APLICACIÓN MÓVIL PARA LA MEDICIÓN DEL CONSUMO DE BATERÍA DE LOS SENSORES EN UN TELÉFONO INTELIGENTE

MÓNICA ALEJANDRA PARRA MURCIA
ERMES GUIDO CERÓN BOLAÑOS

UNIVERSIDAD ANTONIO NARIÑO
FACULTAD DE INGENIERÍA DE SISTEMAS
TRABAJO DE GRADO II
BOGOTÁ
2020

APLICACIÓN MÓVIL PARA LA MEDICIÓN DEL CONSUMO DE BATERÍA DE LOS SENSORES EN UN TELÉFONO INTELIGENTE

MÓNICA ALEJANDRA PARRA MURCIA
ERMES GUIDO CERÓN BOLAÑOS

TESIS PARA OPTAR POR EL TÍTULO DE INGENIERO DE SISTEMAS

DIRECTOR: Dr. JUAN CAMILO RAMÍREZ

UNIVERSIDAD ANTONIO NARIÑO
FACULTAD DE INGENIERÍA DE SISTEMAS
TRABAJO DE GRADO II
BOGOTÁ
2020

Nota de Aceptación

_____

_____

_____

_____

Presidente del Jurado

_____

Jurado

_____

Jurado

Bogotá, 02 de junio de 2020

“Dedico esta tesis a mi madre, a mi padre y a mi hermano, quienes han sido un gran apoyo en todo este proceso. Dedico especialmente esta tesis al Dr. O.R., un gran maestro al que admiro profundamente y quien ha contribuido en mi formación laboral, académica y personal.”

Mónica Alejandra Parra.

“Dedico esta tesis a mi familia, quienes han sido una parte importante de mi vida y siempre me han bridando su apoyo.”

Ermes Guido Cerón

AGRADECIMIENTOS

Los autores del presente trabajo agradecen al proyecto de investigación “Sistema de comunicación para sobrevivientes de un desastre basado en una red ad hoc de teléfonos inteligentes” de la Universidad Antonio Nariño por permitirnos la participación en el mismo. De igual manera a los docentes que colaboraron en la construcción de esta propuesta, cuyas sugerencias enriquecieron el presente documento.

Agradecemos también a los docentes que participaron como jurados, quienes con sus aportes contribuyeron a mejorar la versión final.

TABLA DE CONTENIDO

TABLA DE FIGURAS

Pág.

RESUMEN

Actualmente hay diversos sensores integrados en un teléfono inteligente, los más comunes son: el giroscopio, sensor de proximidad, acelerómetro y sensor de luminosidad. Se identificó que actualmente no hay ninguna aplicación para Android desde la versión 8.0 que permita monitorear el consumo de energía de los sensores nombrados. Esto puede ser útil, dado que a diversas aplicaciones eventualmente les puede interesar conocer el consumo de batería de algún sensor, siendo este un desarrollo independiente que puede integrarse posteriormente a otra aplicación que lo requiera.

Teniendo en cuenta que actualmente no se dispone de una aplicación que permita monitorear el nivel de batería consumida por los sensores de un teléfono inteligente, en este trabajo se desarrolló una aplicación capaz de monitorear el consumo de los principales sensores sobre la batería en un teléfono inteligente que funcione con Android versión 8.0 y versiones posteriores. Los sensores que se someten a este monitoreo son los más comunes en los sistemas Android, como el giroscopio, sensor de proximidad, acelerómetro y sensor de luminosidad.

La aplicación móvil propuesta en este trabajo de grado servirá de apoyo a desarrolladores de otras aplicaciones para teléfonos inteligentes que estén interesados en estimar el consumo de energía de las mismas al utilizar los sensores del dispositivo. En particular, servirá de apoyo para el proyecto de investigación ‘Sistema de comunicación para sobrevivientes de un desastre basado en una red ad hoc de teléfonos inteligentes’ de la Universidad Antonio Nariño, donde se necesita una aplicación para medir el consumo de energía de los sensores.

