectado a una fuente de energía, como por ejemplo vía USB.

Seguidamente provee de la temperatura en grados centígrados y finalmente entrega el voltaje de la batería. Todos estos valores permiten conocer el estado de la batería antes de iniciar las mediciones de los componentes.

- Nivel de carga de la batería del dispositivo en %
- Indica si el dispositivo está cargándose o no
- Temperatura de la batería en °C
- Voltaje de la batería

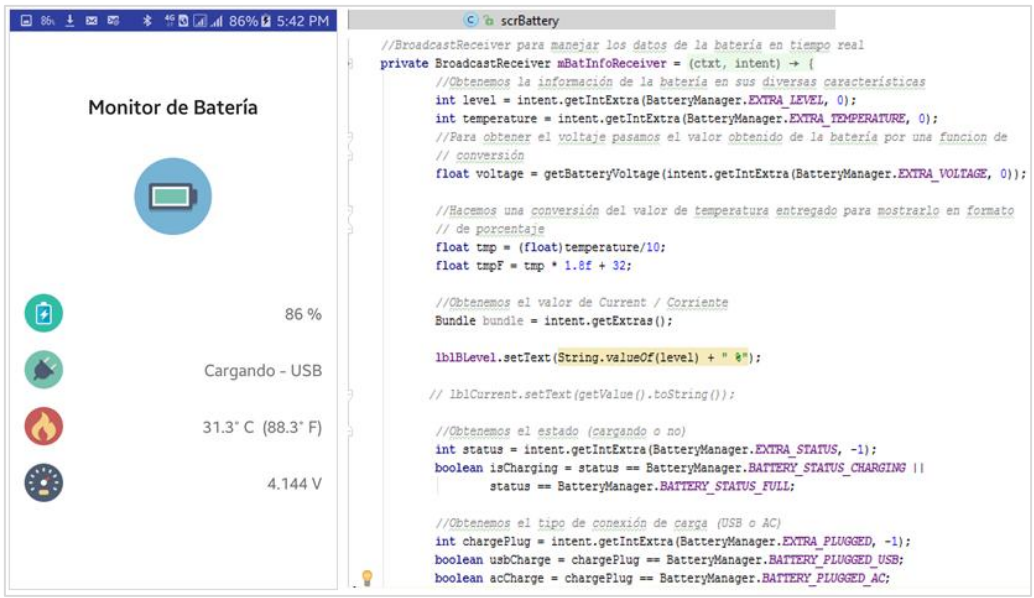


Figura 48. Pantalla – Monitor de batería

2.1.7.5.4. Medidor de componentes

La pantalla principal de la aplicación está compuesta por las opciones de Bluetooth y GPS. Para ambos componentes se muestra las medidas del consumo de energía (mW), partiendo del estado del módulo de cada uno y su amperaje correspondiente, estando activo y en transferencia (mA). Para el cálculo de la potencia sobre la carga se utiliza la fórmula: P=V*I

Los resultados que se muestran en los dos componentes son los siguientes:

- Bluetooth
- GPS

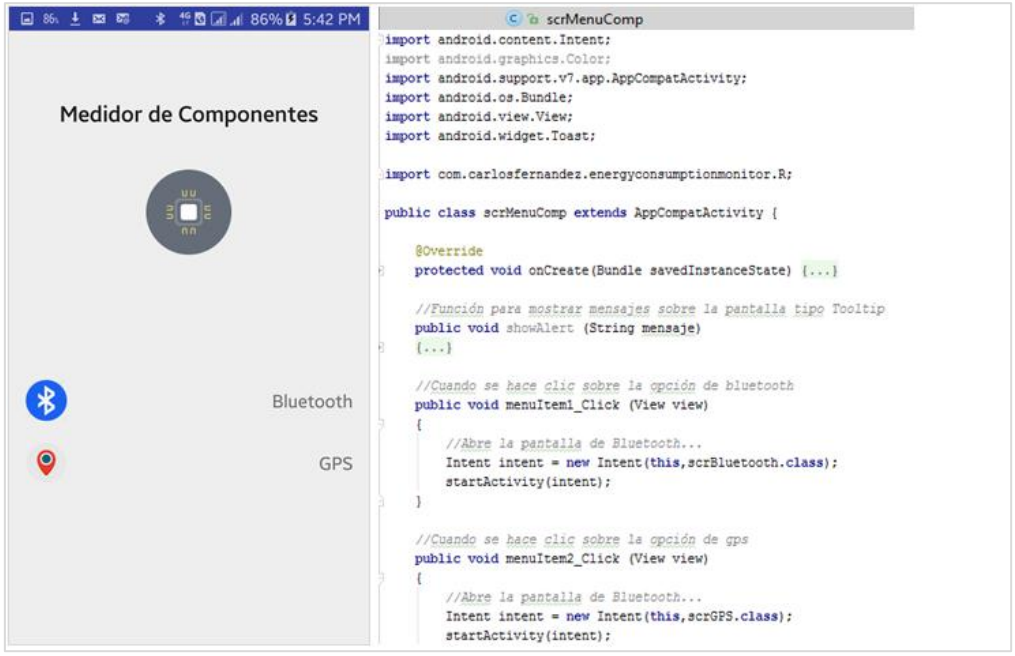


Figura 49. Pantalla – Medidor de componentes Bluetooth y GPS

2.1.7.5.5. Bluetooth

Para este componente se define las siguientes opciones:

- Estado del Bluetooth: Encendido/Apagado
- Potencia del Bluetooth sobre la carga en mW
- Corriente emitida en mA
- Gráfico de consumo: Eje de las (Y) Valor de la medida en (mW); y en el eje de las (X) el tiempo de medición en segundos.

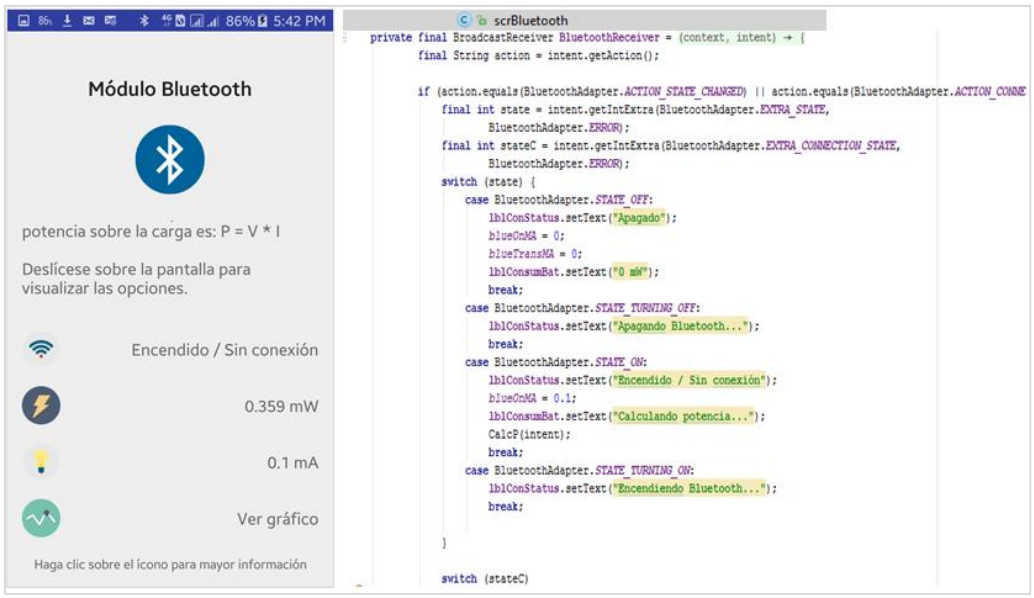


Figura 50. Pantalla - Bluetooth

2.1.7.5.6. Gráfico Bluetooth

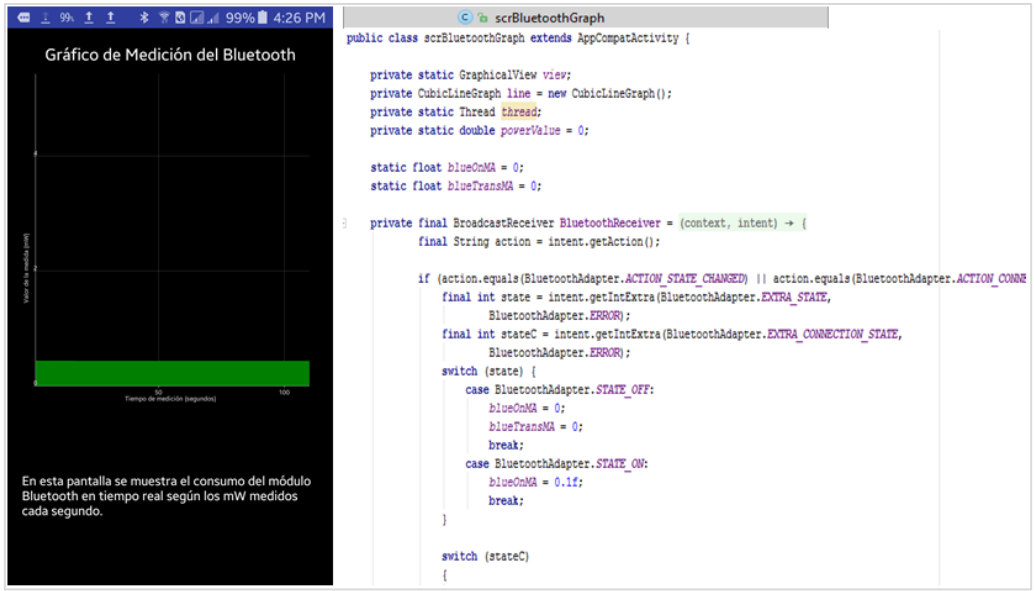


Figura 51. Pantalla – Gráfico Bluetooth

2.1.7.5.7. GPS

Para este componente se define las siguientes opciones:

- Estado del GPS: Activado/Desactivado
- Potencia del GPS sobre la carga en mW
- Corriente emitida en mA

- Gráfico de consumo: Eje de las (Y) Valor de la medida en (mW); y en el eje de las (X) el tiempo de medición en segundos.

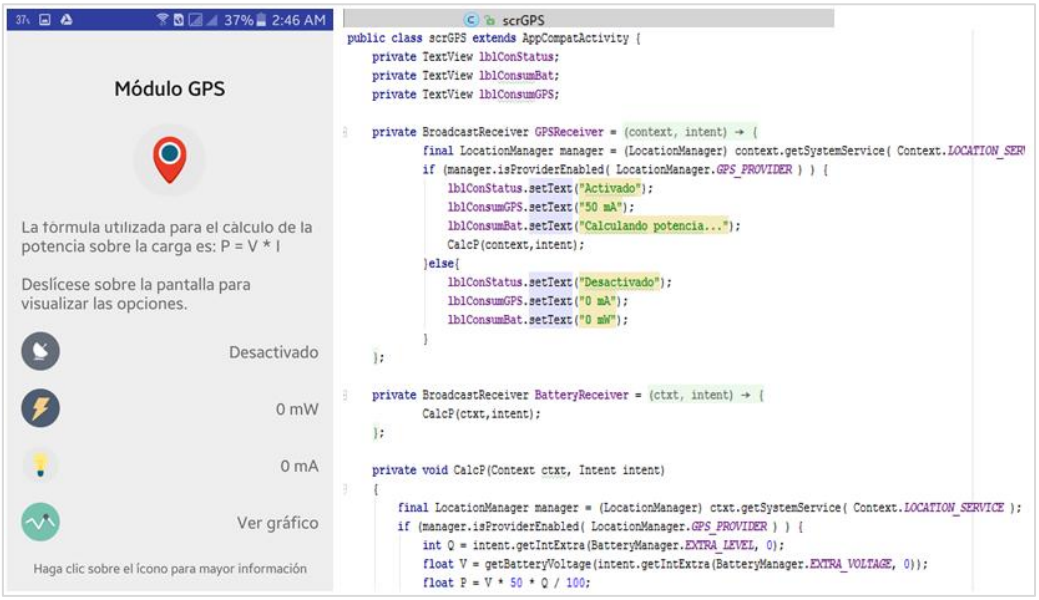


Figura 52. Pantalla – GPS

2.1.7.5.8. Gráfico GPS

Figura 53. Pantalla – Gráfico GPS

2.1.7.5.9. Consumo total Bluetooth y GPS

En esta parte se muestra el consumo total del consumo de energía de ambos componentes. Los resultados que se presentan son los siguientes:

- Componentes: GPS Encendido/apagado | Bluetooth Encendido/apagado
- Potencia del Bluetooth y GPS sobre la carga en mW
- Corriente emitida por Bluetooth y GPS en mA

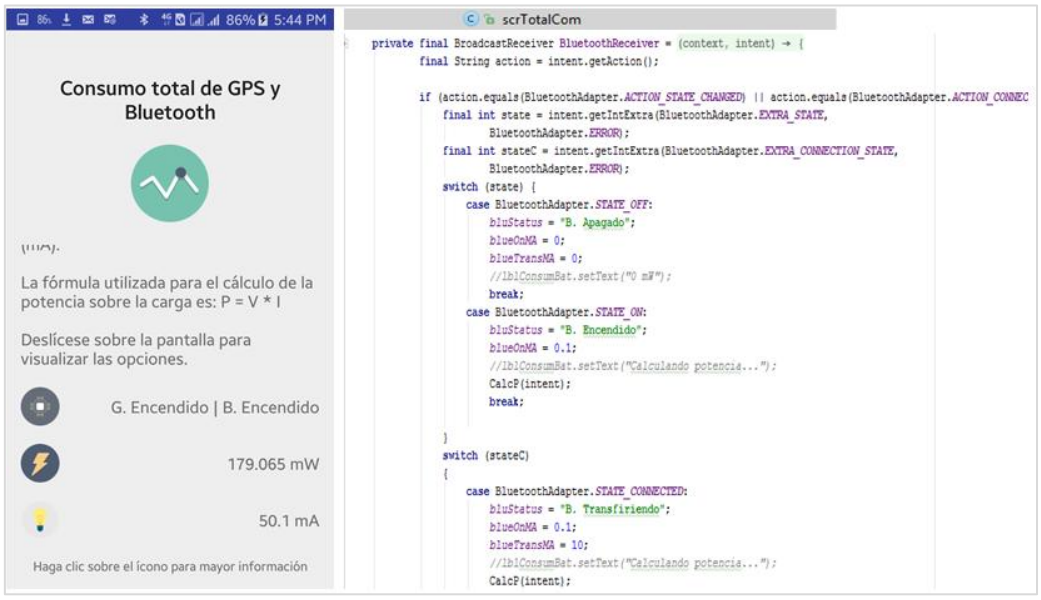


Figura 54. Pantalla – Consumo total Bluetooth y GPS

2.1.7.5.10. Acerca de

Describe de manera general lo que realiza la aplicación en los componentes de Bluetooth y GPS.

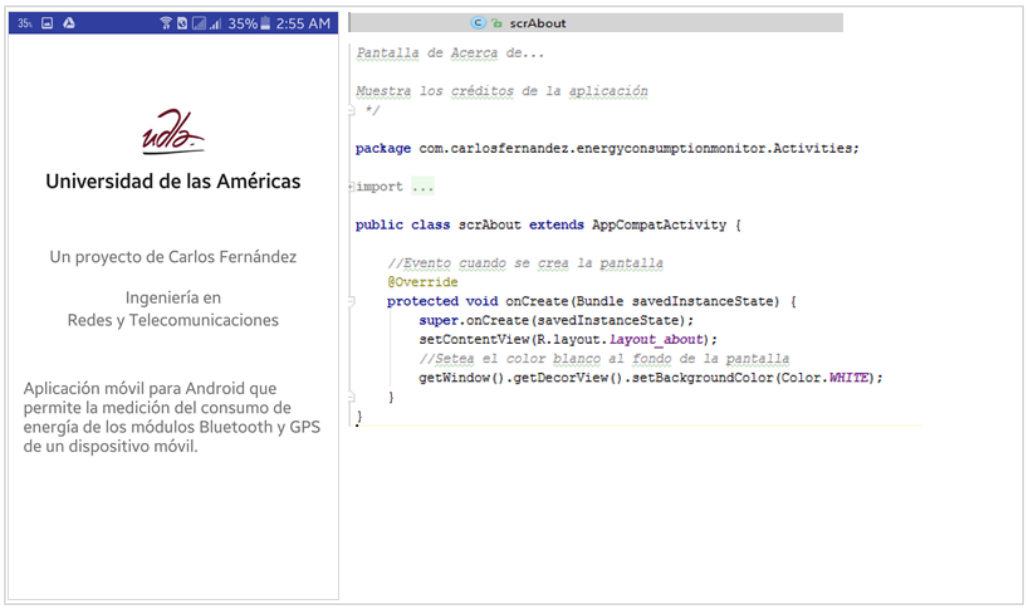


Figura 55. Pantalla – Créditos de la aplicación

3. CAPÍTULO III. MEDICIONES DE POTENCIA

3.1. Medidas de los componentes Bluetooth y GPS

El proceso de medición para el dispositivo móvil tiene como fin conocer el valor de potencia consumida por los componentes Bluetooth y GPS en diferentes intervalos de tiempo y niveles de carga de la batería, el tiempo es una variable a considerar para realizar las estimaciones.

La técnica a utilizar en este proceso es la “observación”; consiste en la recolección de los datos que arrojen cada uno de los componentes a través de la aplicación de monitoreo, en sus diferentes estados y condiciones particulares, seguido de su interpretación y análisis respectivo. Los resultados que se obtiene permiten conocer cuál es el componente que consume mayor energía.

3.1.1. Descarga de la batería en estado suspendido

Se analiza en primer lugar el estado de la batería en el nivel máximo de carga (100%) sin que los componentes de Bluetooth y GPS estén activos y el dispositivo móvil tenga el estado suspendido. Estos valores se los requiere como medida inicial. Para este análisis se tomaron muestras cada dos horas y se constató que la descarga de la batería se realiza en un 2,5% en estado suspendido. El Anexo 8 contiene los datos obtenidos de:

- Los datos de tiempo de descarga (horas),
- Nivel de carga % dispositivo
- Tiempo descarga del CPU
- Tiempo descarga pantalla
- Voltaje (V)
- Temperatura °C

Figura 56. Batería descargada en estado suspendido.

La Figura 56 expone como la batería se descarga en el estado suspendido en aproximadamente 82 horas (alrededor de 4 días). Las mediciones se encuentran en el Anexo 8.

En conclusión, en el nivel del 100% al 85%, la descarga de la batería es pausada, sin embargo cuando llega al nivel del 15% es más rápida, esto se da por el efecto memoria que desarrolla la batería cuando se ha realizado varias cargas por debajo de este porcentaje.

3.1.2. Descarga de la batería con el CPU activo

Con el CPU activo, el proceso de descarga se incrementa en un 4,5 % cada dos horas, llegando a descargase la batería en aproximadamente 46 horas. Este resultado se da porque el procesador está trabajando y tiene varias aplicaciones corriendo en segundo plano como por ejemplo la APP para detectar sismos, el Google Play Services, entre otras.

Las mediciones se encuentran en el Anexo 8.

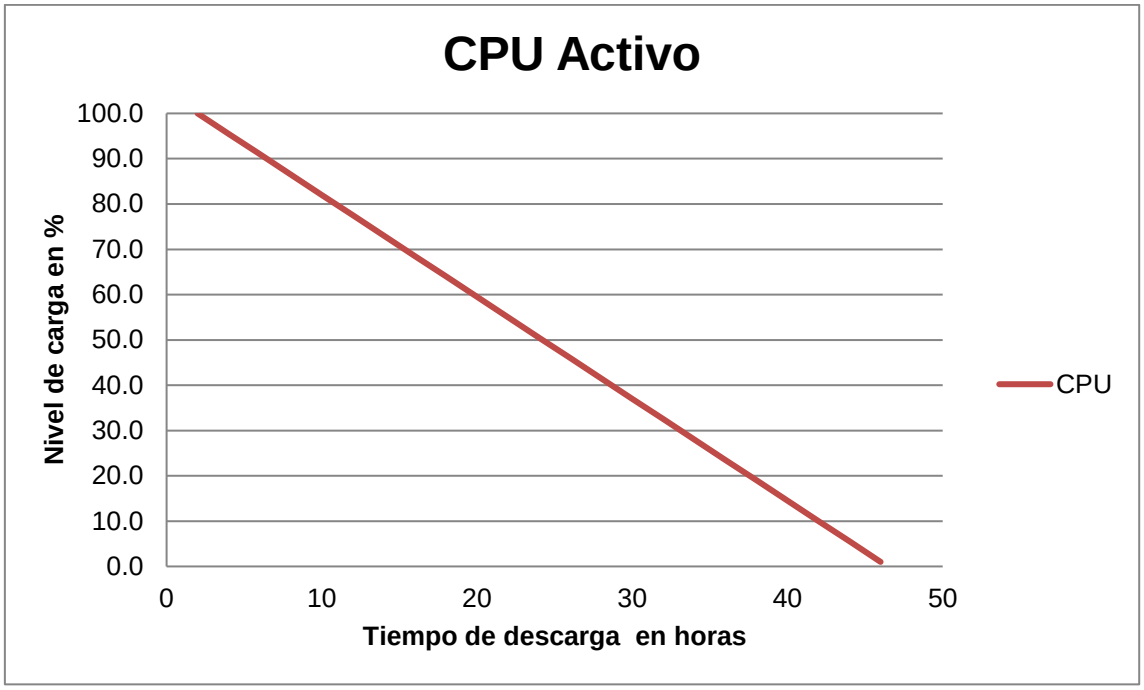


Figura 57. Batería descargada con el CPU activo

Hay que también considerar que cuando el CPU tiene períodos largos de procesamiento se incrementa el consumo de batería. Las mediciones se encuentran en el Anexo 8.

La Tabla 17 describe la velocidad del procesador (MHz), el tiempo de procesamiento y el porcentaje de descarga de la batería cuando hay procesos constantes corriendo en segundo plano.

Tabla 17. Velocidad del procesador

CPU			
MHz	Tiempo	Porcentaje	Aplicaciones
1500	11min:30s	8%	
1100	2min:37s	3%	
900	13min:11s	11%	
700	7min:22s	7%	
300	5min:53s	6%	Rastreador móvil
250	10min:33s	12%	Clima
150	40h:11min:30s	95%	Gmail

Por ejemplo en la Figura 58 se observa que existen varios procesos corriendo al mismo tiempo, de ellos tres son los que se destacan (Rastreador del móvil, Clima, Gmail), sin embargo, hay un proceso (Gmail) ejecutándose a una

velocidad del CPU de 150 MHz y se prolonga por 40 horas aproximadamente, dando como resultado la descarga de la batería al 95%.

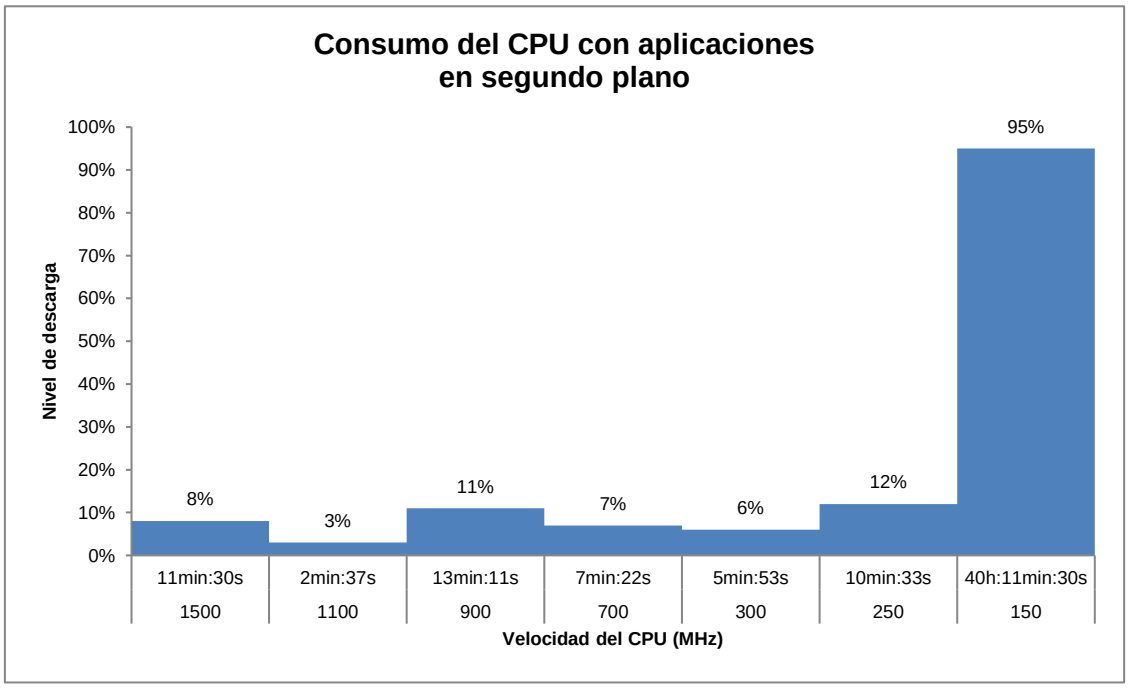


Figura 58. Aplicaciones corriendo en segundo plano

Los datos de la Figura 58 fueron obtenidos con la herramienta de diagnóstico del sistema AIDA64, como complemento para esta parte.

3.1.3. Descarga de la batería con la pantalla encendida

Continuando con las mediciones, se procede a obtener los datos de la descarga cuando se enciende la pantalla. Para esto se desactiva el bloqueo automático de la pantalla.

Se pudo constatar en las mediciones que el patrón de descarga llega a ser de 10 horas aproximadamente (ver Figura 59). Esto se debe a varios factores, por ejemplo las pantallas actuales son más grandes, el aumento del brillo, entre otros. Además el consumo del CPU junto con el de la Pantalla tiene un comportamiento similar con respecto a la descarga.

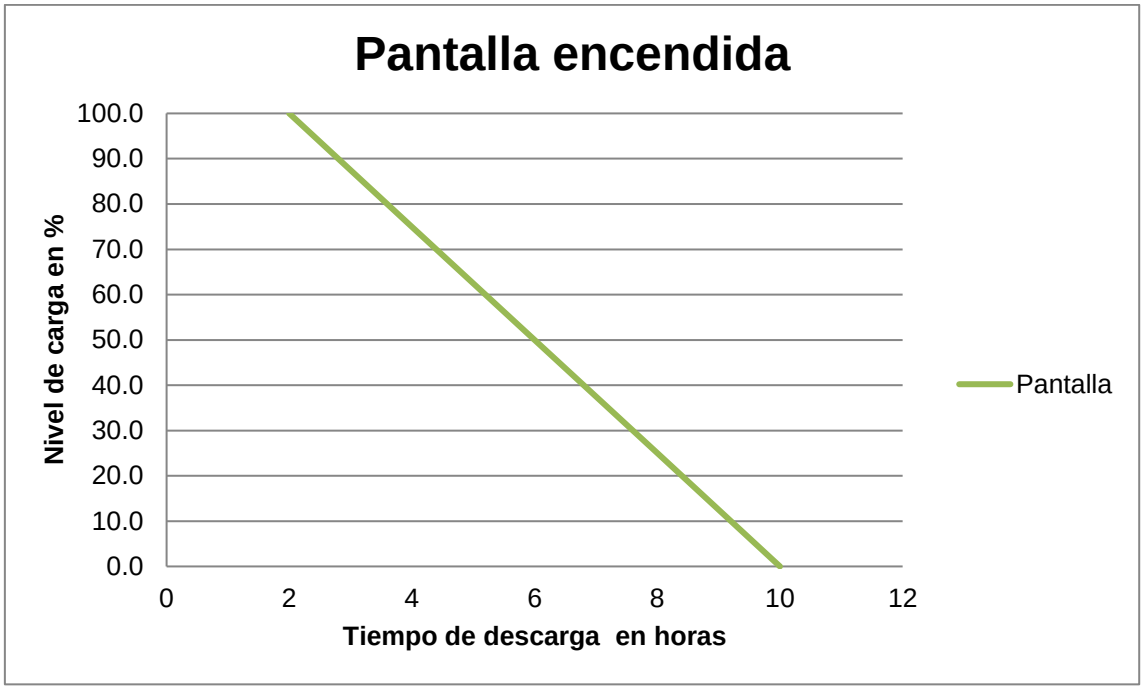


Figura 59. Batería descargada con la Pantalla encendida

3.1.4. Medición

Para medir la potencia consumida por los componentes de Bluetooth y GPS, se configuró valores iníciales en miliamperios (mA) que se mencionaron en la sección anterior (3.1.6.1 Estado de la batería). Las mediciones se encuentran en el Anexo 8.

3.1.4.1. Bluetooth

A continuación se presenta los resultados obtenidos en la medición de este componente partiendo de los siguientes datos de la batería:

- Nivel de carga de la batería del dispositivo = 72%
- Indica si el dispositivo está cargándose o no = Sin cargar
- Temperatura batería = 34 °C
- Voltaje de la batería = 4,019 V

Los datos que se muestran en la pantalla del monitor de batería (ver Figura 60) se los obtiene usando la clase **scrBattery**, tiene las funciones correspondientes para obtener el nivel máximo de la temperatura, voltaje, corriente, además indicar si el dispositivo está cargándose o no. Para este fin se utilizó el

componente **BroadcastReceiver** de Android para registrar todos los eventos en tiempo real ocurridos en el sistema.

Bluetooth apagado

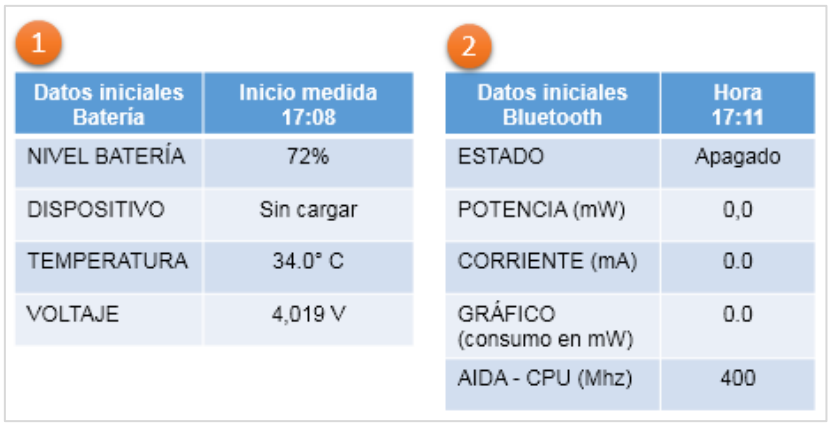
Figura 60. Valores iniciales batería y Bluetooth apagado

Para el Módulo Bluetooth se observa en la Figura 60 que partimos con valores de potencia y corriente de cero, debido a que no está activo. Estos valores varían conforme el componente Bluetooth cambie al estado de encendido.

En resumen, esta prueba inicia obteniendo el porcentaje de carga, la temperatura y el voltaje de la batería y con valores de cero para el consumo de potencia y corriente en el estado apagado del Bluetooth.

Al final de las pruebas, estos valores tienen un comportamiento diferente considerando los estados de activo, emparejado, transmitiendo y desactivado del componente Bluetooth.

La Figura 61, muestra los valores obtenidos por observación cuando el Bluetooth está apagado.



1

Datos iniciales Batería	Inicio medida 17:08
NIVEL BATERÍA	72%
DISPOSITIVO	Sin cargar
TEMPERATURA	34.0° C
VOLTAJE	4,019 V

2

Datos iniciales Bluetooth	Hora 17:11
ESTADO	Apagado
POTENCIA (mW)	0,0
CORRIENTE (mA)	0.0
GRÁFICO (consumo en mW)	0.0
AIDA - CPU (Mhz)	400

Figura 61. Datos iniciales – Bluetooth apagado

Bluetooth encendido

Como se puede apreciar en la Figura 62, el Bluetooth se encuentra encendido y la aplicación entrega las primeras mediciones de potencia 0,29 mW y corriente de 0,1 mA, para este estado. El ahorro de energía que establece el Bluetooth aplica cuando la conexión se realiza en un radio de corto alcance. De igual forma se aprecia en la gráfica de la aplicación de monitoreo, que el valor de potencia está por debajo de los 2mW.

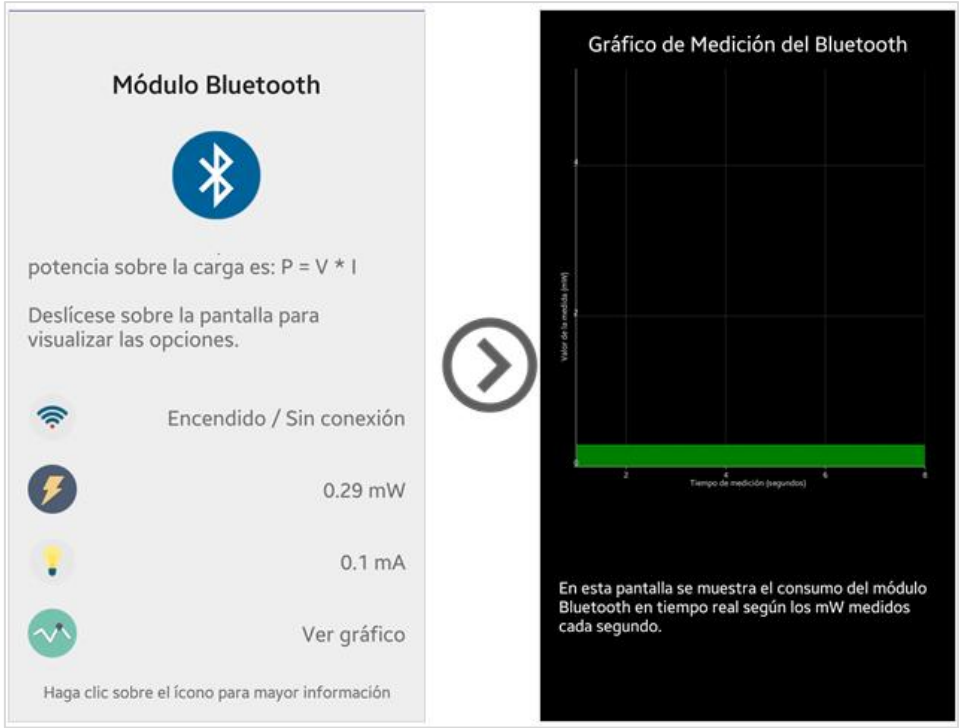


Figura 62. Bluetooth encendido, sin emparejar y gráfico de medición

Bluetooth emparejado y transfiriendo

Con el Bluetooth encendido, se busca los dispositivos habilitados para este componente. En este caso la prueba se complementa usando una Tablet para enviar un archivo de video cuyo tamaño es de 12.31 MB.

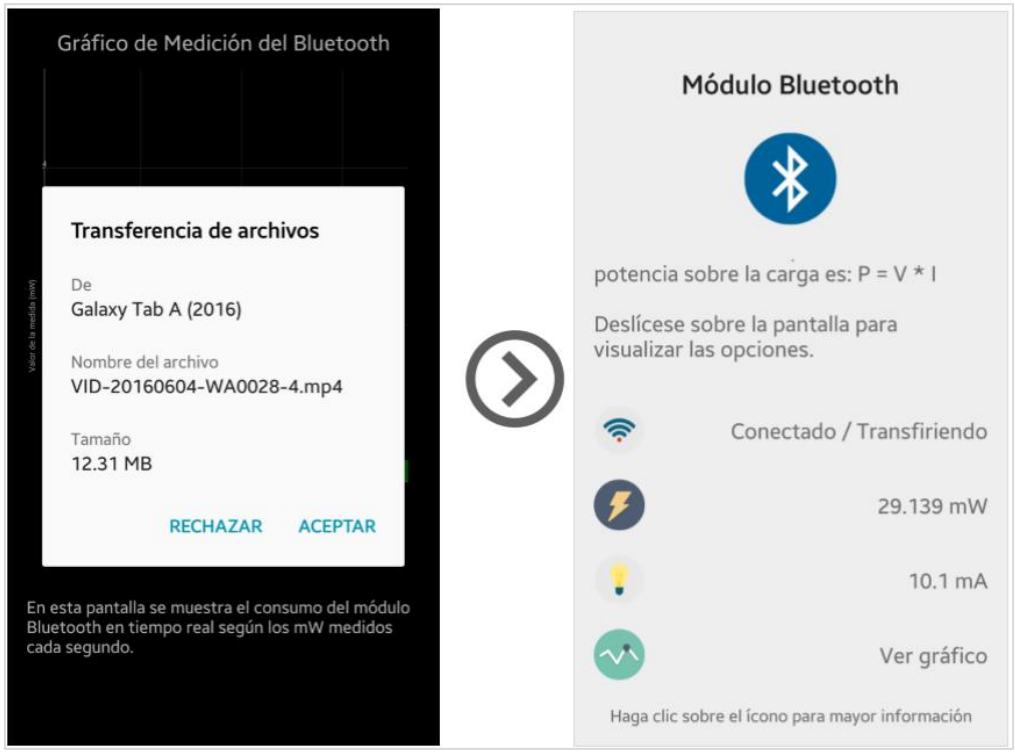


Figura 63. Dispositivo móvil emparejado y transmitiendo

Según la Figura 63, el componente está transfiriendo un archivo de 12,31 MB a un ancho de banda de 32 Mbits/s (Estándar del Bluetooth Versión 4.0) y como resultados se tiene una potencia de 29,139 mW y de 10,1 mA para la corriente.

En el caso de la potencia este valor de 29, 139 mW se encuentra dentro de los intervalos definidos en la Clases 1 y 2 del estándar Bluetooth, para potencias máximas de transmisión permitidas y alcances aproximados.

El resultado de la potencia de consumo para este componente se presenta en una gráfica en el tiempo (ver Figura 64). Al inicio se mantiene en el nivel mínimo de consumo de 0,1 mW, posterior va creciendo hasta llegar a su valor máximo de 29,13 mW y luego se mantiene constante durante el tiempo que

dure la transmisión de información, hasta que finalmente regresa a su posición inicial una vez que termina de transmitir el archivo.

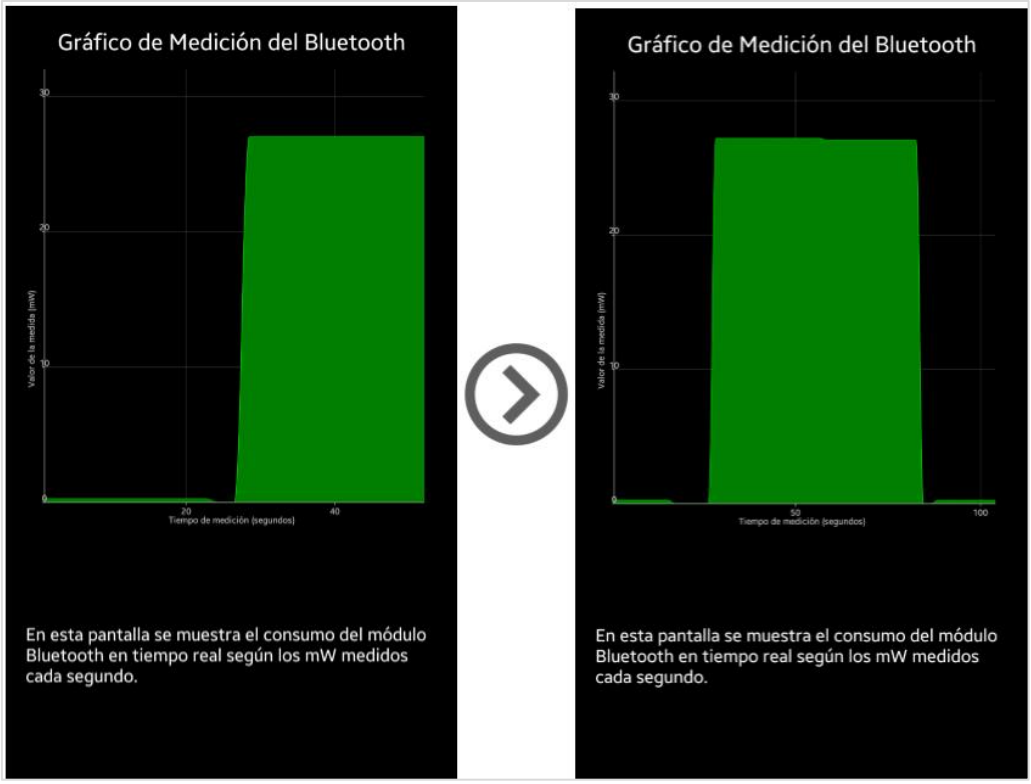


Figura 64. Grafico del consumo del Bluetooth

En definitiva la aplicación de consumo de energía ofrece el módulo que recopila los datos de potencia consumida de los componentes de Bluetooth y GPS durante la transferencia del archivo de video.

Figura 65. Resultados finales

Los resultados finales presentados en la Figura 65, versus los iniciales de la Figura 60 indica que la batería se descarga en un 6%.

Este porcentaje de descarga tiene relación directa con el consumo de potencia del CPU, de la Pantalla, de la misma aplicación monitoreo, además del tiempo que se emplea en realizar cada una de las pruebas bajos los diferentes estados.

3.1.4.2. GPS

Para las medidas del GPS, se plantea en el escenario de prueba la instalación de una aplicación deportiva (Runkeeper), que usa el GPS para recopilar datos de cualquier actividad que se realice.

En este caso se define una ruta de aproximadamente 1 Km para caminar y observar el consumo de potencia que tiene el GPS a lo largo de este trayecto.

Como primer paso se toma en cuenta las medidas iniciales del Monitor de Batería, con un nivel de carga del 80% tal como lo describe la Figura 66.

GPS Activo

Figura 66. Medidas iniciales batería – GPS activo

Con el GPS activo la potencia de consumo en ese momento es de 157,95 mW, y una corriente de 50 mA.

En cuanto a la funcionalidad del GPS con la aplicación deportiva Runkeeper, es necesario recordar que el GPS requiere del sistema de posicionamiento global, de mapas y del sistema de navegación. Para esto el GPS obtiene nuestra posición a través del receptor de dispositivo móvil, este a su vez debe captar la señal de al menos cuatro satélites para fijar una posición a partir del tiempo que demora la señal al llegar a cada satélite y por triangulación se establece nuestra posición, la misma que es dada en coordenadas de latitud y longitud.

Por otra parte la Figura 67, muestra una gráfica constante en el tiempo, es decir no tiene picos, no hay una variación considerable y esto se da por el tipo de conexión establecida, considerando que la señal es óptima, no tiene obstáculos en el trayecto que se establece para realizar la prueba.

Figura 67. Gráfico GPS – Valores iniciales APP

Respecto al procesamiento del CPU (ver Figura 68), se observa que tiene un valor de 1100 MHz dentro de los parámetros normales de 400 a 1500 MHz.

Este valor sin duda representa un alto procesamiento, comparado con el que utiliza el componente Bluetooth; esto se debe por el tipo de conexión que se da en distancias cortas.

También es un indicador de que el CPU está ejecutando un proceso continuo que para este caso es la conexión con el GPS.

De igual forma la Figura 68 muestra que la potencia no es constante, esto se debe a que durante del trayecto se presentan obstáculos y el nivel de la señal se degrada, sin embargo, el patrón de comportamiento vuelve a la normalidad cuando se tiene cobertura y no existen obstáculos.

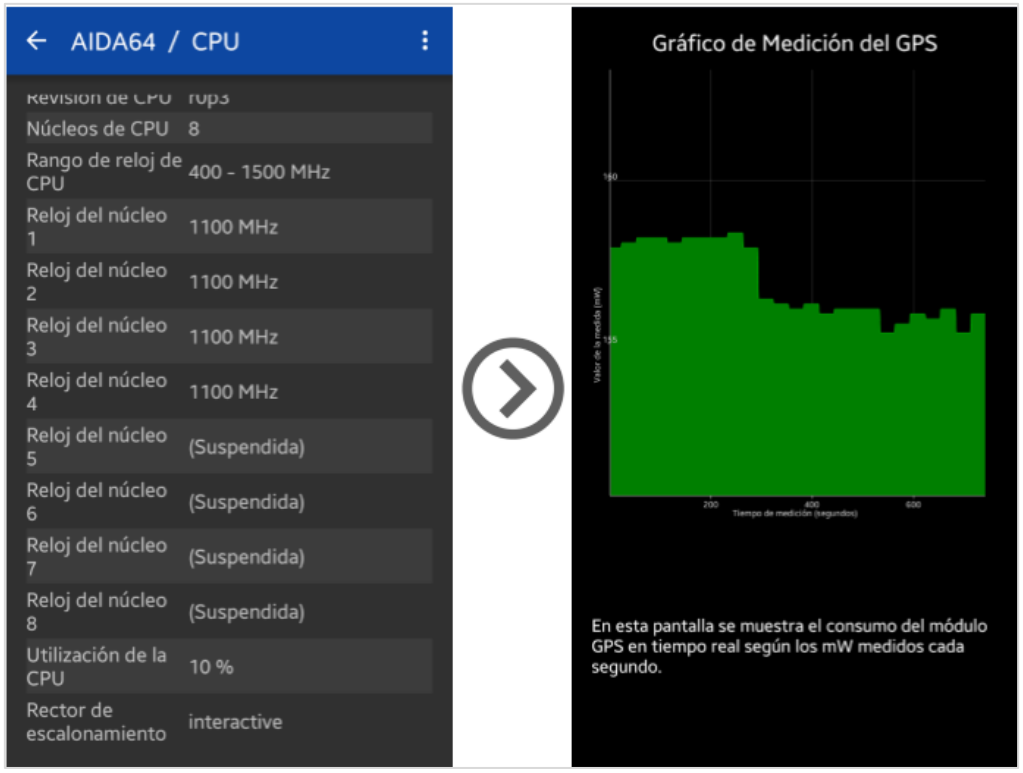


Figura 68. Velocidad CPU – Gráfico GPS

Luego de haber transcurrido 13 minutos, el módulo GPS arroja una potencia de 153,33 mW, es decir es menor a la medida inicial, considerando las particularidades que se encontraron en parte del trayecto.

De la misma manera la Figura 69 del Monitor de Batería indica que se tiene 76% de batería es decir 4% menos respecto a la medida inicial que fue de 80%.