INTRODUCCIÓN

Los teléfonos inteligentes se han convertido en una herramienta fundamental, dado su evolución en el desarrollo de operaciones como el aumento en sus funcionalidades y los potentes sistemas de software con los que trabajan, aun así, esto genera un consumo importante de batería. Este gran consumo de batería es un aspecto fundamental a tener en cuenta, ya que monitorear el nivel de batería ayudaría al usuario a gestionar de mejor manera el dispositivo.

En el marco del proyecto de investigación “Sistema de comunicación para sobrevivientes de un desastre basado en una red ad hoc de teléfonos inteligentes”, se está desarrollando una aplicación móvil de rescate de personas en caso de desastres naturales utilizando los celulares como nodos mediante redes ad hoc cuando no funcionan los sistemas de comunicación, esta aplicación puede hacer uso de algunos de los sensores del teléfono inteligente para localizar a las víctimas y establecer comunicación con ellos.

Al no ser posible la detección y visualización del consumo de los sensores sobre la batería, será difícil para el usuario gestionar la duración de la misma. Si en una situación de riesgo el usuario no conoce cuánta batería está consumiendo un sensor, puede ser crítico debido a que se necesita optimizar al máximo el rendimiento del dispositivo.

Actualmente no se cuenta con una aplicación móvil capaz de detectar la cantidad de energía que consume cada uno de los sensores integrados en un teléfono inteligente. En caso de una situación de desastre es fundamental mantenerse en conexión y para ello es necesario maximizar la duración de la batería.

En este proyecto se desarrolló una aplicación móvil capaz de medir la cantidad de energía que consumen algunos sensores de un celular, la cual será útil para aplicaciones que necesiten monitorear uno o varios sensores en particular y aportará también al proyecto de investigación donde se desarrolló la aplicación de mensajería en desastres naturales. Este proyecto se trabajó con la metodología ágil Scrum, manejando el desarrollo en etapas iterativas, que permiten hacer pruebas y mejoras en cada uno de los procesos. La aplicación está disponible para celulares que funcionen con sistema operativo Android desde la versión 8.0 (Oreo) en adelante, no se garantiza compatibilidad con versiones inferiores.

1. PLANTEAMIENTO DEL PROBLEMA

1.1. DESCRIPCIÓN DEL PROBLEMA

Los teléfonos inteligentes incluyen sensores que optimizan el funcionamiento de determinadas aplicaciones, los sensores comúnmente incluidos son el giroscopio, de proximidad, acelerómetro y de luminosidad. Actualmente, no hay disponible una aplicación capaz de medir el nivel de energía que consume cada uno de estos sensores, aspecto importante si se quiere prolongar la duración de la batería.

En una situación de desastres, los sistemas de comunicación generalmente colapsan, impidiendo que una persona que se encuentre atrapada bajo escombros pueda comunicarse y pedir ayuda. Por esta razón, en el proyecto de investigación “Sistema de comunicación para sobrevivientes de un desastre basado en una red ad hoc de teléfonos inteligentes” de la Universidad Antonio Nariño, se desarrolló una aplicación móvil que utiliza los celulares como nodos de redes ad hoc, en caso de que no funcionen los sistemas de comunicación convencionales que permitirían el rescate de personas en caso de desastres naturales, esta aplicación puede utilizar algunos sensores del teléfono inteligente para llevar a cabo su funcionalidad. Sin embargo, la aplicación no dispone de un monitoreo que permita visualizar la cantidad de energía que consumen los sensores integrados en un celular, lo cual es necesario para ayudar a maximizar la vida útil de la batería.

Los sensores son parte del hardware de un dispositivo Android que les permite capturar información del exterior para usarla a su favor. Para este proceso se requiere el uso de energía, la cual es suministrada por una batería de 3.7 voltios que viene integrada en todos los teléfonos celulares actuales (Android, 2019). El consumo de batería es proporcional al trabajo que realizan los sensores, la capacidad de carga de estas baterías oscila entre los 2400 mAh hasta los 6000 mAh (miliamperios por hora), según la gama del teléfono. Esta cantidad se debe dividir entre el consumo de la batería actual (mA)(miliamperios) y su resultado será las horas de batería restantes antes de descargarse (Android, 2019).

Los sensores están activos y recopilando información todo el tiempo, por lo que monitorearlos es una tarea de gran importancia, así con esto se pueden establecer estimaciones relacionadas con el consumo de batería actual y su duración. Un dispositivo Android tiene una determinada cantidad de detectores de acuerdo a su gama (Android, 2019). No obstante, hay cuatro que son comunes para casi todos los dispositivos actuales, por lo que es útil centrarse en estos cuatro detectores. Los cuatro detectores que comúnmente están integrados a los teléfonos inteligentes son: giroscopio, sensor de proximidad, acelerómetro y sensor de luminosidad.

Debido a la falta de un desarrollo que permita monitorear el rendimiento de la batería en la aplicación desarrollada en el marco del proyecto de investigación “Sistema de comunicación para sobrevivientes de un desastre basado en una red ad hoc de teléfonos inteligentes”, en el presente proyecto se desarrolló una aplicación que permite monitorear el consumo de energía de los cuatro detectores mencionados. Se tiene previsto que esta aplicación pueda ser utilizada por otras aplicaciones que requieran medir la energía que consume uno o varios sensores de los mencionados en este proyecto.

1.2. FORMULACIÓN DEL PROBLEMA

El software para monitoreo del consumo de energía de los sensores es un desarrollo requerido por el proyecto de investigación “Sistema de comunicación para sobrevivientes de un desastre basado en una red ad hoc de teléfonos inteligentes” de la Universidad Antonio Nariño dado que actualmente no cuentan una herramienta para tal fin. El grupo que integra el proyecto mencionado, ha desarrollado una aplicación que pueda ser de ayuda para personas que resulten ser víctimas de un desastre natural como los terremotos y en la que contar con una herramienta que permita el monitoreo del consumo de energía de los sensores resulta vital.

Desde la ingeniería se propuso desarrollar un sistema capaz de monitorear el consumo de energía de los sensores giroscopio, de proximidad, acelerómetro y de luminosidad, incluidos en un teléfono inteligente que funcione con sistema operativo Android desde la versión 8.0 (Oreo) en adelante, aplicando la metodología ágil Scrum.

1.3 JUSTIFICACIÓN DEL PROBLEMA

El enfoque principal del presente proyecto es el monitoreo del consumo de energía de los sensores de un teléfono inteligente. Desarrollo que es útil debido a que no existe este módulo en la aplicación implementada en el marco del proyecto de investigación de una aplicación para situaciones de desastre. Fue necesario crear una aplicación que registre el consumo de energía de determinados sensores puesto que, en una situación de desastre, maximizar la duración de la batería puede ser crucial para mantener la comunicación y proporcionar los datos necesarios para ser ubicado y rescatado. El proceso de rescate de un superviviente en una situación de desastre puede tardar horas, así mismo, con el paso del tiempo las necesidades de la víctima van en aumento. El tener un sistema de comunicación en una situación de desastre permite a los rescatistas estar al tanto del estado del superviviente y tomar las medidas necesarias para el momento de su rescate. El aporte de la aplicación es poder monitorizar el consumo de energía de los sensores más comunes de un teléfono inteligente.

La utilidad se verá reflejada en el estado de la batería, aspecto a tener en cuenta cuando se requiere prologar su duración. En situaciones de desastre como los terremotos, las personas pueden quedar atrapadas bajo escombros, si esto ocurre sus opciones serán limitadas, dado que, no podrán cargar su teléfono ya sea porque no se puedan mover o porque no hay energía. Por esto, las posibilidades de comunicarse dependen de la carga de batería disponible en el momento del desastre.