Este valor de consumo de 4% en 13 minutos utilizando GPS y la aplicación deportiva, nos da una idea de que tan rápido se descarga la batería, a este ritmo la batería se descarga en un 20% en aproximadamente una hora.

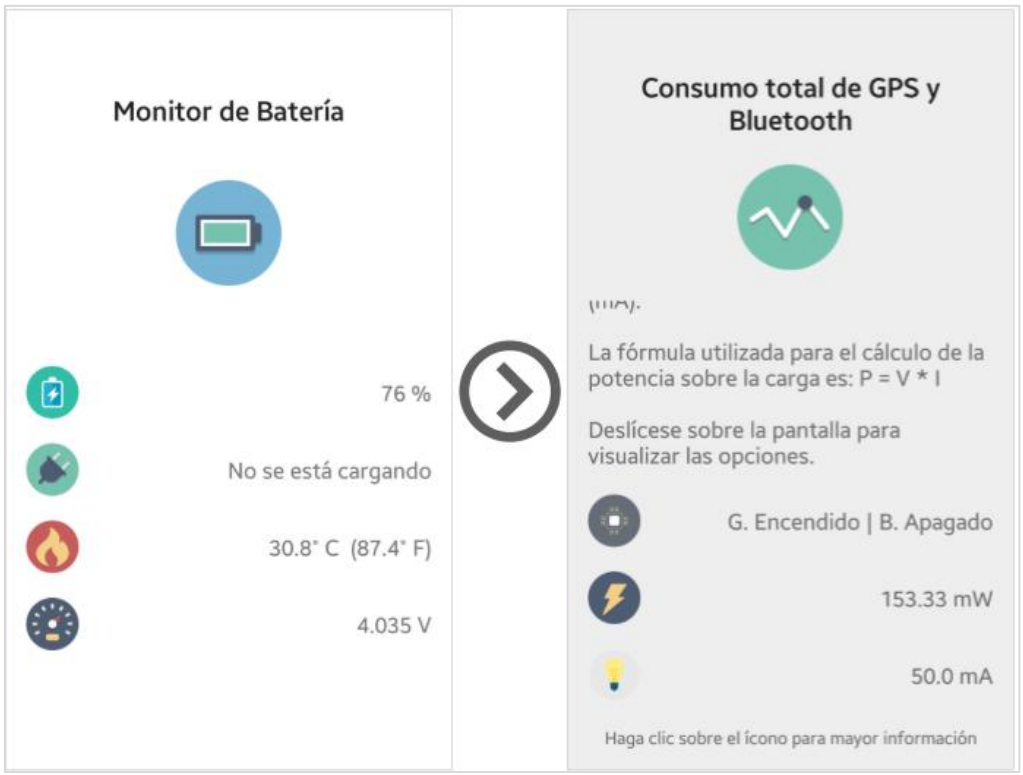


Figura 69. Valores batería – Consumo Bluetooth y GPS

Bluetooth transfiriendo y GPS activado

Continuando con las mediciones, en esta prueba se habilita los dos componentes y se vuelve a realizar los procesos de medición descritos en los apartados anteriores.

Según se puede apreciar en las Figuras 70 y 71, parte tomando las medidas iniciales que arroja la batería, seguidamente se activa ambos componentes, teniendo en primera instancia que el GPS arroja un valor de potencia de 151.31 mW y el Bluetooth 30,474 mW cuando está transfiriendo un archivo. Cabe señalar que se omitió el segundo valor del Bluetooth cuando se enciende por ser pequeño (0,1mW).

Se observa que estos valores son similares en ambos componentes, no hay un cambio sustancial, las variaciones son mínimas entre una y otra medida respecto a las primeras las pruebas individuales.

Figura 70. Datos iniciales batería – Consumo del Módulo

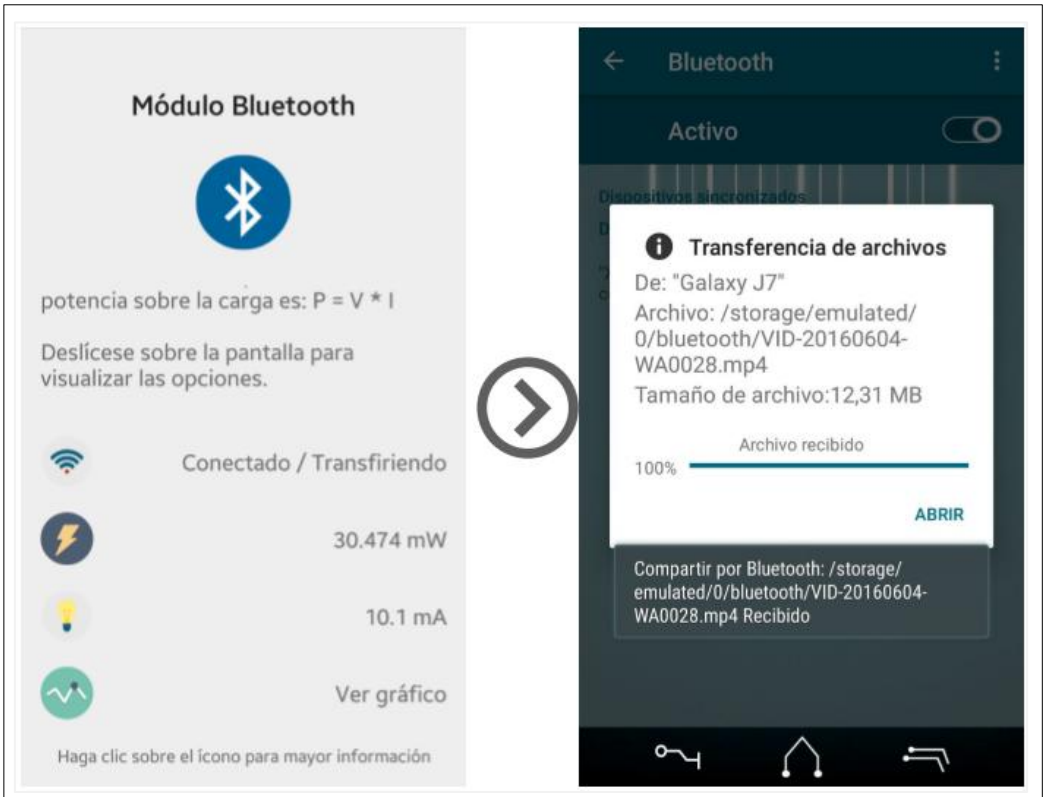


Figura 71. Bluetooth conectado y transfiriendo

Resultado de todo este proceso, la batería se descargó el 1 %, este valor al parecer es significativo comparado con el valor de 4% que se obtuvo cuando solo se midió GPS.

Según el análisis efectuado, el valor que entrega el Monitor de Batería tiene algunas particularidades. Por ejemplo el Bluetooth consume potencia solo cuando está transfiriendo y depende del tiempo que dure la misma.

En la anterior prueba se recorre una distancia de aproximadamente 1 Km, sin embargo, en esta otra prueba solo fueron 50 metros (ver Figura 72), considerando que no es necesario debido a que el archivo que se transfiere por Bluetooth dura aproximadamente unos 3 minutos, tiempo en el que este componente estaría consumiendo potencia a la par con el GPS.

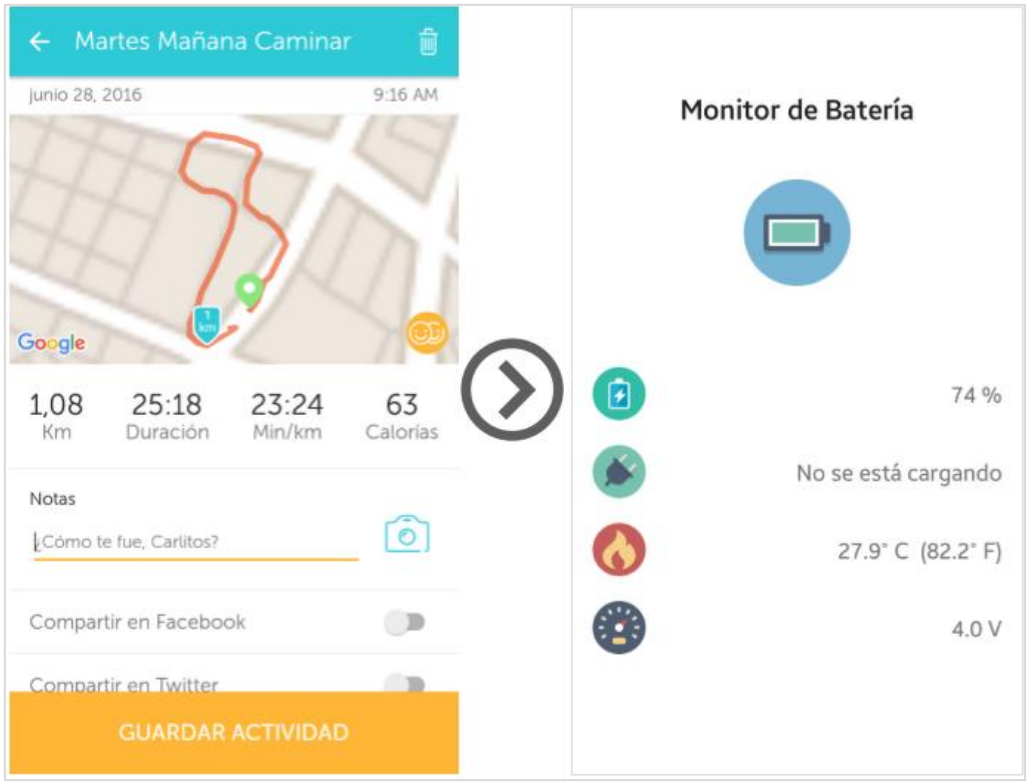


Figura 72. Porcentaje de descarga de la batería

3.1.4.3. Resumen de medidas Bluetooth y GPS

Resultados: Bluetooth

1

Datos iniciales Batería	Inicio medida 17:08
NIVEL BATERÍA	72%
DISPOSITIVO	Sin cargar
TEMPERATURA	34.0° C
VOLTAJE	4,019 V

2

Datos iniciales Bluetooth	Hora 17:11
ESTADO	Apagado
POTENCIA (mW)	0,0
CORRIENTE (mA)	0.0
GRÁFICO (consumo en mW)	0.0
AIDA - CPU (Mhz)	400

3

Datos iniciales Bluetooth	Hora 17:15
ESTADO	Encendido
POTENCIA (mW)	0,29
CORRIENTE (mA)	0.1
GRÁFICO (consumo en mW)	0.1
AIDA - CPU (Mhz)	400

4

Datos iniciales Bluetooth	Hora 17:16
ESTADO	Encendido y transfiriendo
POTENCIA (mW)	29,139
CORRIENTE (mA)	10.1
GRÁFICO (consumo en mW)	28
AIDA - CPU (Mhz)	400

5

Datos Consumo Bluetooth	Hora 17:19
ESTADO GPS	Apagado
ESTADO BLUETOOTH	Encendido
POTENCIA (mW)	0.289
CORRIENTE (mA)	0.1

6

Datos finales Batería	Fin medida 17:23
NIVEL BATERÍA	66%
DISPOSITIVO	Sin cargar
TEMPERATURA	33.0° C
VOLTAJE	3,933 V

Figura 73. Resultados Bluetooth

Resultados: GPS - Activo

1

Datos iniciales Batería	Inicio medida 09:09
NIVEL BATERÍA	80%
DISPOSITIVO	Sin cargar
TEMPERATURA	29.9° C
VOLTAJE	4,074 V

2

Datos iniciales GPS	Hora 09:10
ESTADO	Activado
POTENCIA (mW)	157.95001
CORRIENTE (mA)	50
GRÁFICO (consumo en mW)	157
AIDA - CPU (Mhz)	1100

3

Datos iniciales APP entrenar	Hora 09:16
GPS	Buena señal
Distancia (Km)	0,00
Ritmo actual (min/Km)	0:00
AIDA - CPU (Mhz)	1100
GRÁFICO (consumo en mW)	157

4

Datos finales GPS	Hora 13:44
ESTADO	Activado
POTENCIA (mW)	153.482
CORRIENTE (mA)	50
GRÁFICO (consumo en mW)	156
AIDA - CPU (Mhz)	1100

5

Datos finales Batería	Fin medida 13:54
NIVEL BATERÍA	76%
DISPOSITIVO	Sin cargar
TEMPERATURA	30.8° C
VOLTAJE	4,035 V

6

Datos Consumo GPS y Bluetooth	Hora 14:00
ESTADO GPS	Encendido
ESTADO BLUETOOTH	Apagado
POTENCIA (mW)	153.33
CORRIENTE (mA)	50

Figura 74. Resultados GPS

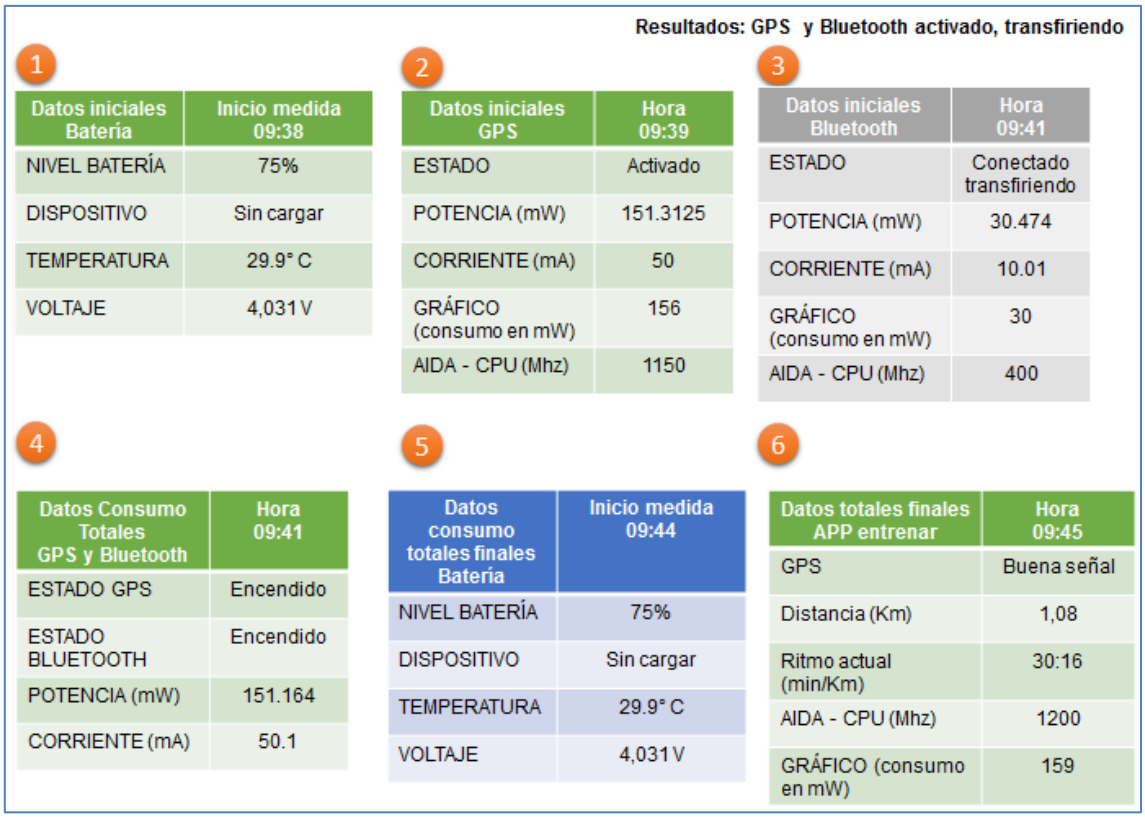


Figura 75. Resultados Bluetooth y GPS

Finalizado las pruebas de medición de los componentes de Bluetooth y GPS se concluye que el GPS consume mayor potencia cuando está conectado, GPS (ver Figuras 73, 74 y 75) tiene un valor de potencia máximo que es el umbral, siendo este cuando la batería este cargada al 100% con un valor de 200mW.

En cambio Bluetooth tiene un valor de potencia máximo que es el umbral, de 40mW en transferencia y solo en actividad de 0.4mW.

El porcentaje de descarga de la batería tiene relación directa con las potencias consumidas del CPU, Pantalla y los componentes que se estén ejecutando, que para este caso es el Bluetooth y GPS.

En el primer caso se tiene que la descarga es del 6% y en el segundo caso fue del 20%.

4. CAPÍTULO IV. MODELO CONSUMO ENERGÉTICO

Es de mucha importancia en la actualidad aportar con alternativas de eficiencia energética al momento de utilizar dispositivos móviles, considerando que existe una gran demanda de estos dispositivos para la realización de tareas que requieren movilidad.

Años atrás bastaba con cargar el dispositivo móvil por un par de horas y su autonomía era de 3 a 4 días, sin embargo hoy por hoy, los dispositivos móviles han evolucionado en sus diseños, tamaño, aplicaciones, hardware, software, sistemas operativos, requiriendo de una fuente de energía constante que garantice la funcionalidad del dispositivo móvil por más tiempo.

En ese sentido se ha incrementado el consumo de energía, por lo que la mayoría de usuarios suelen realizar de 2 a 3 cargas en el día. Desde luego existe un patrón de uso por los usuarios, que por lo general descarga la batería considerablemente, por ejemplo cuando tenemos varias aplicaciones como Facebook, Gmail, corriendo en segundo, de la misma forma cuando se tiene habilitado Wi-Fi y Datos Móviles a la vez.

Esto es innecesario porque el Wi-Fi se lo activa cuando no se dispone de un plan de datos, al tenerlo activado y considerando su forma de operar al buscar redes disponibles para conectarse, puede llegar a descargar el dispositivo móvil en un 8% a lo largo de una hora.

El modelo de consumo energético que se plantea en la Figura 76, está orientado a tratar las siguientes estrategias de eficiencia energética.

- Aplicaciones
- Actualización del sistema operativo
- Velocidad del dispositivo móvil
- Uso y carga de la batería

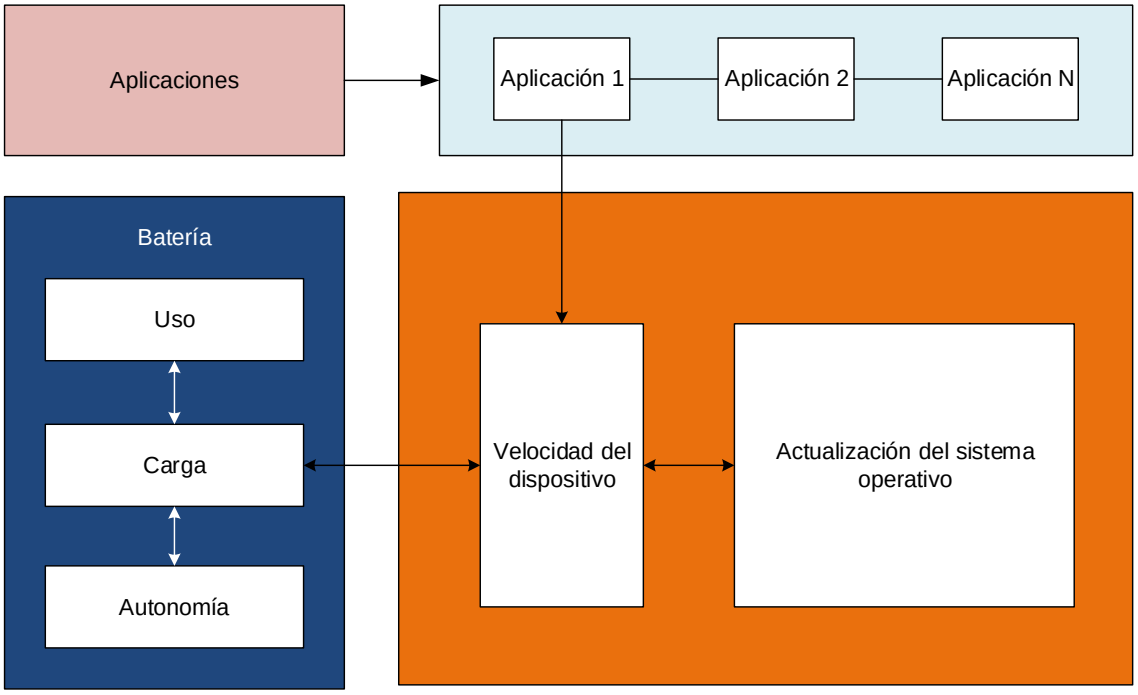


Figura 76. Modelo de consumo energético

4.1. Aplicaciones

Los procesos que corren en segundo plano requieren ser desactivados a la hora de ahorrar batería, por lo general los usuarios comunes desconocen que varias de las aplicaciones instaladas en el dispositivo móvil están consumiendo recursos como por ejemplo el CPU.

Este consumo se da porque las aplicaciones están constantemente sincronizando información con la red que tenga a su alcance. Un ejemplo de ello es la copia de seguridad de aplicaciones en cuentas de Google.

De igual manera los dispositivos móviles tiene cargadas una serie de aplicaciones que vienen de fábrica, la mayoría de veces son innecesarias, a pesar de que no se las use, consumen batería, por lo que es necesario deshabilitarlas o en su defecto desinstalarlas.

Se plantea crear una aplicación que analice el comportamiento de las APP instaladas en el dispositivo móvil, tomando como patrón de comportamiento el

consumo de memoria RAM, el uso del CPU y las que se encuentren corriendo en segundo plano. Una vez identificadas las APP, el aplicativo mediante un proceso de barrido de segundo plano cerrará las aplicaciones y mejorará el rendimiento del dispositivo móvil.

4.2. Actualización del sistema operativo

Considerando que día a día los desarrolladores de sistemas operativos están corrigiendo errores y realizando mejoras, es importante siempre tener actualizado el sistema operativo.

Las últimas versiones están encaminadas a optimizar el rendimiento de los dispositivos móviles. Esta actualización va encaminada a la calibración de la batería tomando como referencia errores pasados.

Se plantea el desarrollo de una APP, de actualización de firmware de prueba para el sistema operativo tomando en cuenta el patrón de comportamiento de la memoria interna del dispositivo móvil, la memoria RAM y el CPU.

4.3. Velocidad del dispositivo móvil

Los dispositivos móviles pueden optimizar varias de las tareas, entre ellas la velocidad de procesamiento a la hora de ahorrar energía. Todo CPU viene de fábrica con una capacidad predeterminada de velocidad de procesamiento, esto incide en el trabajo del CPU por debajo de su capacidad que realmente tiene.

Se plantea el desarrollo de una aplicación que permita setear la velocidad del CPU bajo escenarios de máximos y mínimos para el manejo de frecuencias y creación de perfiles respecto a su velocidad de procesamiento. Un ejemplo de perfil a tomar en cuenta es la carga del dispositivo móvil cuando se realizar por corriente o por USB.

4.4. Uso y carga de la batería

Respecto a la carga de la batería es importante realizar un ciclo para la descarga y otro ciclo para la carga, esto incide en el rendimiento de la batería, entendiéndose que este proceso le sirve a la batería para calibrase en estos ciclos. Cabe señalar que este proceso se lo debe realizar en un intervalo de uno a dos meses, eso sí, no realizarlo antes debido a que la batería está aprendiendo este patrón de carga.