La investigación que se llevó a cabo para realizar este desarrollo puede ser de mucha ayuda para futuros proyectos que requieran una implementación similar que enriquecerá cada vez más el conocimiento, tanto para la comunidad de ingeniería como la sociedad en general. Un desarrollo como este podrá ser usado también por otro tipo de aplicaciones a las que les sea útil realizar la medición del nivel de consumo de energía de los sensores de un dispositivo móvil.

La investigación que requiere el proyecto para su desarrollo enriquece a sus investigadores y permite generar nuevo conocimiento, que puede ser aprovechado por otros ingenieros que necesiten la información tanto de la investigación como del desarrollo. Esto representa una gran oportunidad de adquirir experiencia laboral cercana a la que se tendría en un ambiente laboral común de desarrollo, por lo que la

investigación y desarrollo de la aplicación es una preparación para aplicar en cualquier trabajo de desarrollo que pueda llevarse a cabo posteriormente en el ejercicio de la labor profesional.

En razón a que el software está basado en licencias gratuitas e investigación universitaria, no generará ingresos monetarios a los desarrolladores. Sin embargo, su implementación podría salvar vidas. Dado que la aplicación de rescate desarrollada por el grupo de investigación permite la comunicación para poder ubicar y rescatar a una persona de forma precisa, se disminuiría los gastos de recursos de labores de búsqueda aleatorias, perdiendo tiempo y posiblemente más vidas. Igualmente, la aplicación desarrollada en este trabajo puede ser destinada a complementar otras aplicaciones que demanden monitorear el consumo de energía de determinados sensores.

Si esta problemática no fuera resuelta, los usuarios de la aplicación de rescate verían reducidas sus posibilidades de tener una comunicación eficiente. Sin información que los alerte del estado de la batería, los usuarios tienden a excederse en el uso de su teléfono. Además, sin estimaciones que les informen cuántas horas de batería tienen disponible, no tomarían precauciones para minimizar el consumo. Algunos sensores como el acelerómetro tienden a consumir mucha más batería cuando están en movimiento, por lo que dejar el celular quieto sería una forma de ahorrar batería y eso se vería reflejado en la pantalla de monitoreo. El no contar con una herramienta que permita conocer el consumo de los sensores de un celular podría reducir sus oportunidades de comunicarse y ser rescatado.

Se propuso desarrollar un sistema capaz de monitorear el consumo de energía de los sensores, debido a que actualmente no se dispone de una aplicación que realice esta labor en el sistema operativo Android actual. Al realizar este software se ampliará la disponibilidad para las versiones de Android 8.0 (Oreo) en adelante. Se espera que este desarrollo ayude en futuros proyectos de ingeniería que requieran una implementación similar. Con el sistema desarrollado, el usuario tendrá acceso a información de consumo de energía en tiempo real y con esto puede hacer una mejor gestión de su dispositivo como apagarlo por un intervalo de tiempo para ahorrar batería.

1.4 OBJETIVOS

1.4.1 Objetivo general

Desarrollar una aplicación móvil que permita medir el consumo de los sensores giroscopio, de proximidad, acelerómetro y de luminosidad, incluidos en un teléfono inteligente que funcione con sistema operativo Android desde la versión 8.0 (Oreo) en adelante, aplicando la metodología ágil Scrum.

1.4.2 Objetivos específicos

1. Definir los requerimientos funcionales y no funcionales de la aplicación para conocer el alcance que va a tener la aplicación a través de una plantilla de requerimientos.

2. Diseñar la aplicación para conocer las limitaciones de la misma utilizando diagramas UML.

3. Implementar la aplicación por medio del IDE Android Studio en su última versión, para generar un producto que cumpla con el diseño y la especificación de requerimientos realizada.

4. Validar la funcionalidad de la aplicación a través de pruebas sobre el desarrollo realizado en reuniones con los directores del proyecto

1.5 ALCANCE Y LIMITACIONES DEL PROYECTO

1.5.1 Alcance

- Desarrollar una aplicación para teléfonos inteligentes que funcionen con Android, en la cual sea posible monitorear el consumo de energía en tiempo real de los sensores el giroscopio, de proximidad, acelerómetro y de luminosidad.

- La aplicación se implementará en herramientas para desarrollo como Android Studio en su última versión, en conjunto con las librerías necesarias para su implementación.

- La aplicación a desarrollar será para dispositivos Android exclusivamente.

- La finalidad de la aplicación a desarrollar es brindar estimaciones acerca del consumo de energía de los principales detectores que tiene el teléfono inteligente.

- Al elegir los sensores, se tuvo en cuenta la presencia de los mismos en la mayoría de los teléfonos inteligentes actuales, por lo que se determinó que los sensores a monitorear son: el giroscopio, de proximidad, acelerómetro y de luminosidad.

- Este desarrollo podrá servir de base para cualquier otra aplicación Android que requiera monitorear alguno de los cuatro sensores mencionados.

- Dentro del desarrollo de la aplicación en el framework Android Studio se plantea utilizar un conjunto de librerías que soportarán la implementación a realizar. Las principales clases que se usarán son BatteryManager, Sensor, SensorEvent y SensorManager y la interfaz SensorEventListener.

1.5.2 Limitaciones

- El software será exclusivamente para el sistema operativo móvil Android versión 8.0 (Oreo) en adelante, no se garantiza el funcionamiento con versiones previas.

- El lenguaje de programación será Java el cual se implementará mediante el IDE Android Studio en su versión más actual.

- No se incluye compatibilidad con tecnologías ajenas a Android.

- Las fases de implementación usarán el modelo ágil Scrum.

- Aunque la aplicación a desarrollar podría ser compatible con cualquier dispositivo móvil con versión Android 8.0 en adelante (Tablets, Relojes), solo se garantiza su funcionamiento en teléfonos inteligentes (Smartphones).

2. METODOLOGÍA

Se eligió para el desarrollo de la aplicación el método de “Scrum”. Este método se ha elegido para este proyecto debido a que es una metodología ágil y se realizan ciclos iterativos (Sprints) en el desarrollo.