Se plantea la creación de una APP para el registro de las cargas que se realice a la batería. Esta información permite conocer la capacidad del dispositivo móvil, el amperaje requerido y cuantas cargas soportaría la batería durante su vida útil.

5. CAPÍTULO V. CONCLUSIONES Y RECOMENDACIONES

5.1. CONCLUSIONES

El desarrollo del aplicativo define ocho módulos que interactúan entre sí para obtener las medidas de potencia, corriente, porcentaje de descarga y temperatura de los componentes de Bluetooth y GPS. Una ventaja que tiene el aplicativo es presentar el patrón de comportamiento de consumo de potencia mediante una gráfica en tiempo real, se puede instalar en cualquier dispositivo que tenga el sistema operativo Android, además el consumo de recursos es mínimo respecto a memoria RAM y CPU.

Una vez finalizado el proceso de medición de potencia de los componentes de Bluetooth y GPS, se tiene como resultados preliminares que la descarga de la batería en estado suspendido se realiza en aproximadamente 82 horas, mientras que el CPU en 46 horas y finalmente la pantalla en 10 horas.

El Bluetooth en el estado emparejado y transfiriendo obtuvo una potencia máxima de 29,139 mW y de 10,1 mA para la corriente, provocando que la batería se descargue en un 6%.

El GPS en estado activo obtuvo una potencia de consumo de 157,95 mW, y una corriente de 50 mA, como consecuencia de esta prueba la batería se descargó en un 20%. En conclusión el componente que descarga la batería considerablemente es el GPS. Cabe señalar que la potencia del CPU y de la pantalla, también tienen incidencia en el porcentaje de descarga de la batería.

Como parte de las estrategias de eficiencia energética se propone un modelo de consumo energético orientado a desarrollar aplicativos para dispositivos móviles tomando como base aspectos principales relacionados con las aplicaciones instaladas, la actualización del sistema operativo, la velocidad de procesamiento y al uso y carga de la batería. Esta propuesta permite brindar una autonomía a la batería y garantiza la funcionalidad del dispositivo móvil por más tiempo.

5.2. RECOMENDACIONES

Siendo el consumo de energía uno de los aspectos que preocupa a la hora de proteger el medio ambiente y considerando que somos responsables de cuidarlo mediante una administración eficaz y reducción de energía, recomiendo se complete este trabajo a través del desarrollo y la implementación de una aplicación de optimización de energía para dispositivos móviles, basado en la metodología y en el modelo de consumo energético propuesto, para otros componentes a ser evaluados como audio o video.

REFERENCIAS

Ana, H. y Gader,I. (2011). Desarrollo de Aplicaciones para dispositivos Móviles sobre la plataforma Android de Google. Recuperado el 11 de abril de 2016 de http://www.dit.ing.unp.edu.ar/graduate/bitstream/123456789/206/1/Informe.pdf

Acosta, M. (2006). Estudio del éstandar IEEE 802.15. 4 ZIGBEE para comunicaciones inalámbricas de área personal de bajo consumo de energía y su comparación en el estándar IEEE 802.15. 1 BLUETOOTH. Recuperado el 11 de abril de 2016 de http://bibdigital.epn.edu.ec/handle/15000/55

Becvar, Z., Mach, P., y Pravda, I. (2013). Redes móviles. Recuperado el 11 de Abril de 2016 de http://improvet.cvut.cz/project/download/C4ES/Redes_moviles.pdf

Cisco. (2016). Global Mobile Data Traffic Forecast Update 2015-2020. Recuperado el 08 de Abril de 2016 de http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html

López, E. (2012). Estudio del consumo de energía en un dispositivo Android. Recuperado el 10 de marzo de 2016 de http://e-archivo.uc3m.es/handle/10016/16713

Márquéz, A. (2015). Consumo energético en widgets de navegación para dispositivos móviles con pantallas táctiles. Recuperado el 16 de Abril de 2016 de https://www.cs.cinvestav.mx/TesisGraduados/2015/TesisAnaMarquez.pdf

Martinez, F. (2011). Aplicaciones para dispositivos móviles. Recuperado el 18 de Abril de 2016 de

https://riunet.upv.es/bitstream/handle/10251/11538/Memoria.pdf?sequence=1

Metri, G., Shi, W., y Monica, B. (2015). EnergyEfficiency Comparison of Mobile Platforms and Applications A Quantitative Approach. Recuperado el 16 de Abril de 2016 de http://www.cs.wayne.edu/~weisong/papers/metri15-comparison.pdf

Motlhabi, M. (2008). Advanced Android power management and implementation of wakelocks.. Recuperado el 14 de Abril de 2016, de http://www. cs. uwc. ac. za/~ mmotlhabi/apm2. pdf (nd).

Pastor, J. (2014). Xataka. Recuperado el 14 de Abril de 2016, de http://www.xataka.com/moviles/que-es-android-que-es-aosp-que-es-libre-y-abierto-y-que-no

Ramirez, G. (2013). Seguridad en Aplicaciones Móviles. Recuperado el 13 de Abril de 2016, de http://datateca.unad.edu.co/contenidos/233016/EXE_SAM/leccin_5_dispositivos_mviles.html

Revelo, J. (2014). Hermosa Programación. Recuperado el 27 de Mayo de 2016, de http://www.hermosaprogramacion.com/2014/08/aprendiendo-la-arquitectura-de-android/

Rica, A. (2013). Diseño y desarrollo de una red BAN Bluetooth para la monitorización en enfermos de Parkinsón. Recuperado el 28 de Mayo de 2016 de http://upcommons.upc.edu/handle/2099.1/22758

Sei, T. (2013). Android Jefe. Recuperado el 20 de Abril de 2016, de http://www.androidjefe.com/wakelock-detector-descubrir-aplicaciones-android-mas-bateria-usan/

Technologies. (2016). Market Share Statistics for Internet. Recuperado el 16 de Abril de 2016, de http://www.netmarketshare.com/

UNAD. (s.f.). Seguridad en aplicaciones móviles. Recuperado el 14 de Abril de 2016, de http://datateca.unad.edu.co/contenidos/233016/EXE_SAM/leccin_1_sistemas_operativos_moviles.html

Universidad Carlos III de Madrid. (s.f.). Programación en dispositivos móviles portables. Recuperado el 17 de Abril de 2016, de https://sites.google.com/site/swcuc3m/home/android/generalidades/2-2-arquitectura-de-android

Uriol, P. (2011). Migración a Android de una aplicación de visión artificial para el ámbito educattivo. Recuperado el 25 de Mayo de 2016 de https://e-archivo.uc3m.es/handle/10016/12740

Varela, M. (2007). Conceptos fundamentales de un Middleware y razones de su importancia en el mundo de hoy. Recuperado el 25 de mayo de 2016 de http://docplayer.es/2787001-Conceptos-fundamentales-de-un-middleware-y-razones-de-su-importancia-en-el-mundo-de-hoy.html

VNI, C. (2014). Cisco Visual Networking Index. Recuperado el 27 de Mayo de 2016, de http://www.cisco.com/assets/sol/sp/vni/forecast_highlights_mobile/index.html

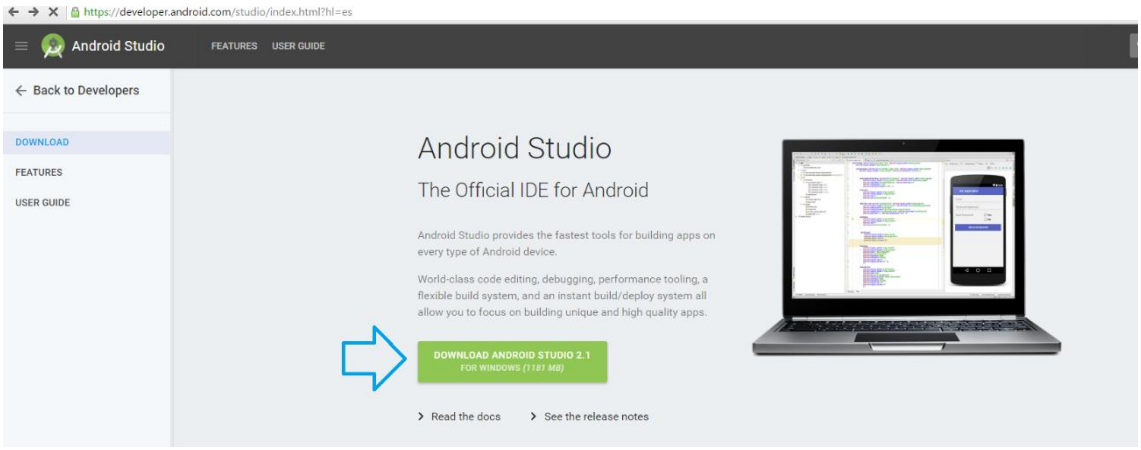
ANEXOS

ANEXO 1 - INSTALACIÓN ANDROID STUDIO

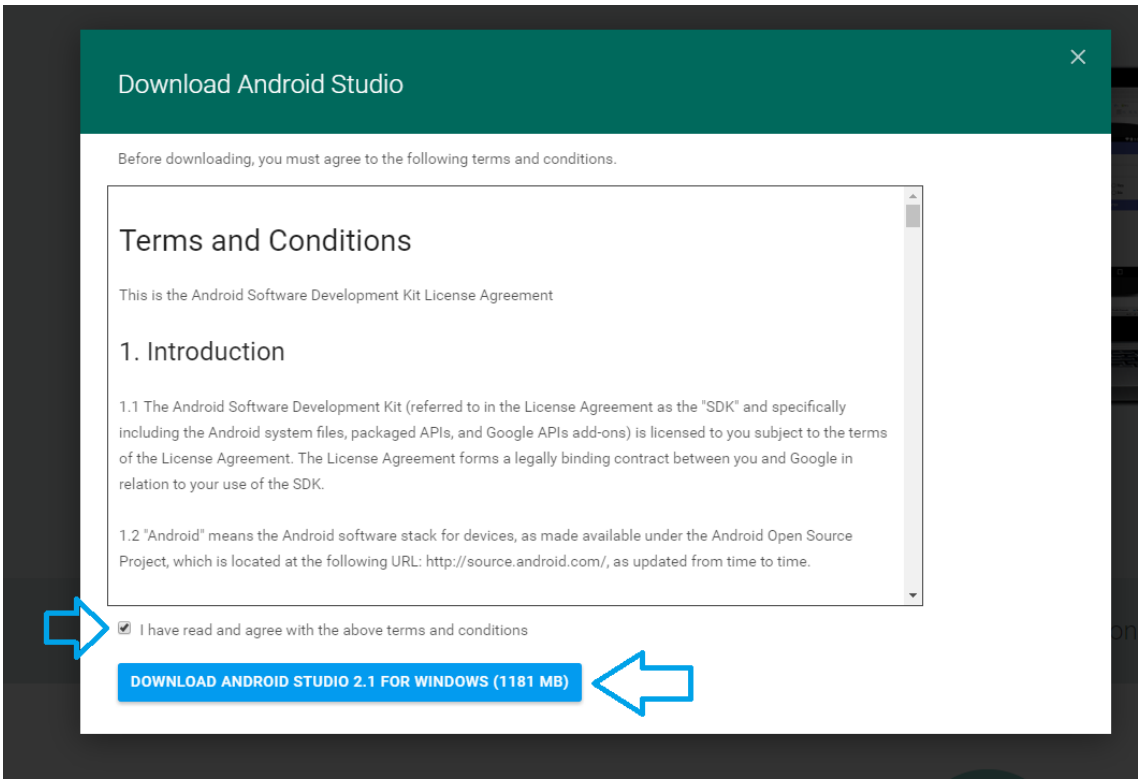
Descargar Android de la página:

https://developer.android.com/studio/index.html?hl=es

Hacer clic en la descarga

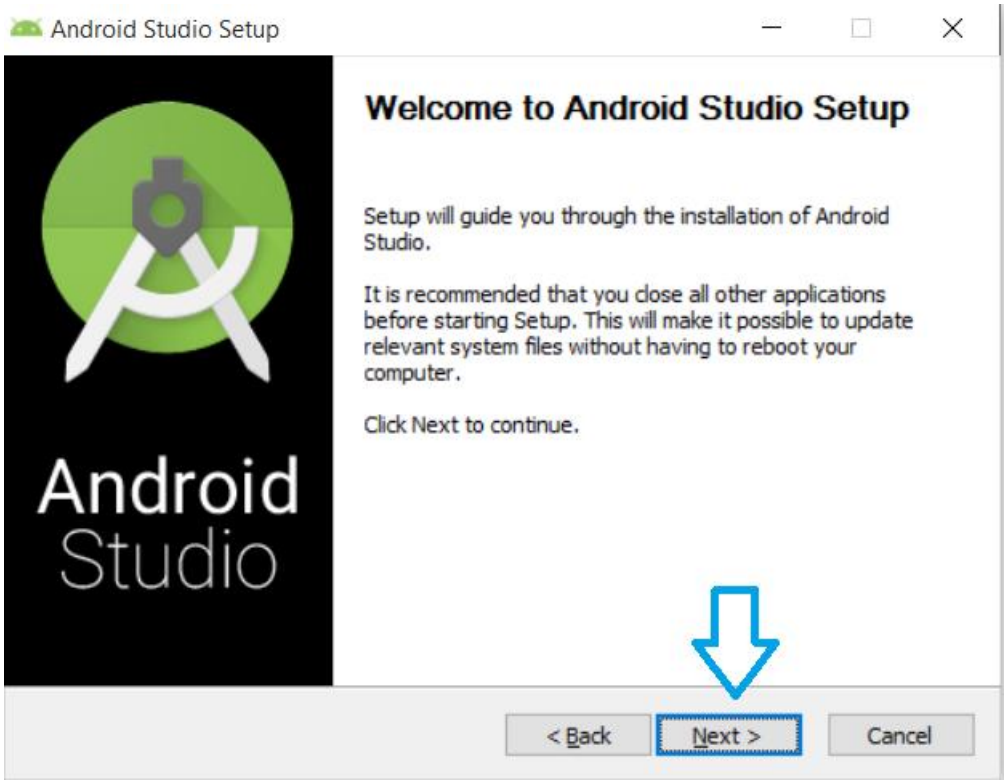


Después de leer los términos y condiciones aceptar, hacer clic en la descarga

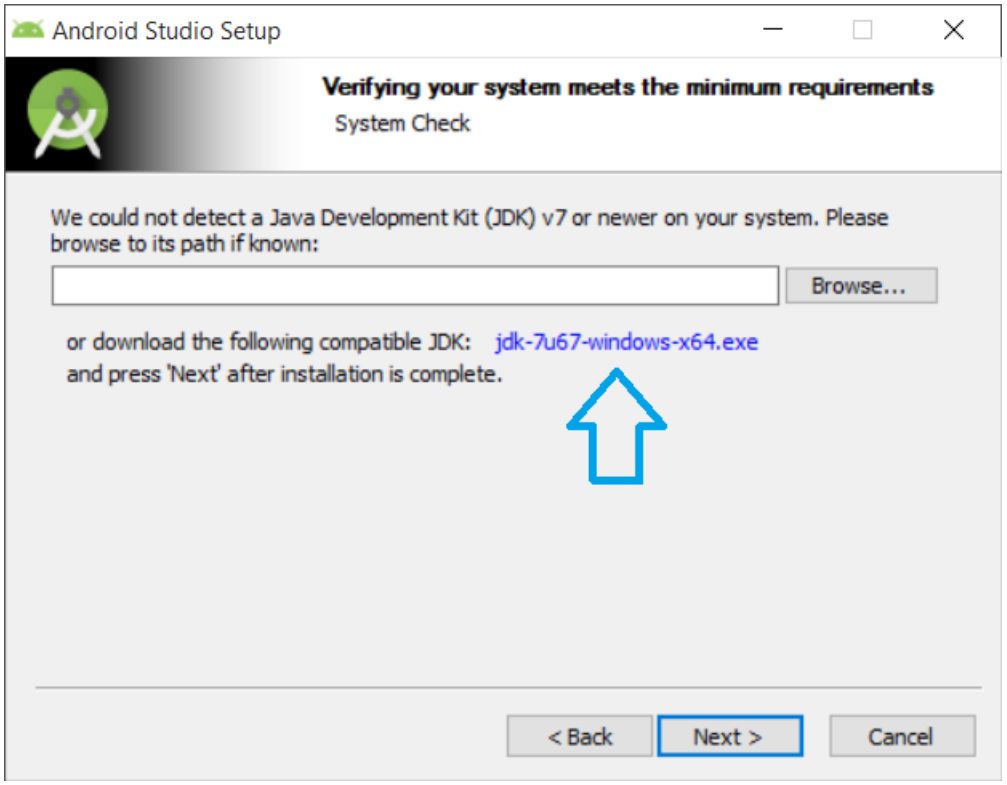


Ejecutar el programa descargado

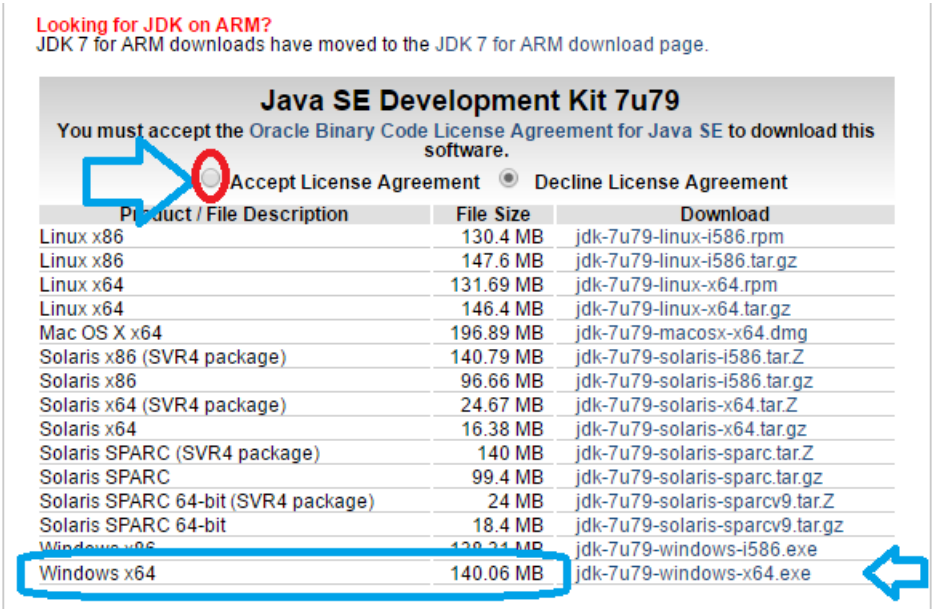
Dar clic a siguiente



Hacer clic en descargar JDK



De la página mostrada a continuación, primero aceptar la licencia y a continuación según el sistema operativo ya sea de x86 o x64, descargar su JDK



Looking for JDK on ARM?
JDK 7 for ARM downloads have moved to the JDK 7 for ARM download page.

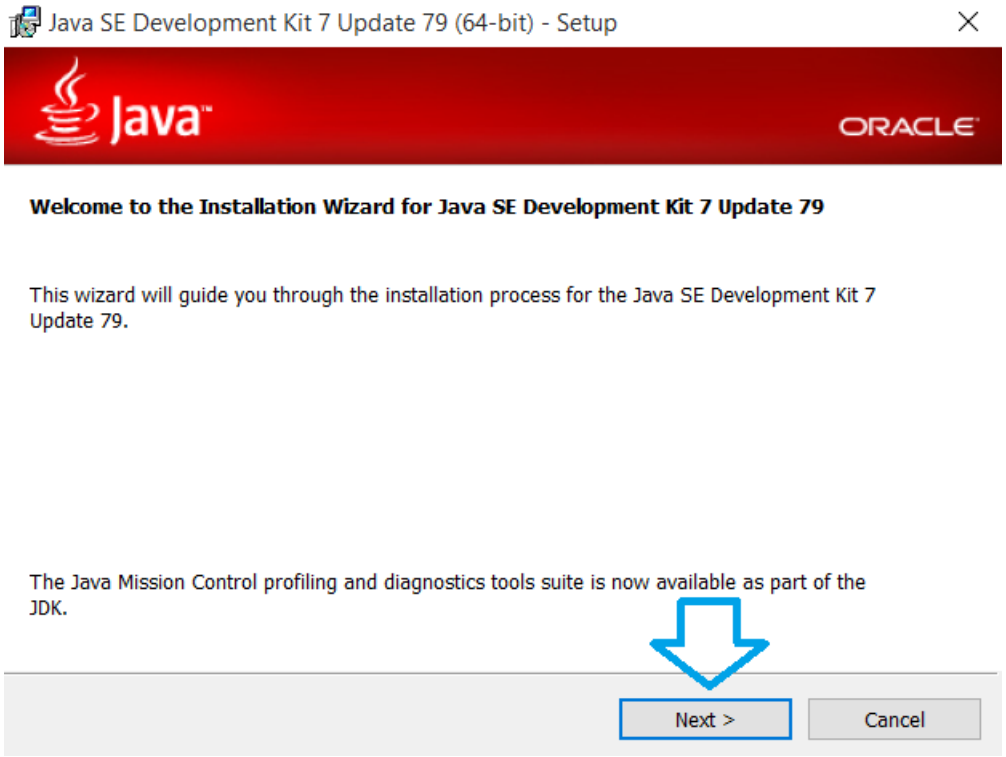
Java SE Development Kit 7u79

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

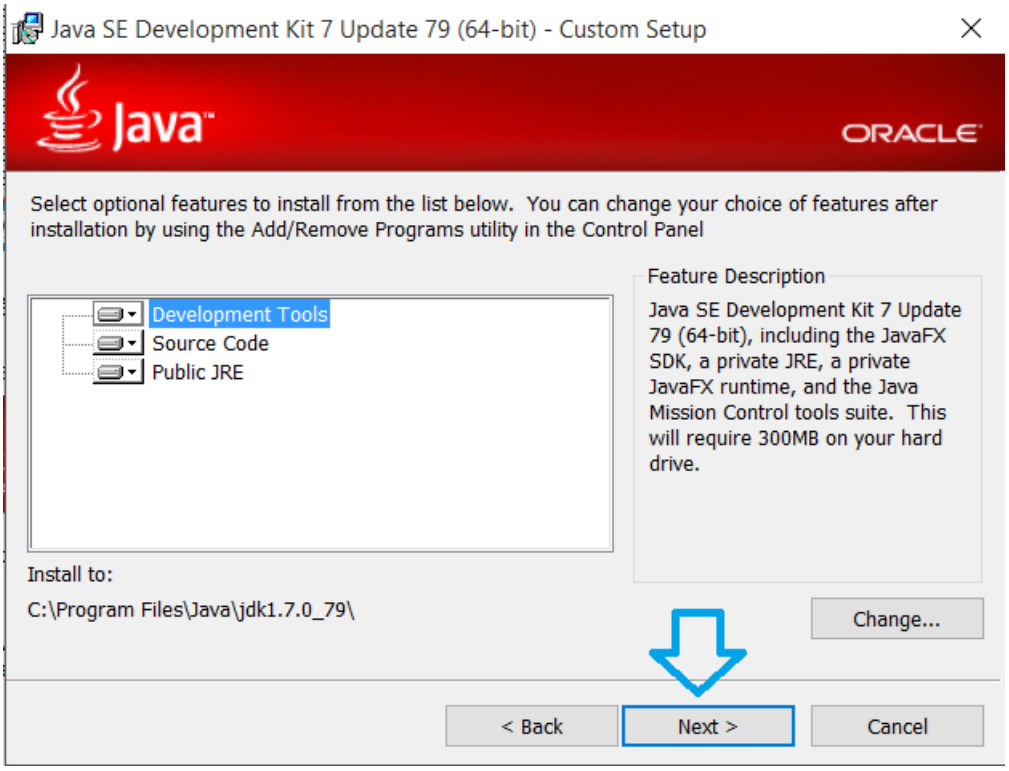
Accept License Agreement Decline License Agreement