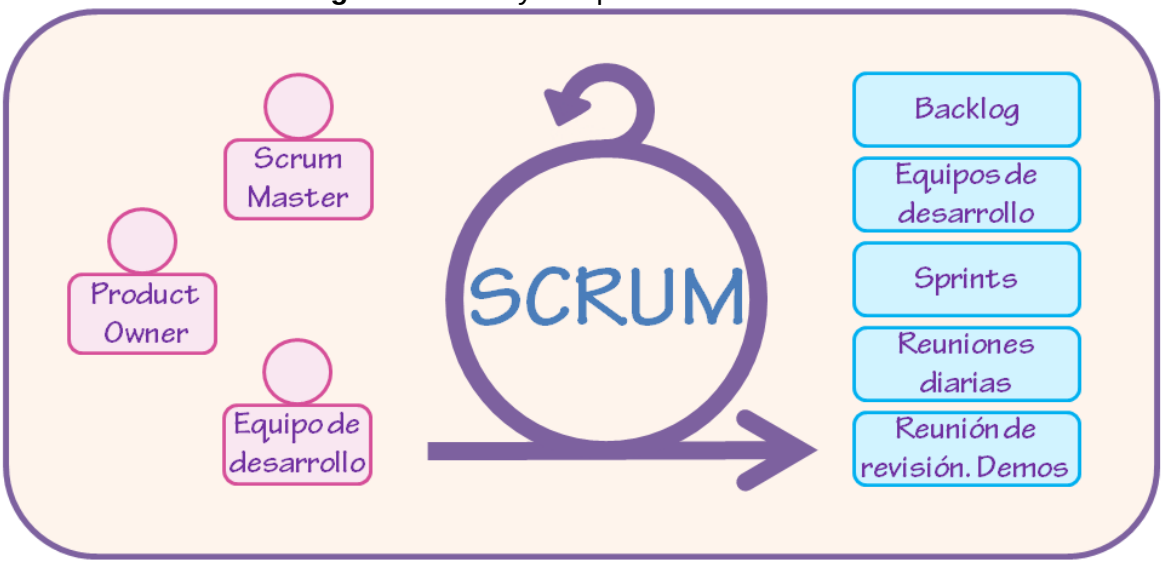
2.1 DEFINICIÓN

Según Laínez (2015), esta metodología permite desarrollar software de manera incremental, es decir en ciclos (Sprints) que permiten tener una mayor claridad para el desarrollo. Es ampliamente útil en un proyecto donde los requerimientos pueden cambiar a menudo, como fue el caso de este proyecto, donde al trabajar dentro un grupo de investigación, el desarrollo estuvo sujeto a las necesidades del mismo.

Actualmente es una de las metodologías más utilizadas debido a que se ajusta a las nuevas formas de gestión en las empresas y está enfocada a la productividad, resaltando la importancia de la comunicación y el trabajo en equipo, además de permitir entregas de software funcional de forma incremental, disminución de errores y acceso a la información en todo el proceso.

En la Figura 1 se ilustran los roles y componentes de la metodología ágil Scrum.

Figura 1. Roles y componentes de Scrum.



Fuente: elaboración propia

2.2 ROLES

En la metodología Scrum el personal se clasifica en dos grupos (Trigas, 2012):

A. Personas que intervienen directamente con el proyecto y con el proceso Scrum:

2.2.1 Product Owner

Es el encargado de la toma de decisiones, conoce con claridad el negocio del cliente y lo que espera del producto, escribe las ideas del cliente en el backlog según su prioridad.

2.2.2 Scrum Master

Es el encargado de verificar que tanto el modelo como la metodología funcionen, comunicarse con el cliente y revisar los problemas que se presenten en el proceso.

2.2.3 Equipo de desarrollo

Pequeño grupo de personas que pueden tomar decisiones para cumplir con los objetivos planteados.

B. Personas que no intervienen directamente con el proyecto pero son necesarias para los procesos de retroalimentación y las revisiones y planeaciones de los Sprints:

2.2.4 Usuarios

Es el destinatario final a quien va dirigido el software

2.2.5 Stakeholders

Son las personas que se beneficiarán del proyecto, intervienen en las revisiones del sprint.

2.2.6 Managers

Es el encargado de tomar las decisiones finales, se involucra en la declaración de objetivos y requerimientos.

2.3 COMPONENTES

En la metodología Scrum se trabaja con los siguientes componentes (Laínez, 2015):

2.3.1 Backlog

Es una lista con el trabajo pendiente. Se forma con las ideas que van surgiendo para la implementación. Se decidirá cuáles serán los objetivos y el trabajo a realizar para esa iteración.

2.3.2 Equipos de desarrollo

Son pequeños grupos de trabajo en equipo, donde la responsabilidad de sus éxitos y fracasos es de todo el equipo, generalmente no se dirigen a otros sectores cuando no existe una necesidad real.

2.3.3 Sprints

Es el periodo de tiempo en el cual se realizarán las tareas establecidas en el backlog, que generalmente se conforma de 15 días. Al finalizar el periodo de tiempo se efectúan reuniones para asignar nuevas tareas a ejecutarse, definiendo el tiempo necesario para su desarrollo que depende esencialmente de la prioridad y complejidad de cada tarea.

2.3.4 Reuniones diarias

Son encuentros que se realizan cada día, donde se habla de que se hizo el día anterior, que se hará el día actual, que problemas se han encontrado y el por qué no se han podido resolver. La finalidad de estas reuniones es corregir problemas y evitar atrasos en el proyecto, generalmente tienen una duración de 15 minutos.