Product / File Description	File Size	Download
Linux x86	130.4 MB	jdk-7u79-linux-i586.rpm
Linux x86	147.6 MB	jdk-7u79-linux-i586.tar.gz
Linux x64	131.69 MB	jdk-7u79-linux-x64.rpm
Linux x64	146.4 MB	jdk-7u79-linux-x64.tar.gz
Mac OS X x64	196.89 MB	jdk-7u79-macosx-x64.dmg
Solaris x86 (SVR4 package)	140.79 MB	jdk-7u79-solaris-i586.tar.Z
Solaris x86	96.66 MB	jdk-7u79-solaris-i586.tar.gz
Solaris x64 (SVR4 package)	24.67 MB	jdk-7u79-solaris-x64.tar.Z
Solaris x64	16.38 MB	jdk-7u79-solaris-x64.tar.gz
Solaris SPARC (SVR4 package)	140 MB	jdk-7u79-solaris-sparc.tar.Z
Solaris SPARC	99.4 MB	jdk-7u79-solaris-sparc.tar.gz
Solaris SPARC 64-bit (SVR4 package)	24 MB	jdk-7u79-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	18.4 MB	jdk-7u79-solaris-sparcv9.tar.gz
Windows x86	129.21 MB	jdk-7u79-windows-i586.exe
Windows x64	140.06 MB	jdk-7u79-windows-x64.exe

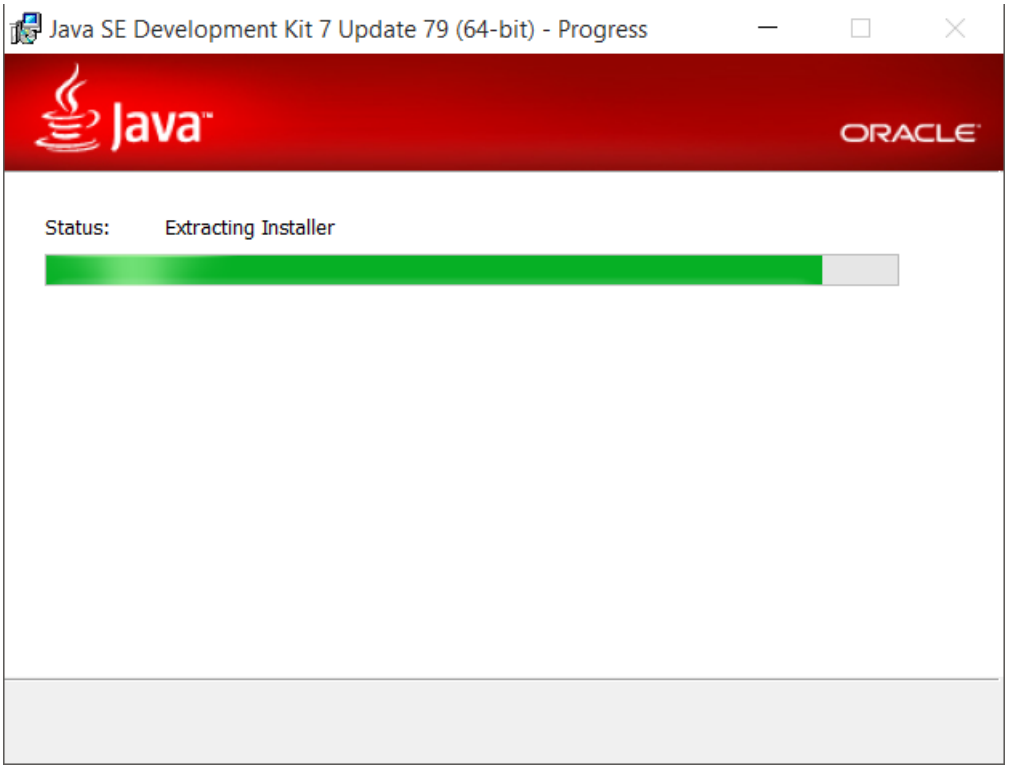
Después de la descarga, abrir el programa descargado y hacer clic en siguiente



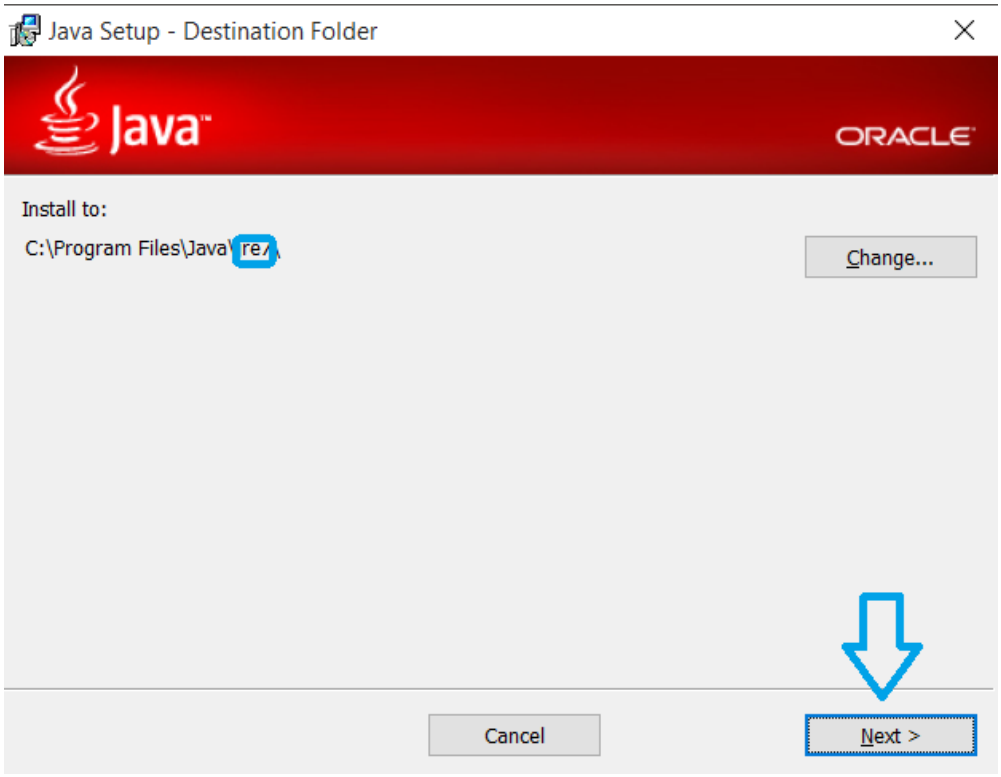
Hacer clic en siguiente



Esperar la carga

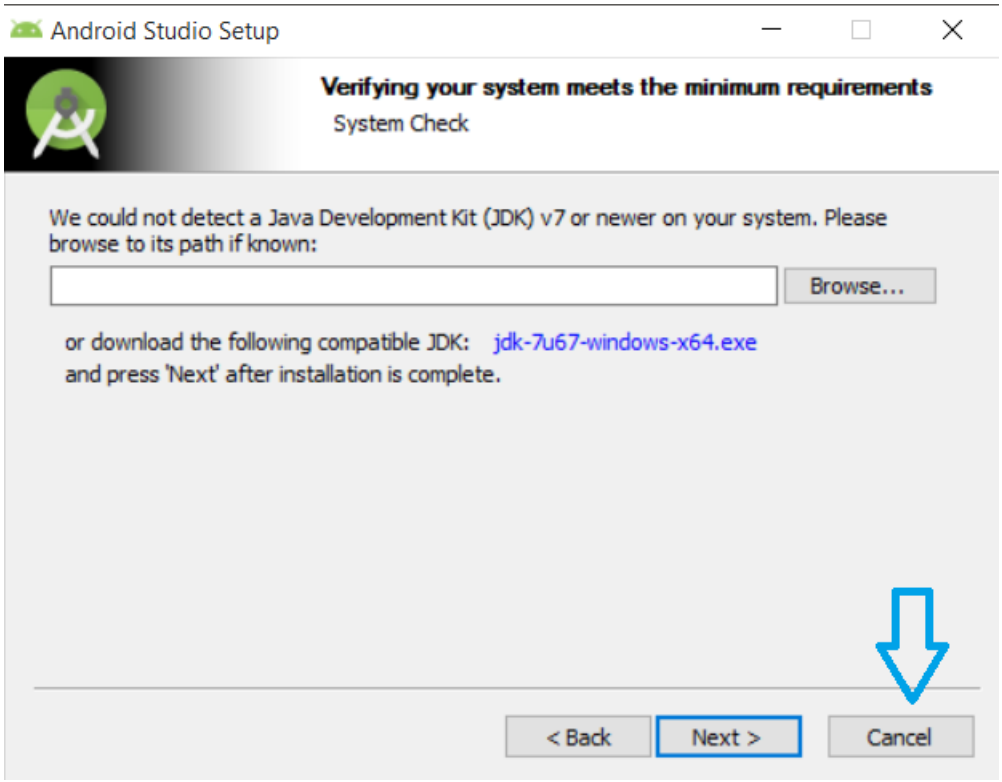


Hacer clic en siguiente

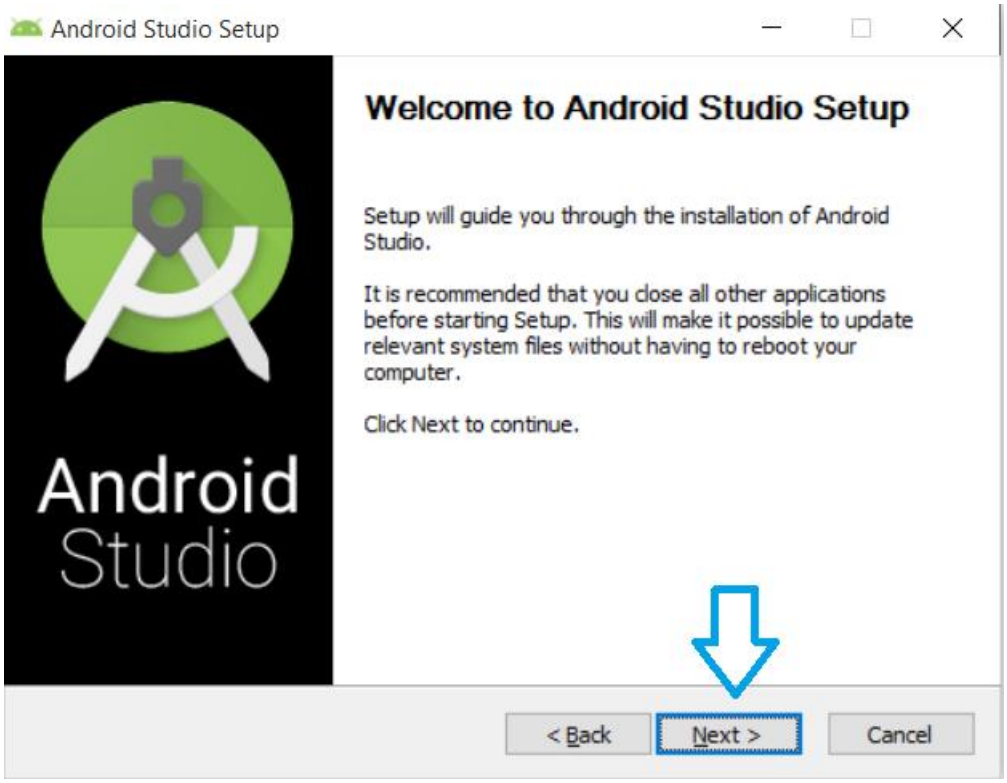


Después de la carga, finalizar cerrando la instalación

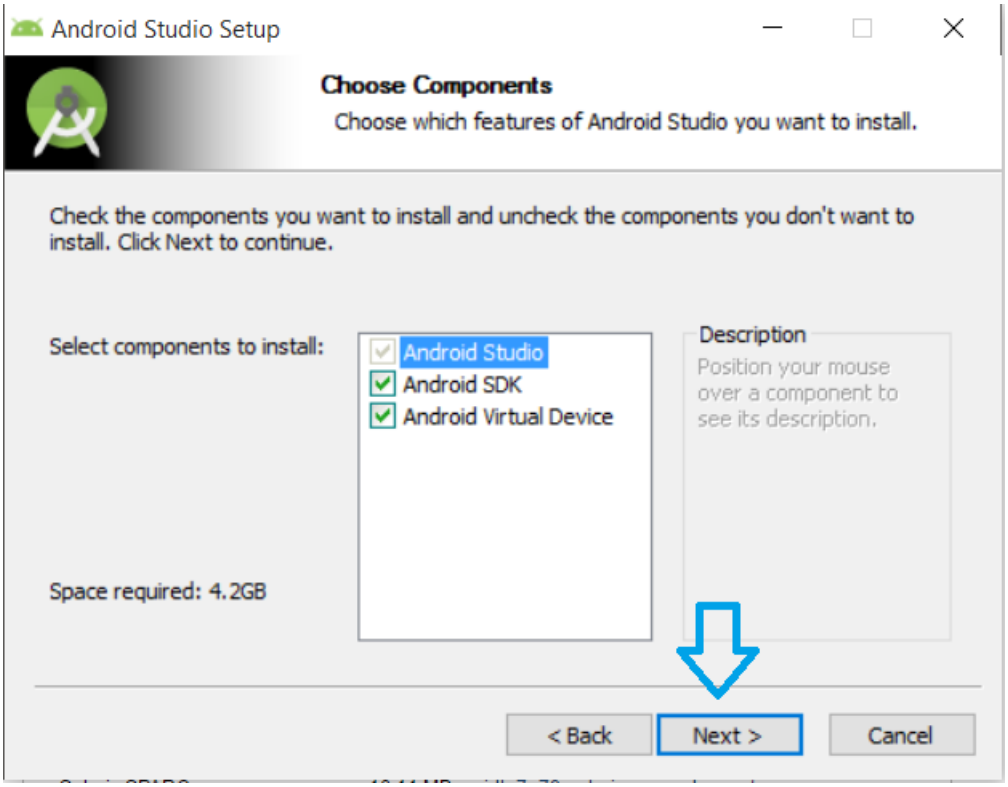
Volver a la instalación de Android, salir y volver a instalar



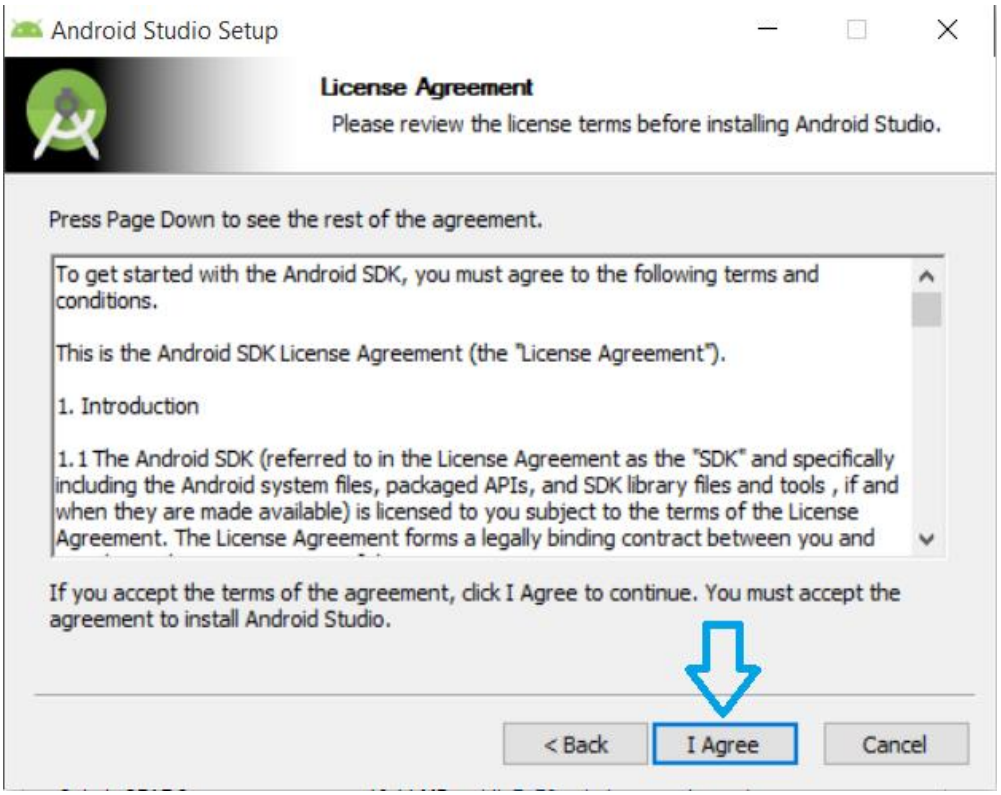
Hacer clic en siguiente



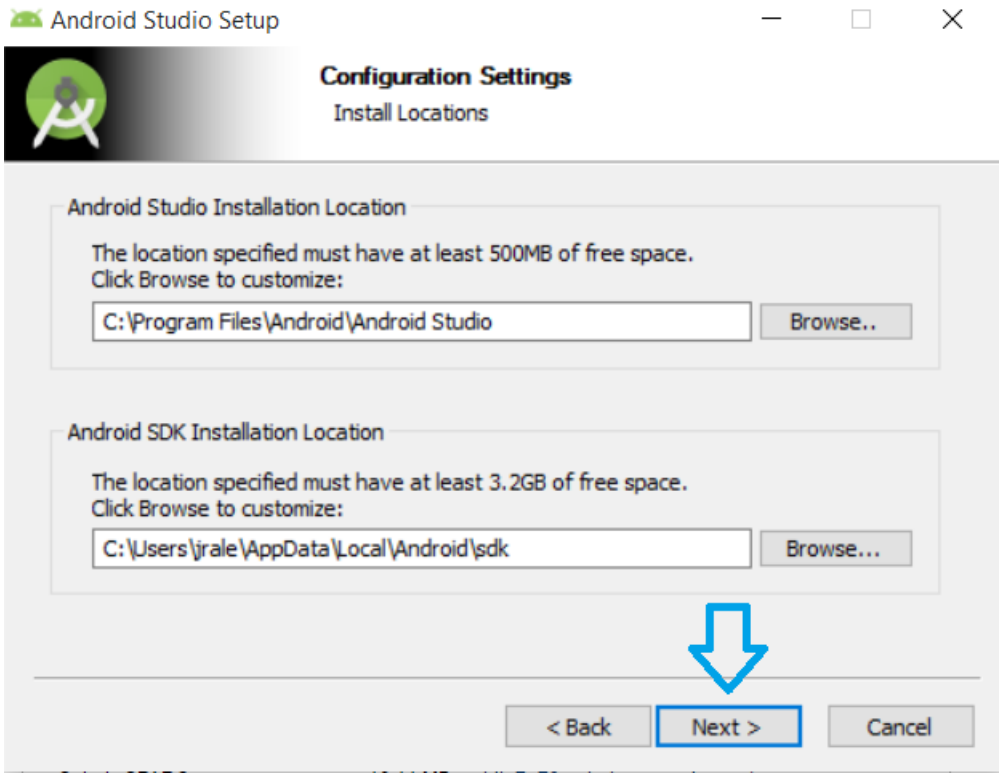
Hacer clic en siguiente



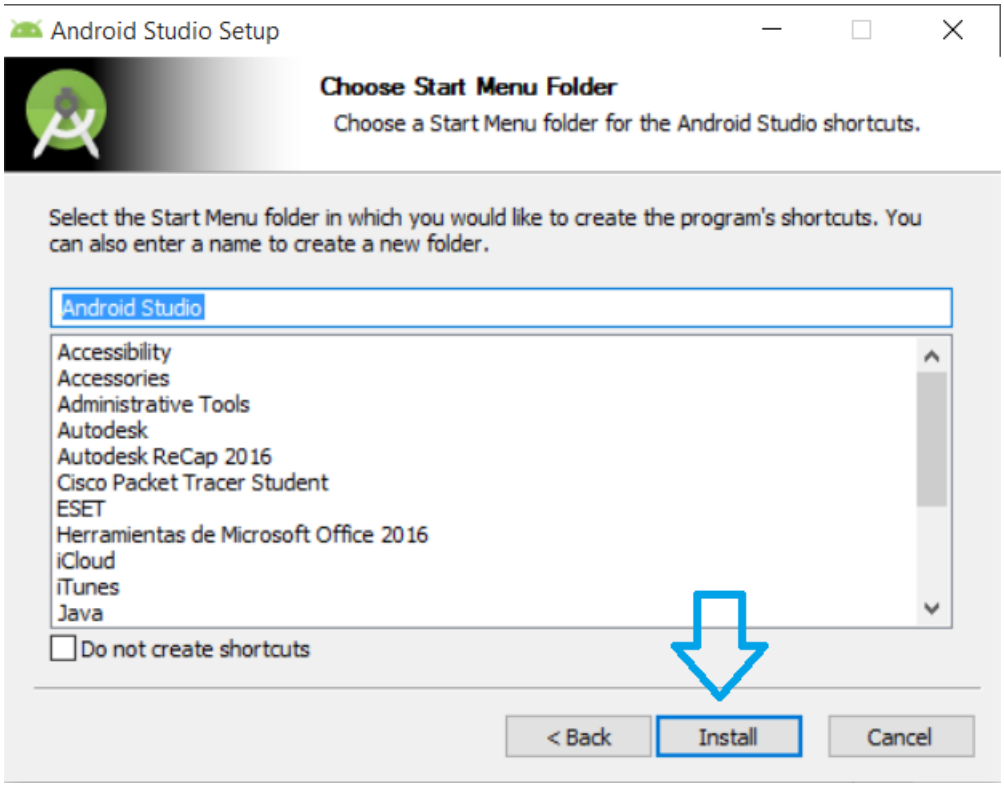
Acordar con la instalación



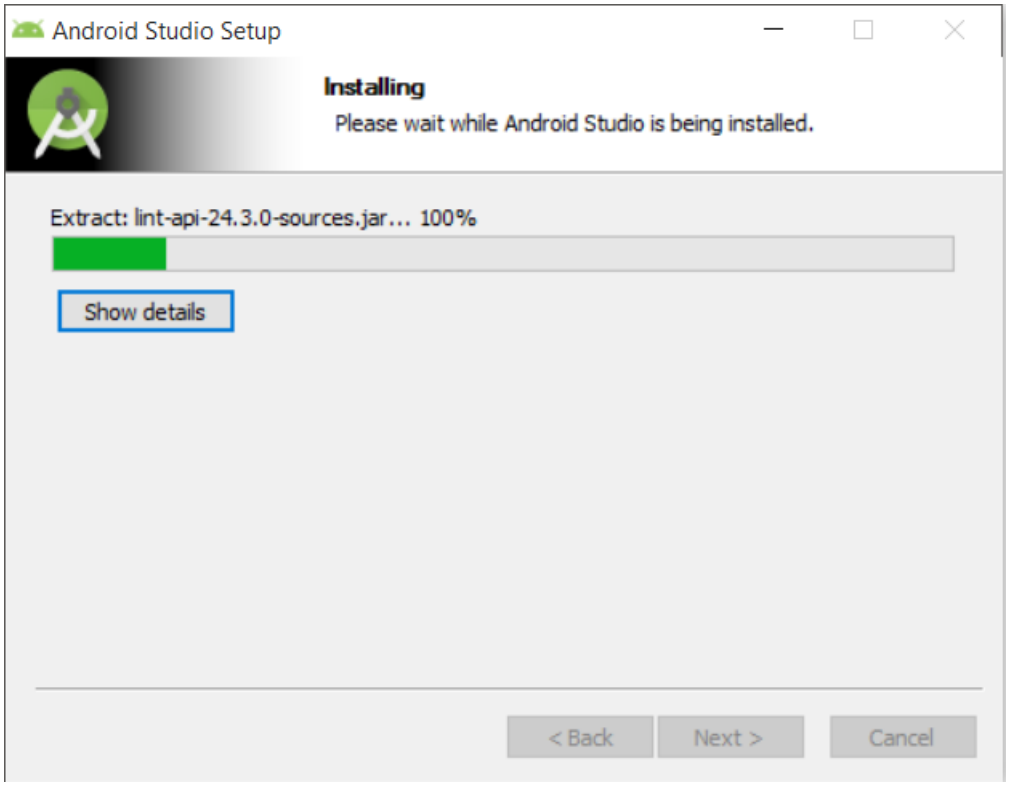
Elegir las rutas y hacer clic en siguiente



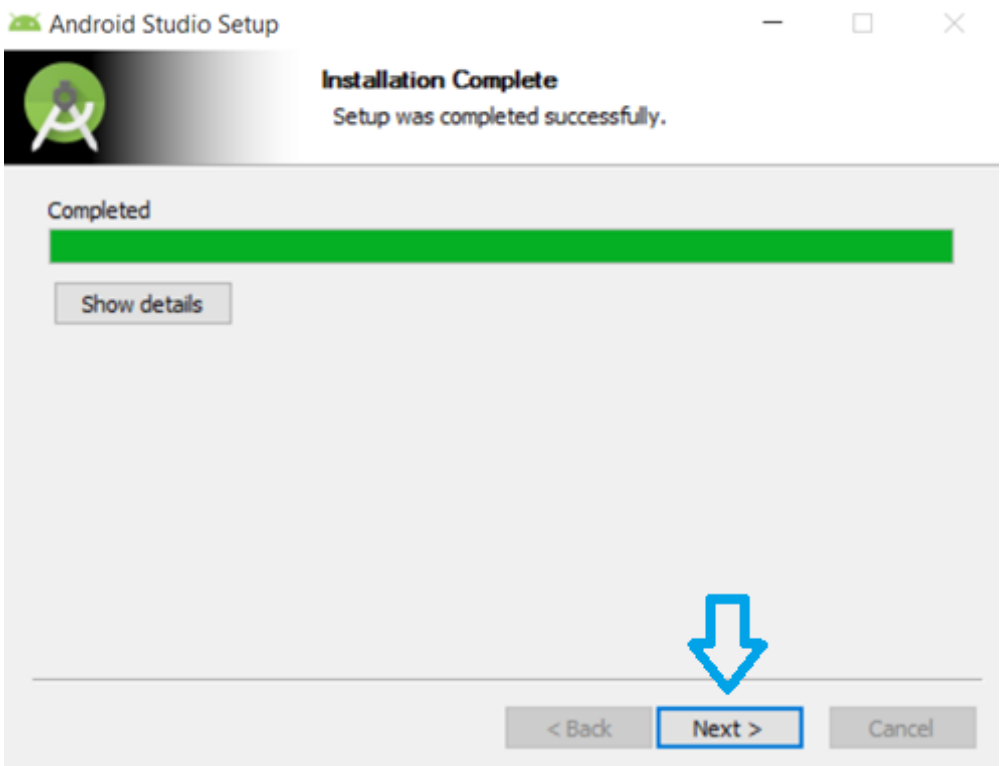
Hacer clic en instalar con el nombre deseado

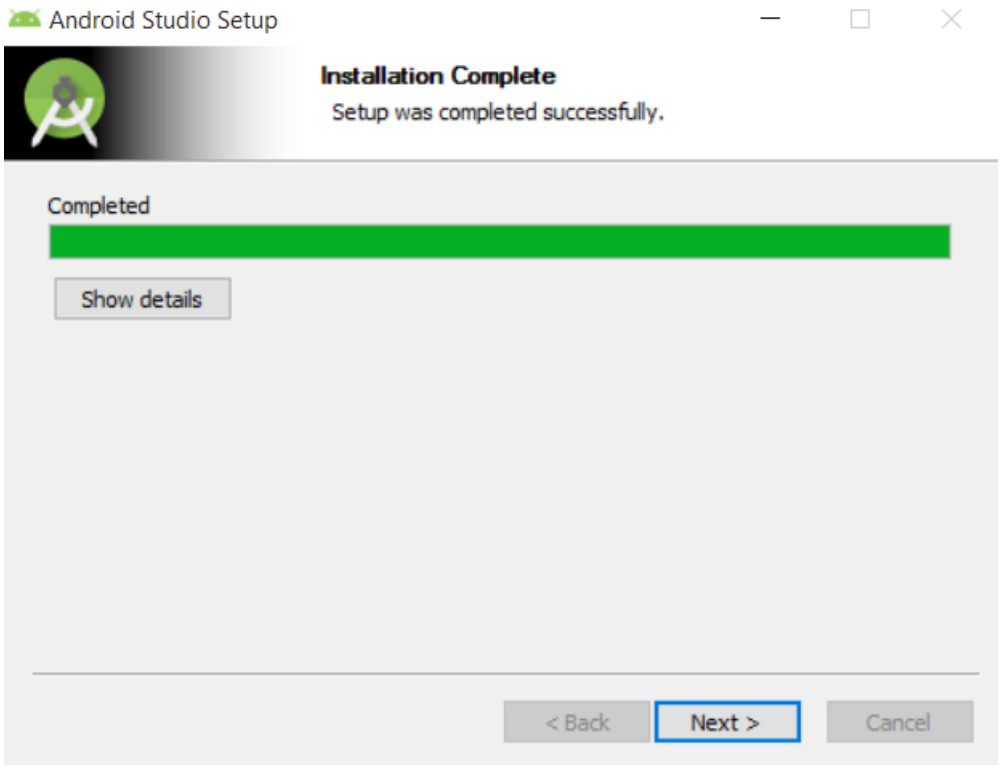


Esperar la carga

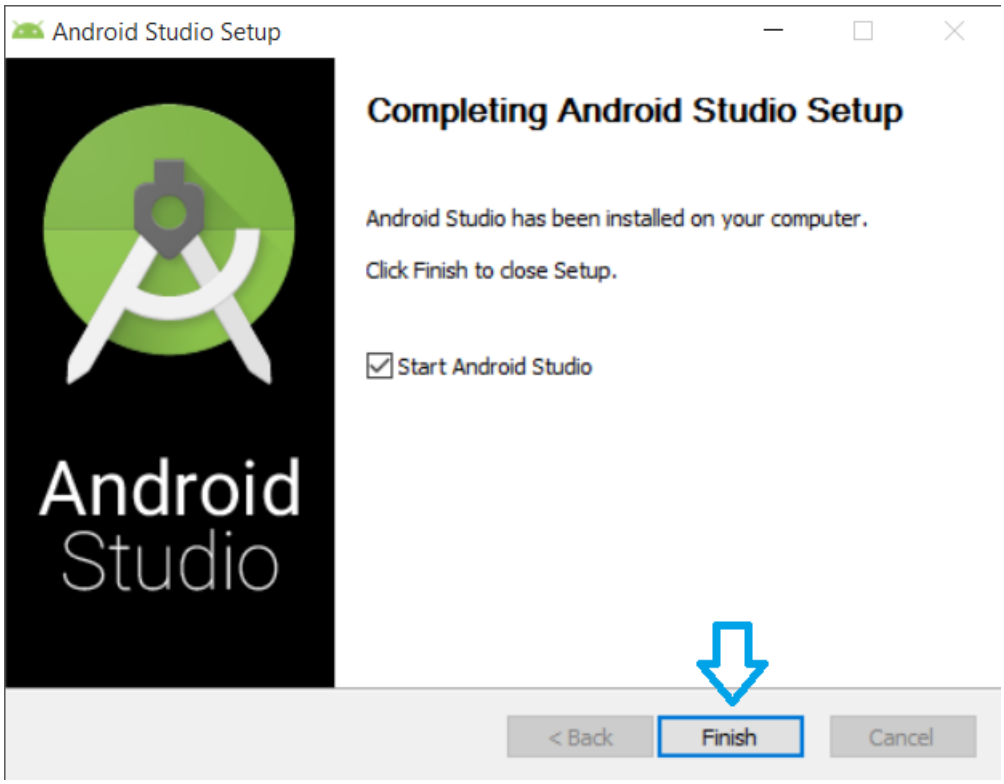


Después de la instalación dar clic en siguiente





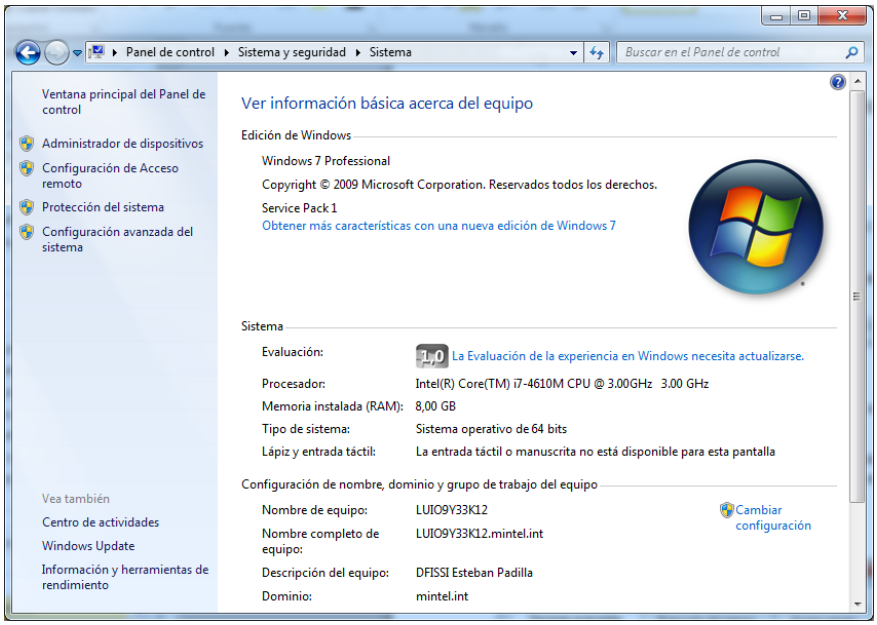
Finalizar y arrancar



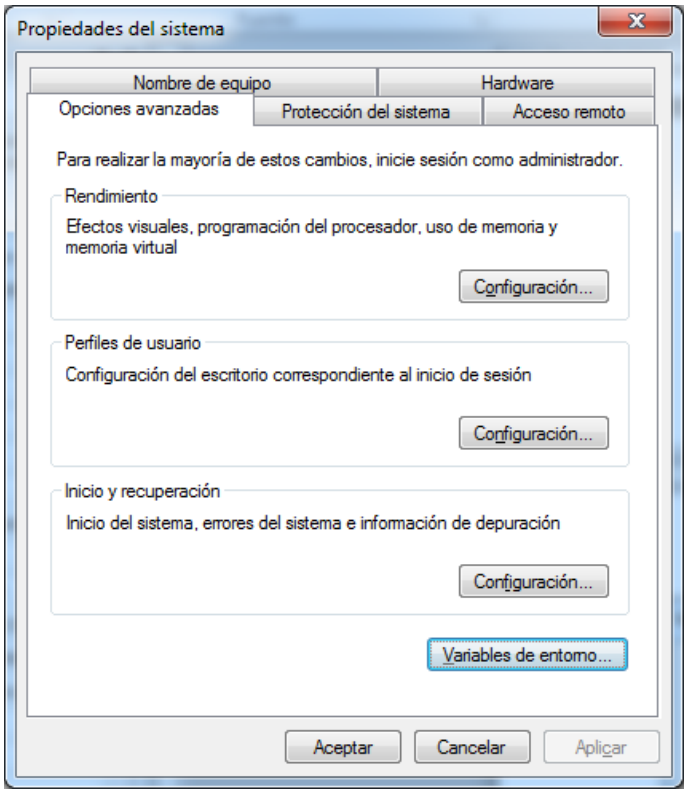
ANEXO 2 – REGISTRO VARIABLE DE ENTORNO

Continuando con la instalación se requiere registrar una nueva variable de entorno para esto ingresamos a propiedades del sistema:

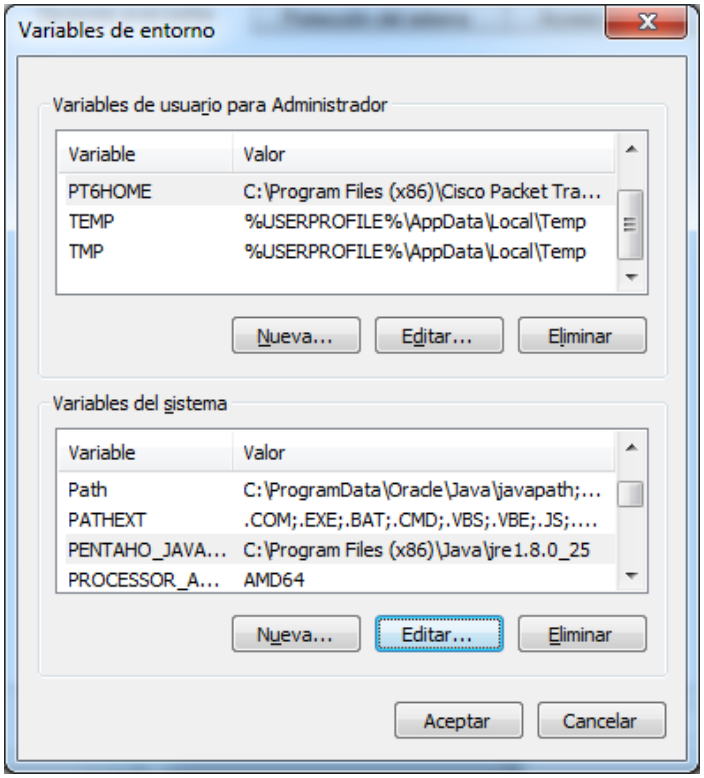
Inicio/Equipo/Propiedades del sistema/**Configuración avanzada del sistema**



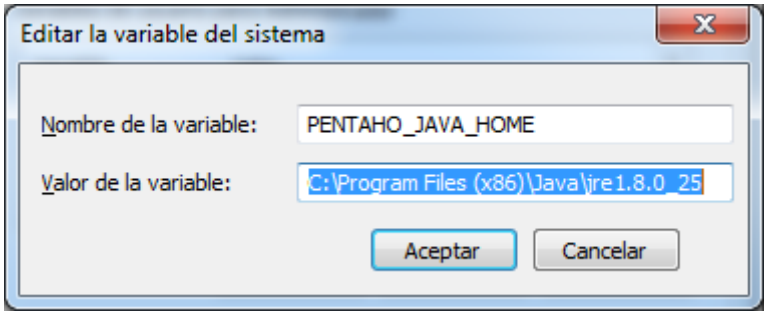
Opciones avanzadas/**Variables de entorno**



Variables del sistema/**Nueva**



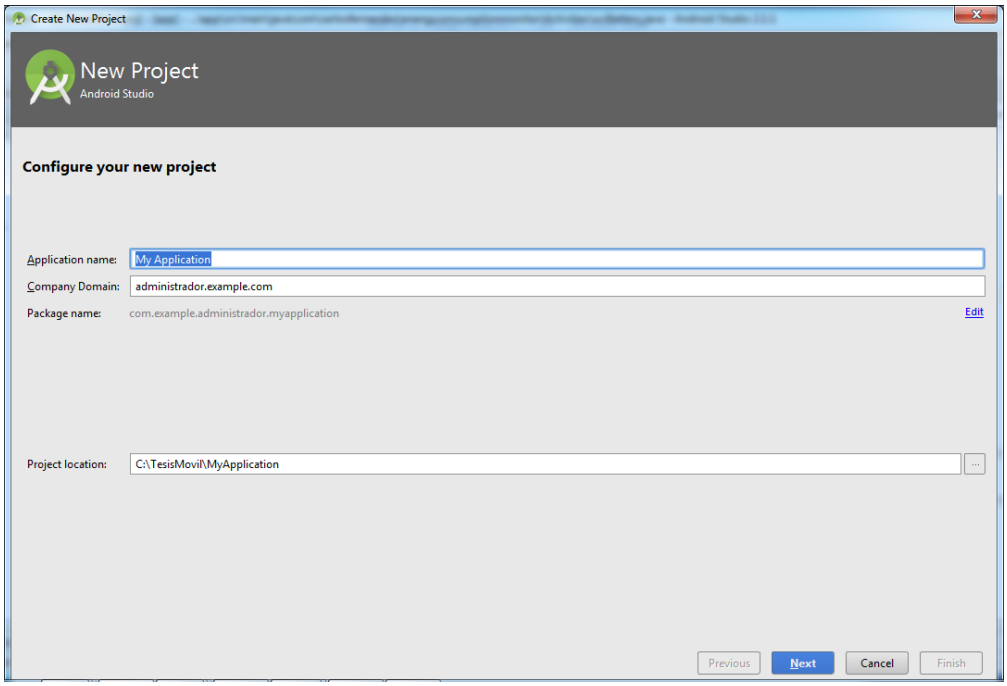
Ingresar el **Nombre de la variable** y el **Valor de la variable**



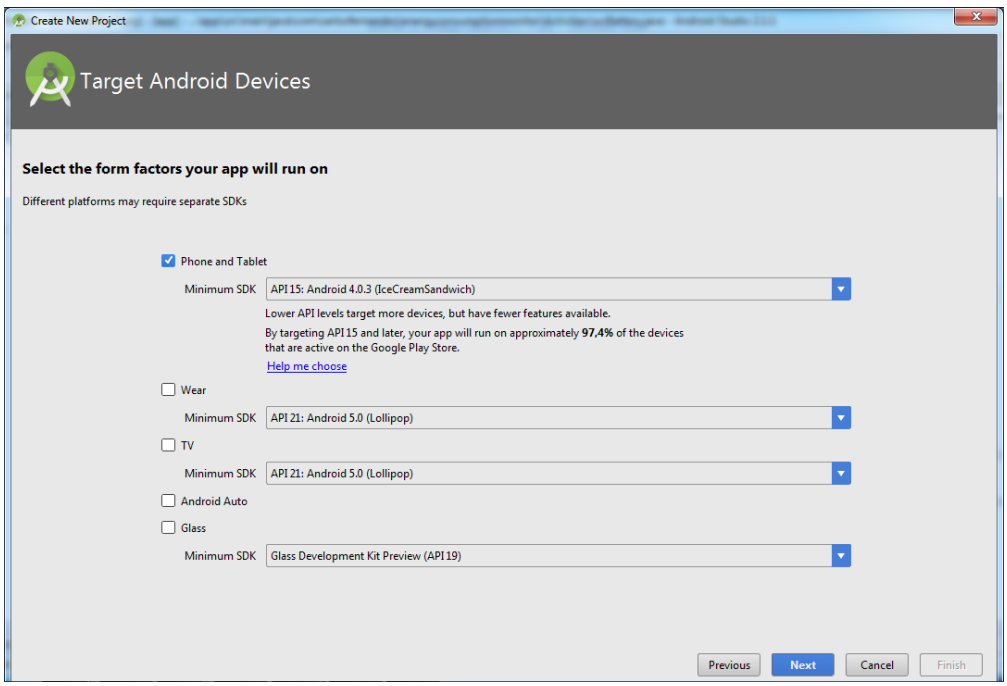
Finalmente ejecutamos Android Studio, en el caso que solicite importar una versión anterior marcamos la opción predeterminada.

ANEXO 3 – CREACIÓN DEL PROYECTO

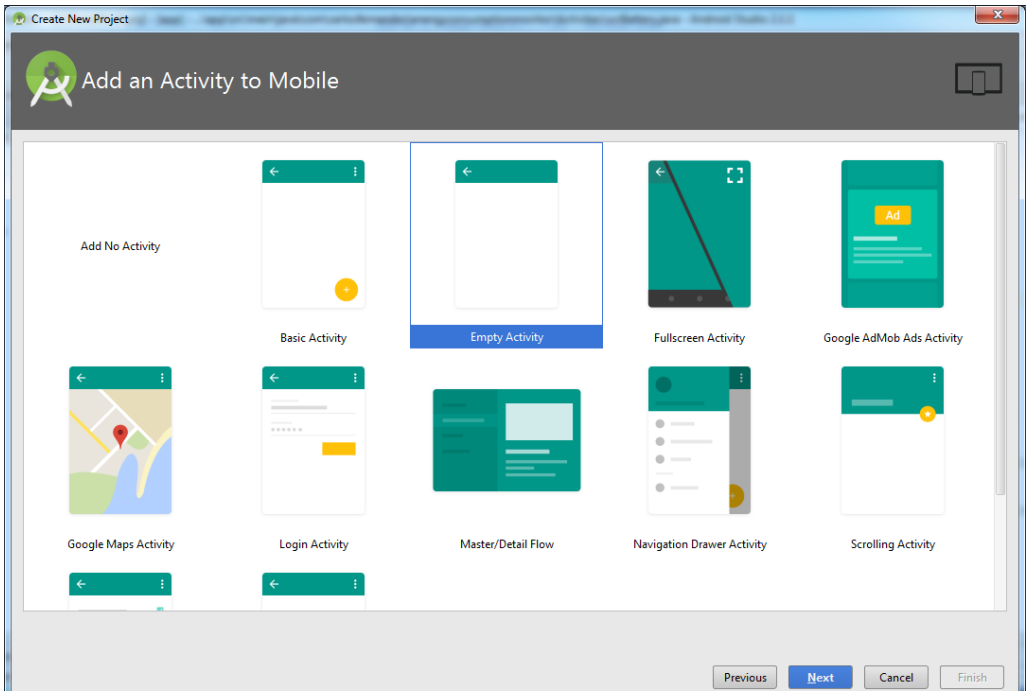
Se ingresa datos del **Nombre del proyecto**, **Empresa** y se selecciona la ubicación donde se almacenará todos los archivos del proyecto. Para este caso se definió la carpeta **C:\TesisMovil** luego clic en **Siguiente.**



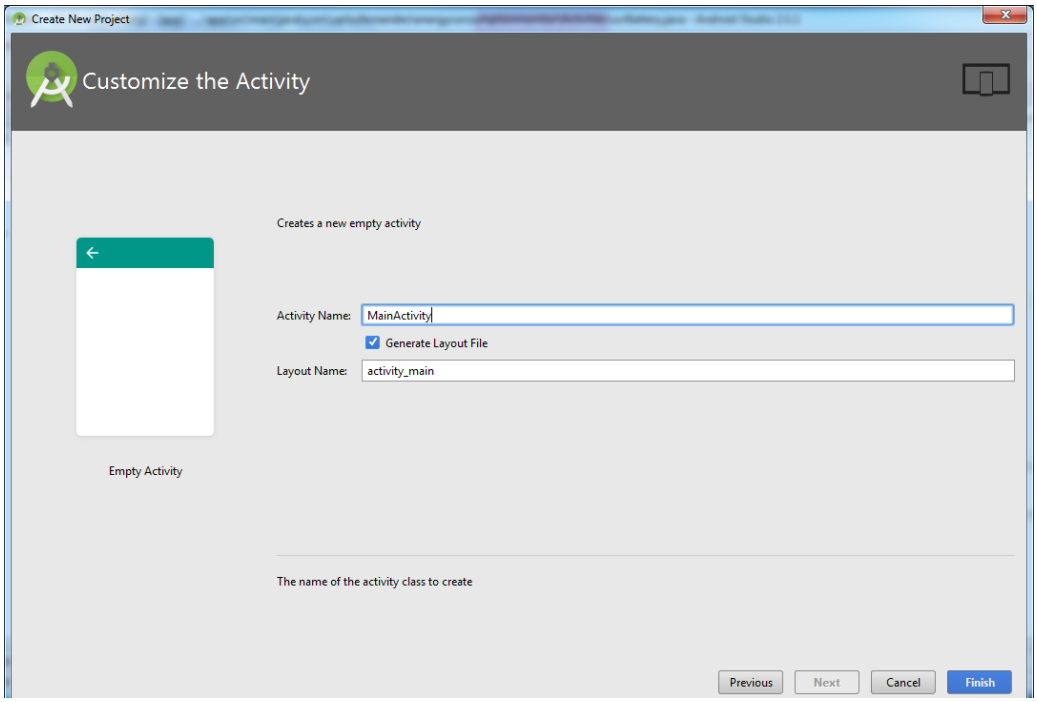
Para la construcción del proyecto activar la opción **"Phone and Tablet"** y del listado que aparece escoger el **SDK mínimo**, más actual, **API15: Android 4.0.3 (IceCreamSandwich)**, dejar las otras opciones activas por defecto.



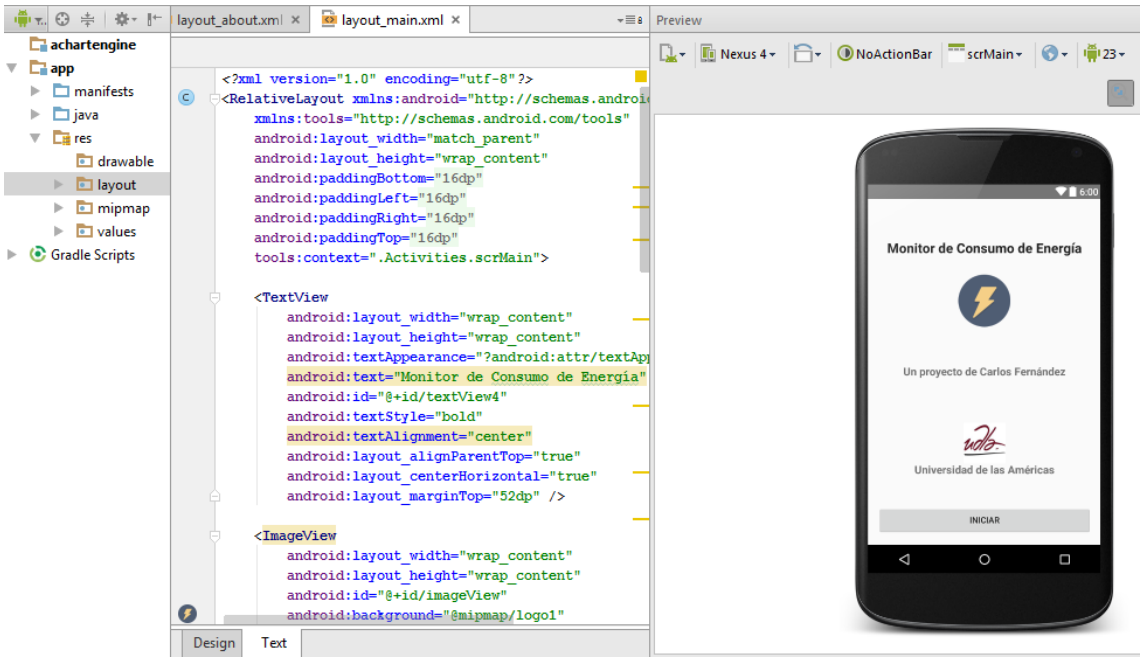
En esta parte Android Studio ofrece varios diseños para la aplicación, seleccionamos el patrón de ejecución visual que se ajuste a nuestro proyecto en función de las actividades que se tengan previsto que realice la aplicación. Click en siguiente.



Llegamos al último paso, esta pantalla permite cambiar el nombre de la actividad y del layout. Click en Final.

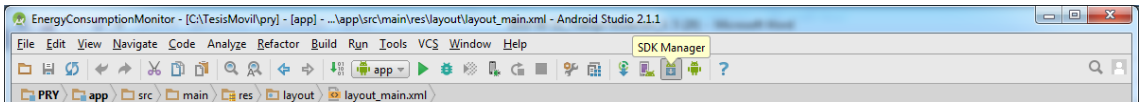


Como resultado final se tiene el entorno de desarrollo. En esta ocasión la imagen corresponde al diseño de la pantalla de inicio de la aplicación de monitoreo. Sin embargo se la incluye a modo de ejemplo con el fin de complementar este parte.

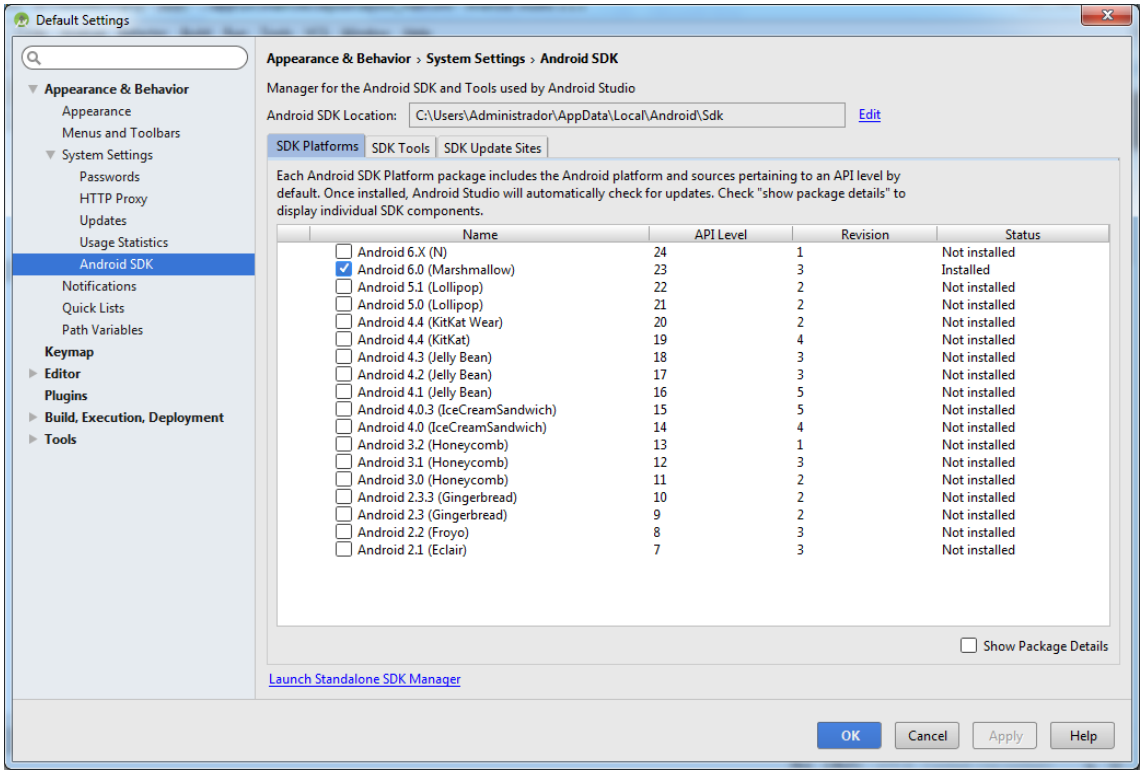


ANEXO 4 - INSTALACIÓN DE COMPLEMENTOS

Una vez realizado los pasos descritos en los Anexos del 1 al 3, se requiere conocer los complementos que se tienen instalados en el equipo, para esto damos click en el botón **SDK Manager** de la barra de herramientas.



Aparece la pantalla de configuración del sistema, nos situamos en el lado izquierdo de la pantalla de configuración y damos click en **Android SDK**, este a su vez muestra un listado de complementos a instalarse, seleccionamos el complemento que se ajusta a la aplicación a desarrollarse. Es importante señalar, que este procedimiento se realizó con anterioridad, por lo tanto, se incluye esta imagen con el fin de exponer cual fue el complemento que se utilizó (Android 6.0) en esta parte.



En resumen y como recomendación que realizan los desarrolladores para Android en los diferentes tutoriales, foros, etcétera., los componentes principales mínimos a instalar para Android SDK son: Tools, Platform-tools, Build-tools (versión actual), una o más versiones de la plataforma Android, Android Support Repository (extras), Google Repository (extras), Google Play Services (extras).

ANEXO 5 CODIGO FUENTE

scrAbout.java

```
/*
Pantalla de Acerca de...

Muestra los créditos de la aplicación
 */

package com.carlosfernandez.energyconsumptionmonitor.Activities;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.graphics.Color;
import android.os.BatteryManager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

import com.carlosfernandez.energyconsumptionmonitor.R;

public class scrAbout extends AppCompatActivity {

    //Evento cuando se crea la pantalla
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.layout_about);
        //Setea el color blanco al fondo de la pantalla
        getWindow().getDecorView().setBackgroundColor(Color.WHITE);
    }

}
```

scrBattery.java

```
/*
Pantalla de Monitor de Batería

Muestra información de la batería en tiempo real

- Nivel de carga
- Indicador de si está conectado a una fuente de energía
- Temperatura de la batería
- Voltaje de la batería

 */

package com.carlosfernandez.energyconsumptionmonitor.Activities;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.graphics.Color;
import android.os.BatteryManager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import com.carlosfernandez.energyconsumptionmonitor.R;
import java.math.BigDecimal;

public class scrBattery extends AppCompatActivity {

    //Instancia de los objetos TextView de la pantalla
    private TextView lblBLevel;
    private TextView lblBPlugged;
    private TextView lblBTemperature;
```

```
    private TextView lblBVoltage;
    private TextView lblCurrent;

    //BroadcastReceiver para manejar los datos de la batería en tiempo real
    private BroadcastReceiver mBatInfoReceiver = new BroadcastReceiver(){
        @Override
        public void onReceive(Context ctxt, Intent intent) {
            //Obtenemos la información de la batería en sus diversas características
            int level = intent.getIntExtra(BatteryManager.EXTRA_LEVEL, 0);
            int temperature = intent.getIntExtra(BatteryManager.EXTRA_TEMPERATURE, 0);
            //Para obtener el voltaje pasamos el valor obtenido de la batería por una
funcion de conversión
            float voltage =
getBatteryVoltage(intent.getIntExtra(BatteryManager.EXTRA_VOLTAGE, 0));

            //Hacemos una conversión del valor de temperatura entregado para mostrarlo
en formato de porcentaje
            float tmp = (float)temperature/10;
            float tmpF = tmp * 1.8f + 32;

            //Obtenemos el valor de Current / Corriente
            Bundle bundle = intent.getExtras();

            lblBLevel.setText(String.valueOf(level) + " %");

           // lblCurrent.setText(getValue().toString());

            //Obtenemos el estado (cargando o no)
            int status = intent.getIntExtra(BatteryManager.EXTRA_STATUS, -1);
            boolean isCharging = status == BatteryManager.BATTERY_STATUS_CHARGING ||
                    status == BatteryManager.BATTERY_STATUS_FULL;

            //Obtenemos el tipo de conexión de carga (USB o AC)
            int chargePlug = intent.getIntExtra(BatteryManager.EXTRA_PLUGGED, -1);
            boolean usbCharge = chargePlug == BatteryManager.BATTERY_PLUGGED_USB;
            boolean acCharge = chargePlug == BatteryManager.BATTERY_PLUGGED_AC;

            //Condicionamos si se está cargando y qué tipo de carga es
            if (isCharging)
            {
                if (usbCharge)
                {
                    lblBPlugged.setText("Cargando - USB");
                }
                else if (acCharge)
                {
                    lblBPlugged.setText("Cargando - AC");
                }
                else
                {
                    lblBPlugged.setText("Cargando");
                }
            }
            else
            {
                lblBPlugged.setText("No se está cargando");
            }

            //Mostramos la temperatura en formato de grados
            lblBTemperature.setText( tmp +  Character.toString ((char) 176) + " C  (" +
round(tmpF,1) +  Character.toString ((char) 176) + " F)");
            lblBVoltage.setText(voltage + " V");
        }
    };

    public float round(float d, int decimalPlace) {
        BigDecimal bd = new BigDecimal(Float.toString(d));
        bd = bd.setScale(decimalPlace, BigDecimal.ROUND_HALF_UP);
        return bd.floatValue();
    }
```

scrBluetooth.java

```
/*
Pantalla de medición del Bluetooth...

* Consumo mA
* Consumo mW
*
* Ref:
https://android.googlesource.com/platform/frameworks/base/+/master/core/res/res/xml/powe
r_profile.xml
*
 */

package com.carlosfernandez.energyconsumptionmonitor.Activities;

import android.app.DownloadManager;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.BatteryManager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import com.carlosfernandez.energyconsumptionmonitor.R;

import java.math.BigDecimal;

public class scrBluetooth extends AppCompatActivity {

    private TextView lblConStatus;
    private TextView lblConBlue;
    private TextView lblConsumBat;

    //private TextView lblS1;
    //private TextView lblS2;

    static double blueOnMA = 0;
    static int blueTransMA = 0;
    //BroadcastReceiver para manejar los datos del bluetooth en tiempo real
    private final BroadcastReceiver BluetoothReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {

            //Obtenemos la información de la acción del bluetooth
            final String action = intent.getAction();

//De acuerdo a la acción procede según las etiquetas

            if (action.equals("android.bluetooth.device.action.ACL_CONNECTED"))
            {
                lblConStatus.setText("Conectado / Transfiriendo");
                blueOnMA = 0.1;
                blueTransMA = 10;
                lblConsumBat.setText("Calculando potencia...");
                lblConBlue.setText((blueOnMA + blueTransMA) + " mA");
                CalcP(intent);

            }
            else if (action.equals("android.bluetooth.device.action.ACL_DISCONNECTED"))
            {
                lblConStatus.setText("Encendido / Sin conexión");
                blueOnMA = 0.1;
                blueTransMA = 0;
                lblConsumBat.setText("Calculando potencia...");
                lblConBlue.setText((blueOnMA + blueTransMA) + " mA");
                CalcP(intent);
```

```
            }
            else
            {
                if (action.equals(BluetoothAdapter.ACTION_STATE_CHANGED) ||
action.equals(BluetoothAdapter.ACTION_CONNECTION_STATE_CHANGED)) {
                    final int state = intent.getIntExtra(BluetoothAdapter.EXTRA_STATE,
                            BluetoothAdapter.ERROR);
                    final int stateC =
intent.getIntExtra(BluetoothAdapter.EXTRA_CONNECTION_STATE,
                            BluetoothAdapter.ERROR);
                    switch (state) {
                        case BluetoothAdapter.STATE_OFF:
                            lblConStatus.setText("Apagado");
                            blueOnMA = 0;
                            blueTransMA = 0;
                            lblConsumBat.setText("0 mW");
                            break;
                        case BluetoothAdapter.STATE_TURNING_OFF:
                            lblConStatus.setText("Apagando Bluetooth...");
                            break;
                        case BluetoothAdapter.STATE_ON:
                            lblConStatus.setText("Encendido / Sin conexión");
                            blueOnMA = 0.1;
                            blueTransMA = 0;
                            lblConsumBat.setText("Calculando potencia...");
                            CalcP(intent);
                            break;
                        case BluetoothAdapter.STATE_TURNING_ON:
                            lblConStatus.setText("Encendiendo Bluetooth...");
                            break;
                    }

                    switch (stateC)
                    {
                        case BluetoothAdapter.STATE_CONNECTED:
                            lblConStatus.setText("Conectado / Transfiriendo");
                            blueOnMA = 0.1;
                            blueTransMA = 10;
                            lblConsumBat.setText("Calculando potencia...");
                            CalcP(intent);
                            break;

                        case BluetoothAdapter.STATE_DISCONNECTED:
                            lblConStatus.setText("Encendido / Sin conexión");
                            blueOnMA = 0.1;
                            blueTransMA = 0;
                            lblConsumBat.setText("Calculando potencia...");
                            CalcP(intent);
                            break;
                    }
                    //
                    lblConBlue.setText((blueOnMA+blueTransMA) + " mA");
                }
            }


        }
    };
//Controla los eventos de la batería
    private BroadcastReceiver BatteryReceiver = new BroadcastReceiver()
    {
        @Override
        public void onReceive(Context ctxt, Intent intent) {
            CalcP(intent);
        }
    };

    //Función para calcular la potencia
    private void CalcP (Intent intent)
    {
        int Q = intent.getIntExtra(BatteryManager.EXTRA_LEVEL, 0);
```

scrBluetoothGraph.java

```
package com.carlosfernandez.energyconsumptionmonitor.Activities;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.graphics.Color;
import android.os.BatteryManager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.LinearLayout;
import android.widget.TextView;
import com.carlosfernandez.energyconsumptionmonitor.R;
import com.carlosfernandez.energyconsumptionmonitor.Graph.CubicLineGraph;
import org.achartengine.GraphicalView;
import java.util.Random;

public class scrBluetoothGraph extends AppCompatActivity {

    private static GraphicalView view;
    private CubicLineGraph line = new CubicLineGraph();
    private static Thread thread;
    private static double powerValue = 0;

    static float blueOnMA = 0;
    static float blueTransMA = 0;

    //Realiza la misma función que la pantalla del Bluetooth (scrBluetooth) sólo tomando
los valores para el gráfico
    private final BroadcastReceiver BluetoothReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            final String action = intent.getAction();

            if (action.equals("android.bluetooth.device.action.ACL_CONNECTED"))
            {
                blueOnMA = 0.1f;
                blueTransMA = 10f;
                CalcP(intent);
            }
            else if (action.equals("android.bluetooth.device.action.ACL_DISCONNECTED"))
            {
                blueOnMA = 0.1f;
                blueTransMA = 0;
                CalcP(intent);
            }
            else
            {
                if (action.equals(BluetoothAdapter.ACTION_STATE_CHANGED) ||
action.equals(BluetoothAdapter.ACTION_CONNECTION_STATE_CHANGED)) {
                    final int state = intent.getIntExtra(BluetoothAdapter.EXTRA_STATE,
                            BluetoothAdapter.ERROR);
                    final int stateC =
intent.getIntExtra(BluetoothAdapter.EXTRA_CONNECTION_STATE,
                            BluetoothAdapter.ERROR);
                    switch (state) {
                        case BluetoothAdapter.STATE_OFF:
                            blueOnMA = 0;
                            blueTransMA = 0;
                            break;
                        case BluetoothAdapter.STATE_ON:
                            blueOnMA = 0.1f;
                            break;
                    }

                    switch (stateC)
                    {
                        case BluetoothAdapter.STATE_CONNECTED:
                            blueOnMA = 0.1f;
                            blueTransMA = 10f;
```

```
                            break;

                        case BluetoothAdapter.STATE_DISCONNECTED:
                            blueOnMA = 0.1f;
                            blueTransMA = 0;
                            break;
                    }
                }
            }
        }
    };

    private BroadcastReceiver BatteryReceiver = new BroadcastReceiver()
    {
        @Override
        public void onReceive(Context ctxt, Intent intent) {
            CalcP(intent);
        }
    };

    private void CalcP(Intent intent)
    {
        int Q = intent.getIntExtra(BatteryManager.EXTRA_LEVEL, 0);
        float V = getBatteryVoltage(intent.getIntExtra(BatteryManager.EXTRA_VOLTAGE,
0));
        float P = V * (blueOnMA+blueTransMA) * Q / 100;
        //
        powerValue=P;
    }

    protected float getBatteryVoltage(int v)
    {
        if (v > 1000) return v / 1000f;
        else
            return v;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.layout_bluetooth_grap);
        getWindow().getDecorView().setBackgroundColor(Color.BLACK);
        TextView lblTitle = (TextView) this.findViewById(R.id.title);
        TextView lblstip = (TextView) this.findViewById(R.id.lblstip);
        lblTitle.setTextColor(Color.WHITE);
        lblstip.setTextColor(Color.WHITE);

        Intent i = getIntent();
        blueOnMA = i.getFloatExtra("blueOnMA",0);
        blueTransMA = i.getFloatExtra("blueTransMA",0);

        BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        if (mBluetoothAdapter == null) {

        } else {

            if (mBluetoothAdapter.isEnabled()) {
                blueOnMA = 0.1f;
            }
            else
            {
                blueOnMA = 0;
            }

            CalcP(this.getIntent());

            this.registerReceiver(this.BluetoothReceiver, new
IntentFilter(BluetoothAdapter.ACTION_STATE_CHANGED));
            this.registerReceiver(this.BluetoothReceiver, new
IntentFilter(BluetoothAdapter.ACTION_CONNECTION_STATE_CHANGED));
            this.registerReceiver(this.BluetoothReceiver, new
IntentFilter(BluetoothDevice.ACTION_ACL_CONNECTED));
```

scrGPS.java

```
/*
Pantalla de medición del GPS...

* Consumo mA
* Consumo mW
*
* Ref:
https://android.googlesource.com/platform/frameworks/base/+/master/core/res/res/xml/powe
r_profile.xml
*
 */
package com.carlosfernandez.energyconsumptionmonitor.Activities;

import android.bluetooth.BluetoothAdapter;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.graphics.Color;
import android.location.LocationManager;
import android.os.BatteryManager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AbsoluteLayout;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;



import com.carlosfernandez.energyconsumptionmonitor.R;

import org.achartengine.ChartFactory;
import org.achartengine.GraphicalView;
import org.achartengine.chart.*;
import org.achartengine.model.TimeSeries;
import org.achartengine.model.XYMultipleSeriesDataset;
import org.achartengine.model.XYSeries;
import org.achartengine.renderer.XYMultipleSeriesRenderer;
import org.achartengine.renderer.XYSeriesRenderer;
import org.w3c.dom.Text;

import java.util.Date;


public class scrGPS extends AppCompatActivity {

    private TextView lblConStatus;
    private TextView lblConsumBat;
    private TextView lblConsumGPS;

    private BroadcastReceiver GPSReceiver = new BroadcastReceiver()
    {
        @Override
        public void onReceive( Context context, Intent intent )
        {
            final LocationManager manager = (LocationManager) context.getSystemService(
Context.LOCATION_SERVICE );
            if (manager.isProviderEnabled( LocationManager.GPS_PROVIDER ) ) {
                lblConStatus.setText("Activado");
                lblConsumGPS.setText("50 mA");
                lblConsumBat.setText("Calculando potencia...");
                CalcP(context,intent);
            }else{
                lblConStatus.setText("Desactivado");
                lblConsumGPS.setText("0 mA");
                lblConsumBat.setText("0 mW");
            }
        }
    };
```

```
    private BroadcastReceiver BatteryReceiver = new BroadcastReceiver()
    {
        @Override
        public void onReceive(Context ctxt, Intent intent) {
            CalcP(ctxt,intent);
         }
    };

    private void CalcP(Context ctxt, Intent intent)
    {
        final LocationManager manager = (LocationManager) ctxt.getSystemService(
Context.LOCATION_SERVICE );
        if (manager.isProviderEnabled( LocationManager.GPS_PROVIDER ) ) {
            int Q = intent.getIntExtra(BatteryManager.EXTRA_LEVEL, 0);
            float V = getBatteryVoltage(intent.getIntExtra(BatteryManager.EXTRA_VOLTAGE,
0));
            float P = V * 50 * Q / 100;
            //
            lblConsumBat.setText(P + " mW");
        }
        else
        {
            lblConsumBat.setText("0 mW");
        }
    }

    protected float getBatteryVoltage(int v)
    {
        if (v > 1000) return v / 1000f;
        else
            return v;
    }


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.layout_gps);
        lblConStatus = (TextView) this.findViewById(R.id.lblConStatus);
        lblConsumGPS = (TextView) this.findViewById(R.id.lblConGPS);
        lblConsumBat = (TextView) this.findViewById(R.id.lblConBat);
        //
        this.registerReceiver(this.GPSReceiver, new
IntentFilter(LocationManager.PROVIDERS_CHANGED_ACTION));
        this.registerReceiver(this.BatteryReceiver, new
IntentFilter(Intent.ACTION_BATTERY_CHANGED));
        //
        LocationManager locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);

        if (locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)){
            lblConStatus.setText("Activado");
            lblConsumGPS.setText("50 mA");
            lblConsumBat.setText("Calculando potencia...");
            //
            CalcP(this.getApplicationContext(),this.getIntent());
        }else{
            lblConStatus.setText("Desactivado");
            lblConsumGPS.setText("0 mA");
            lblConsumBat.setText("0 mW");
        }

       }

    public void menuItem1_Click (View view)
    {
        showAlert("Indica si está encendido o no el GPS del dispositivo");
    }

    public void menuItem2_Click (View view)
```

scrGPSGraph.java

```
package com.carlosfernandez.energyconsumptionmonitor.Activities;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.graphics.Color;
import android.location.LocationManager;
import android.os.BatteryManager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.LinearLayout;
import android.widget.TextView;
import com.carlosfernandez.energyconsumptionmonitor.R;
import com.carlosfernandez.energyconsumptionmonitor.Graph.CubicLineGraph;
import org.achartengine.GraphicalView;
import java.util.Random;

public class scrGPSGraph extends AppCompatActivity {

    private static GraphicalView view;
    private CubicLineGraph line = new CubicLineGraph();
    private static Thread thread;
    private static double powerValue = 0;


    private BroadcastReceiver BatteryReceiver = new BroadcastReceiver()
    {
        @Override
        public void onReceive(Context ctxt, Intent intent) {
            final LocationManager manager = (LocationManager) ctxt.getSystemService(
Context.LOCATION_SERVICE );
            if (manager.isProviderEnabled( LocationManager.GPS_PROVIDER ) ) {
                int Q = intent.getIntExtra(BatteryManager.EXTRA_LEVEL, 0);
                float V =
getBatteryVoltage(intent.getIntExtra(BatteryManager.EXTRA_VOLTAGE, 0));
                float P = V * 50 * Q / 100;
                //
                powerValue=P;
            }
            else
            {
                powerValue=0;
            }
        }
    };

    protected float getBatteryVoltage(int v)
    {
        if (v > 1000) return v / 1000f;
        else
            return v;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.layout_gpsgraph);
        getWindow().getDecorView().setBackgroundColor(Color.BLACK);
        TextView lblTitle = (TextView) this.findViewById(R.id.title);
        TextView lblstip = (TextView) this.findViewById(R.id.lblstip);
        lblTitle.setTextColor(Color.WHITE);
        lblstip.setTextColor(Color.WHITE);
        //
        thread = new Thread() {
            public void run()
            {
                Random rnd = new Random();
                int counter = 1;

                while (true)
                {
                    try {
                        Thread.sleep(1000);
```

```
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                    line.addNewPoints(counter,powerValue);

                    view.repaint();

                    counter++;
                }
            }
        };
        thread.start();
        //
        this.registerReceiver(this.BatteryReceiver, new
IntentFilter(Intent.ACTION_BATTERY_CHANGED));
    }

    @Override
    protected void onStart() {
        super.onStart();
        view = line.getView(this);
        LinearLayout chartContainer = (LinearLayout) findViewById(R.id.chart_container);
        chartContainer.addView(view);
    }



}
```

scrMain.java

```
/*
Pantalla Principal

Es la pantalla de inicio de la aplicación

 */
package com.carlosfernandez.energyconsumptionmonitor.Activities;

import android.content.Intent;
import android.graphics.Color;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;

import com.carlosfernandez.energyconsumptionmonitor.R;

public class scrMain extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.layout_main);
        //Setea el color blanco al fondo de la pantalla
        getWindow().getDecorView().setBackgroundColor(Color.WHITE);
    }

    //Pasa a la pantalla del menú
    public void GoNextScreen (View view)
    {
        Intent intent = new Intent(this,scrMenu.class);
        startActivity(intent);
    }

}
```

scrMain.java

```
/*
Pantalla de Menú de Opciones

Muestra el menú de opciones de la aplicación

 */
package com.carlosfernandez.energyconsumptionmonitor.Activities;
```

```
import android.content.Intent;
import android.graphics.Color;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

import com.carlosfernandez.energyconsumptionmonitor.R;


public class scrMenu extends AppCompatActivity {

    //Cuando se crea la pantalla
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.layout_menu);
        getWindow().getDecorView().setBackgroundColor(Color.WHITE);
        //
        showAlert("Bienvenido al menú de opciones, pulsa sobre la opción que deseas
abrir");
    }

    //Cuando se hace clic sobre la opción de ver el Monitor de Batería
    public void menuItem1_Click (View view)
    {
        Intent intent = new Intent(this,scrBattery.class);
        startActivity(intent);
    }

    //Cuando se hace clic sobre la opción de ver el Monitor de Componentes (Bluetooth y
GPS)
    public void menuItem2_Click (View view)
    {
        //Abre la pantalla de menu de componentes
        Intent intent = new Intent(this,scrMenuComp.class);
        startActivity(intent);
    }

    //Cuando se hace clic sobre la opción de ver Total de Consumo
    public void menuItem3_Click (View view)
    {
        //Abre la pantalla de total de consumo
        Intent intent = new Intent(this,scrTotalCom.class);
        startActivity(intent);
    }

    //Cuando se hace clic sobre la opción de Acerca de...
    public void menuItem4_Click (View view)
    {
        //Abre la pantalla de acerca de...
        Intent intent = new Intent(this,scrAbout.class);
        startActivity(intent);
    }

    //Función para mostrar mensajes sobre la pantalla tipo Tooltip
    public void showAlert (String mensaje)
    {
        Toast.makeText(getApplicationContext(), mensaje, Toast.LENGTH_LONG).show();
    }
}
```

scrMenuComp.java

```
/*
Pantalla de Menú de Componentes

Muestra el menú de componentes a medir en la aplicación

 */
package com.carlosfernandez.energyconsumptionmonitor.Activities;

import android.content.Intent;
```

```
import android.graphics.Color;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

import com.carlosfernandez.energyconsumptionmonitor.R;

public class scrMenuComp extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.layout_menu_comp);
    }

    //Función para mostrar mensajes sobre la pantalla tipo Tooltip
    public void showAlert (String mensaje)
    {
        Toast.makeText(getApplicationContext(), mensaje, Toast.LENGTH_LONG).show();
    }

    //Cuando se hace clic sobre la opción de bluetooth
    public void menuItem1_Click (View view)
    {
        //Abre la pantalla de Bluetooth...
        Intent intent = new Intent(this,scrBluetooth.class);
        startActivity(intent);
    }

    //Cuando se hace clic sobre la opción de gps
    public void menuItem2_Click (View view)
    {
        //Abre la pantalla de Bluetooth...
        Intent intent = new Intent(this,scrGPS.class);
        startActivity(intent);
    }
}
```

scrTotalCom.java

```
package com.carlosfernandez.energyconsumptionmonitor.Activities;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.location.LocationManager;
import android.os.BatteryManager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import com.carlosfernandez.energyconsumptionmonitor.R;

import java.math.BigDecimal;

public class scrTotalCom extends AppCompatActivity {

    private TextView lblConStatus;
    private TextView lblConBlue;
    private TextView lblConsumBat;

    static double blueOnMA = 0;
    static int blueTransMA = 0;
    static int gpsOnMA = 0;

    static String bluStatus = "";
    static String gpsStatus = "";
```

```
    //Controla los eventos del Bluetooth
    private final BroadcastReceiver BluetoothReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            final String action = intent.getAction();

            if (action.equals("android.bluetooth.device.action.ACL_CONNECTED"))
            {
                bluStatus = "B. Transfiriendo";
                blueOnMA = 0.1;
                blueTransMA = 10;
                CalcP(intent);
                lblConStatus.setText(gpsStatus + " | " + bluStatus);
                lblConBlue.setText((blueOnMA + blueTransMA + gpsOnMA) + " mA");
            }
            else if (action.equals("android.bluetooth.device.action.ACL_DISCONNECTED"))
            {
                bluStatus = "B. Encendido";
                blueOnMA = 0.1;
                blueTransMA = 0;
                CalcP(intent);
                lblConStatus.setText(gpsStatus + " | " + bluStatus);
                lblConBlue.setText((blueOnMA+blueTransMA+gpsOnMA) + " mA");
            }
            else
            {
                if (action.equals(BluetoothAdapter.ACTION_STATE_CHANGED) ||
action.equals(BluetoothAdapter.ACTION_CONNECTION_STATE_CHANGED)) {
                    final int state = intent.getIntExtra(BluetoothAdapter.EXTRA_STATE,
                            BluetoothAdapter.ERROR);
                    final int stateC =
intent.getIntExtra(BluetoothAdapter.EXTRA_CONNECTION_STATE,
                            BluetoothAdapter.ERROR);
                    switch (state) {
                        case BluetoothAdapter.STATE_OFF:
                            bluStatus = "B. Apagado";
                            blueOnMA = 0;
                            blueTransMA = 0;
                            CalcP(intent);
                            break;
                        case BluetoothAdapter.STATE_ON:
                            bluStatus = "B. Encendido";
                            blueOnMA = 0.1;
                            CalcP(intent);
                            break;

                    }

                    switch (stateC)
                    {
                        case BluetoothAdapter.STATE_CONNECTED:
                            bluStatus = "B. Transfiriendo";
                            blueOnMA = 0.1;
                            blueTransMA = 10;
                            CalcP(intent);
                            break;

                        case BluetoothAdapter.STATE_DISCONNECTED:
                            bluStatus = "B. Encendido";
                            blueOnMA = 0.1;
                            blueTransMA = 0;
                            CalcP(intent);
                            break;
                    }
                    //
                    CalcP(intent);
                    lblConStatus.setText(gpsStatus + " | " + bluStatus);
                    lblConBlue.setText((blueOnMA+blueTransMA+gpsOnMA) + " mA");
                }
            }
```

CubicLineGraph.java

```
package com.carlosfernandez.energyconsumptionmonitor.Graph;

/**
 * Created by DEVPC on 6/11/2016.
 */
import org.achartengine.ChartFactory;
import org.achartengine.GraphicalView;
import org.achartengine.chart.CubicLineChart;
import org.achartengine.chart.PointStyle;
import org.achartengine.model.TimeSeries;
import org.achartengine.model.XYMultipleSeriesDataset;
import org.achartengine.model.XYSeries;
import org.achartengine.renderer.XYMultipleSeriesRenderer;
import org.achartengine.renderer.XYSeriesRenderer;

import android.content.Context;
import android.graphics.Color;


public class CubicLineGraph {

    private GraphicalView view;

    private XYSeries dataset = new XYSeries("GPS");
    private XYMultipleSeriesDataset mDataset = new XYMultipleSeriesDataset();

    private XYSeriesRenderer renderer = new XYSeriesRenderer();
    private XYMultipleSeriesRenderer mRenderer = new XYMultipleSeriesRenderer();

    public CubicLineGraph()
    {
        mDataset.addSeries(dataset);

        int clr = Color.GREEN;
        renderer.setColor(clr);
        renderer.setFillBelowLine(true);
        renderer.setFillBelowLineColor(((clr >> 1) & 0x7F7F7F) |
                (clr & 0xFF000000));
        renderer.setShowLegendItem(false);

        mRenderer.setXAxisMin(1);
       // mRenderer.setYAxisMin(0);
       // mRenderer.setYAxisMax(5);
        mRenderer.setYTitle("Valor de la medida (mW)");
        mRenderer.setXTitle("Tiempo de medición (segundos)");
        mRenderer.setPanEnabled(false);
        mRenderer.setZoomRate(0.2f);
        mRenderer.setZoomEnabled(false, false);
        mRenderer.setInScroll(true);

        mRenderer.addSeriesRenderer(renderer);
    }

    public GraphicalView getView(Context context)
    {
        view = new GraphicalView(context,new CubicLineChart(mDataset,mRenderer,0.5f));
        return view;
    }

    public void addNewPoints(int x, double y)
    {
        if ((y+5) > mRenderer.getYAxisMax()) {
            mRenderer.setYAxisMax(y+5);
        }

        if ((y-5) < mRenderer.getYAxisMin()) {
            if (y < 1)
            {
                mRenderer.setYAxisMin(0);
            }
            else
            {
```

```
                mRenderer.setYAxisMin(y-5);
            }

        }
        dataset.add(x, y);

    }

}
```

layout_about.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.carlosfernandez.energyconsumptionmonitor.Activities.scrAbout">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageView4"
        android:background="@mipmap/udla"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="62dp" />


    <TableLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@+id/imageView4"
        android:stretchColumns="1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="130px">


        <TableRow
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="40px"
            android:gravity="center_vertical">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:text="Un proyecto de Carlos Fernández"
                android:id="@+id/txt1"
                android:layout_centerHorizontal="true"
                android:layout_centerVertical="true"
                android:layout_gravity="center_vertical|center_horizontal" />
        </TableRow>

        <TableRow
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="40px"
            android:gravity="center_vertical">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:text="Ingeniería en"
                android:id="@+id/txt3"
                android:layout_centerHorizontal="true"
```

ANEXO 6 PAQUETES Y OBJETOS UTILIZADOS EN EL DESARROLLO

Tabla 18. Paquetes y objetos

Paquete	Objeto	Tipo	Descripción
Activities	scrAbout	Activity	Código de pantalla para él Acerca de…
Activities	scrBattery	Activity	Código de pantalla para el Monitor de Batería
Activities	scrBluetooth	Activity	Código de pantalla para el Monitor de Bluetooth
Activities	scrBluetoothGraph	Activity	Código de pantalla para el Gráfico del Bluetooth
Activities	scrGPS	Activity	Código de pantalla para el Monitor de GPS
Activities	scrGPSGraph	Activity	Código de pantalla para el Gráfico de GPS
Activities	scrMain	Activity	Código de pantalla para la entrada principal
Activities	scrMenu	Activity	Código de pantalla para el Menú de Opciones
Activities	scrMenuComp	Activity	Código de pantalla para el Menú de Componentes
Activities	scrTotalCom	Activity	Código de pantalla para el Total de Consumo de Componentes
Layout	layout_about	Layout	Diseño de pantalla para describir Acerca de…
Layout	layout_battery	Layout	Diseño de pantalla para el Monitor de Batería
Layout	layout_bluetooth	Layout	Diseño de pantalla para el Monitor del Bluetooth
Layout	layout_bluetooth_graph	Layout	Diseño de pantalla para el Gráfico del Bluetooth
Layout	layout_gps	Layout	Diseño de pantalla para el Monitor de GPS
Layout	layout_gpsgraph	Layout	Diseño de pantalla para el Gráfico de GPS
Layout	layout_main	Layout	Diseño de pantalla para la entrada principal
Layout	layout_menu	Layout	Diseño de pantalla para el Menú de Opciones
Layout	layout_menu_comp	Layout	Diseño de pantalla para el Menú de Componentes
Layout	layout_totalc	Layout	Diseño de pantalla para el Total de Consumo de Componentes
Graph	CubicLineGraph	Class	Código de Clase para manejar el gráfico en las pantalla
Library	achartengine	Library	Librería para graficar en Android

ANEXO 7 MEDIDAS INICIALES

Tiempo descarga (Horas)	Nivel de Carga% dispositivo	Tiempo descarga CPU	Tiempo descarga Pantalla	Voltaje (V)	Temperatura ° C	Estado
2	100	100,0	100,0	3,859	31,9	Descargando
4	98	95,5	75,0	3,898	32,4	Descargando
6	95	91,0	50,0	3,847	33,1	Descargando
8	93	86,5	25,0	3,898	33,6	Descargando
10	90	82,0	0,0	3,843	33,9	Descargando
12	88	77,5		3,839	34,4	Descargando
14	85	73,0		3,804	34,6	Descargando
16	83	68,5		3,824	34,5	Descargando
18	80	64,0		3,804	34,5	Descargando
20	78	59,5		3,785	34,8	Descargando
22	75	55,0		3,796	34,4	Descargando
24	73	50,5		3,734	34,2	Descargando
26	70	46,0		3,714	34,4	Descargando
28	68	41,5		3,765	35	Descargando
30	65	37,0		3,769	33,2	Descargando
32	63	32,5		3,777	31,2	Descargando
34	60	28,0		3,765	31,9	Descargando
36	58	23,5		3,761	32,1	Descargando
38	55	19,0		3,761	32,1	Descargando
40	53	14,5		3,761	33	Descargando
42	50	10,0		3,699	33,1	Descargando
44	48	5,5		3,707	33	Descargando
46	45	1,0		3,832	31,4	Descargando
48	43			3,835	32	Descargando
50	40			3,855	32,9	Descargando
52	38			3,882	33,4	Descargando
54	35			3,894	33,9	Descargando
56	33			3,792	34,5	Descargando
58	30			3,894	34,8	Descargando
60	28			3,898	35,1	Descargando
62	25			3,898	35,2	Descargando
64	23			3,691	35,5	Descargando
66	20			3,703	34,4	Descargando
68	18			3,695	32,5	Descargando
70	15			3,738	31,9	Descargando
72	13			3,691	31	Descargando
74	10			3,773	32,3	Descargando
76	8			3,785	32,9	Descargando
78	5			3,765	33,4	Descargando
80	3			3,722	33,5	Descargando
82	0			3,761	34,2	Descargando

Tabla 19. Medidas iniciales

ANEXO 8 MEDIDAS BLUETOOTH

Apagado			Encendido			Encendido y paridad			Encendido y transfiriendo	
Potencia mW	Tiempo X		Potencia mW	Tiempo X		Potencia mW	Tiempo X		Potencia mW	Tiempo X
0	10		0,1	10		0,1	10		0,1	10
0	20		0,1	20		0,1	20		0,1	20
0	30		0,1	30		0,1	30		0,1	30
0	40		0,1	40		0,1	40		0,1	40
0	50		0,1	50		0,1	50		0,1	50
0	60		0,1	60		0,1	60		0,1	60
0	70		0,1	70		0,1	70		0,1	70
0	80		0,1	80		0,1	80		0,1	80
0	90		0,1	90		0,1	90		0,1	90
0	100		0,1	100		0,1	100		0,1	100
0	110		0,1	110		0,1	110		0,1	110
0	120		0,1	120		0,1	120		0,1	120
0	130		0,1	130		0,1	130		0,1	130
0	140		0,1	140		0,1	140		0,1	140
0	150		0,1	150		0,1	150		0,1	150
0	160		0,1	160		0,1	160		0,1	160
0	170		0,1	170		0,1	170		0,1	170
0	180		0,1	180		0,1	180		0,1	180
0	190		0,1	190		0,1	190		0,1	190
0	200		0,1	200		0,1	200		0,1	200
						10	210		10	210
						10	220		10	220
						10	230		10	230
						10	240		10	240
						10	250		10	250
						10	260		10	260
						10	270		10	270
						10	280		10	280
						10	290		10	290
						10	300		10	300
						10	310		10	310
						10	320		10	320
						10	330		10	330
						10	340		10	340
						10	350		10	350
						10	360		10	360

						10	370		10	370
						10	380		10	380
						0,1	390		10	390
						0,1	400		10	400
						0,1	410		10	410
						0,1	420		10	420
						0,1	430		10	430
						0,1	440		10	440
						0,1	450		10	450
						0,1	460		10	460
						0,1	470		10	470
						0,1	480		10	480
						0,1	490		10	490
						0,1	500		10	500
						0,1	510		10	510
						0,1	520		10	520
						0,1	530		10	530
						0,1	540		10	540
						0,1	550		10	550
						0,1	560		10	560
						0,1	570		10	570
						0,1	580		10	580
						10	590		10	590
						10	600		10	600
						10	610		10	610
						10	620		10	620
						10	630		10	630
						10	640		10	640
						10	650		10	650
						10	660		10	660
						10	670		10	670
						10	680		10	680
						10	690		10	690
						10	700		10	700
						10	710		10	710
						10	720		10	720
						10	730		10	730
						10	740		10	740
						10	750		10	750
						10	760		10	760
						10	770		10	770

						10	780		10	780
						10	790		10	790
						10	800		10	800
						10	810		10	810
						10	820		10	820
						10	830		10	830
						10	840		10	840
						10	850		10	850
						10	860		10	860
						10	870		10	870
						10	880		10	880
						10	890		10	890
						10	900		10	900
						10	910		10	910
						10	920		10	920
						10	930		10	930
						10	940		10	940
						10	950		10	950
						10	960		10	960
						10	970		10	970
						10	980		10	980
						10	990		10	990
						10	1000		10	1000
						10	1010		10	1010
						10	1020		10	1020
						10	1030		10	1030
						10	1040		10	1040
						10	1050		10	1050
						10	1060		10	1060
						10	1070		10	1070
						10	1080		10	1080
									10	1090
									10	1100
									10	1110
									10	1120
									10	1130
									10	1140
									10	1150
									10	1160
									10	1170
									10	1180

									10	1190
									10	1200
									10	1210
									10	1220
									10	1230
									10	1240
									10	1250
									10	1260
									10	1270
									10	1280
									10	1290
									10	1300
									10	1310
									10	1320
									10	1330
									10	1340
									10	1350
									10	1360
									10	1370
									10	1380
									10	1390
									10	1400