2.3.5 Reuniones de revisiones y presentación de demos

Son encuentros que se realizan cuando el periodo de desarrollo se concluye. Se hacen revisiones del sprint que incluyen pruebas y demostraciones, así como reflexiones sobre los errores y mejoras.

2.4 APLICACIÓN DE LA METODOLOGÍA

Para el desarrollo de este trabajo se definieron los siguientes aspectos:

2.4.1 Roles asignados

- Product Owner: María del Pilar Salamanca, directora del proyecto de investigación
- Scrum Master: Juan Camilo Ramírez, director del trabajo de grado.
- Equipo de desarrollo: Ermes Guido Cerón, Mónica Alejandra Parra, estudiantes de Ingeniería de Sistemas

2.4.2 Definición del Sprint

Se realizó una reunión cada dos semanas de 60 minutos de duración con el Scrum Master, en donde se hacía una retroalimentación, se presentaban los demos de la aplicación y las correcciones y proyecciones para el siguiente Sprint. Se realizaron en total cinco Sprints.

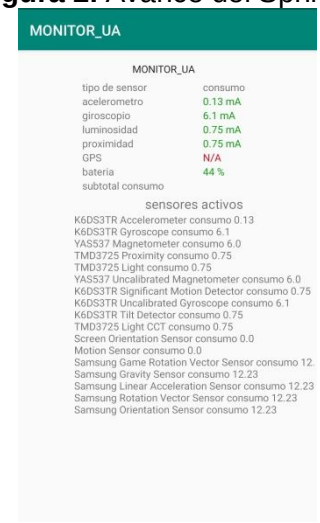
Adicionalmente, se realizó una reunión cada mes con el equipo del proyecto de investigación, en donde se hicieron revisiones de avances, proyecciones, presentaciones de demos y retroalimentaciones. Se realizaron en total tres sesiones.

2.4.3 Sprint 1

Avances

- Se presentó un demo de la aplicación que mostraba en cajas de texto la corriente de los sensores presentes en el teléfono inteligente (Figura 2).

Figura 2. Avance del Sprint 1



Proyecciones

- El Scrum Master planteó presentar los datos de potencia en Miliwatts (Mw) de los cuatro sensores.

2.4.4 Sprint 2

Avances

- Se presentó un demo de la aplicación que mostraba en cajas de texto la potencia en Miliwatts (Mw) de los sensores activos en el teléfono inteligente (Figura 3).

Figura 3. Avance del Sprint 2

Proyecciones

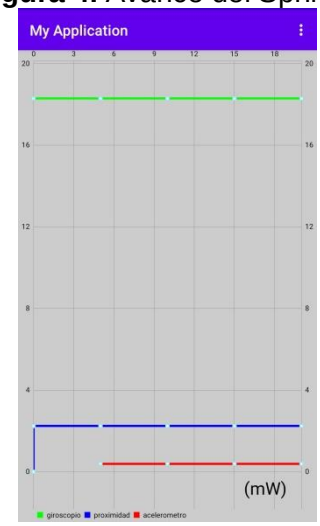
- Se definió presentar los datos en una graficadora.

2.4.5 Sprint 3

Avances

- Se presentó un demo de la aplicación que mostraba la potencia de los sensores en Miliwatts (Mw), en una gráfica de tipo plano cartesiano (ejes *x* y *y*). En la graficadora se mostraba el monitoreo durante 20 segundos y se detenía (Figura 4).

Figura 4. Avance del Sprint 3



Proyecciones

- Se definió que la graficadora debía mostrar el monitoreo durante el tiempo que la aplicación esté activa.

- Se definió que los datos que refleja la graficadora deben mostrarse de forma entendible al usuario

- Se estableció que en la graficadora se deben mostrar los rótulos en los ejes *x* y *y*.

- Incluir la funcionalidad de Capturar datos, incluyendo un archivo donde se guarden los datos de las lecturas

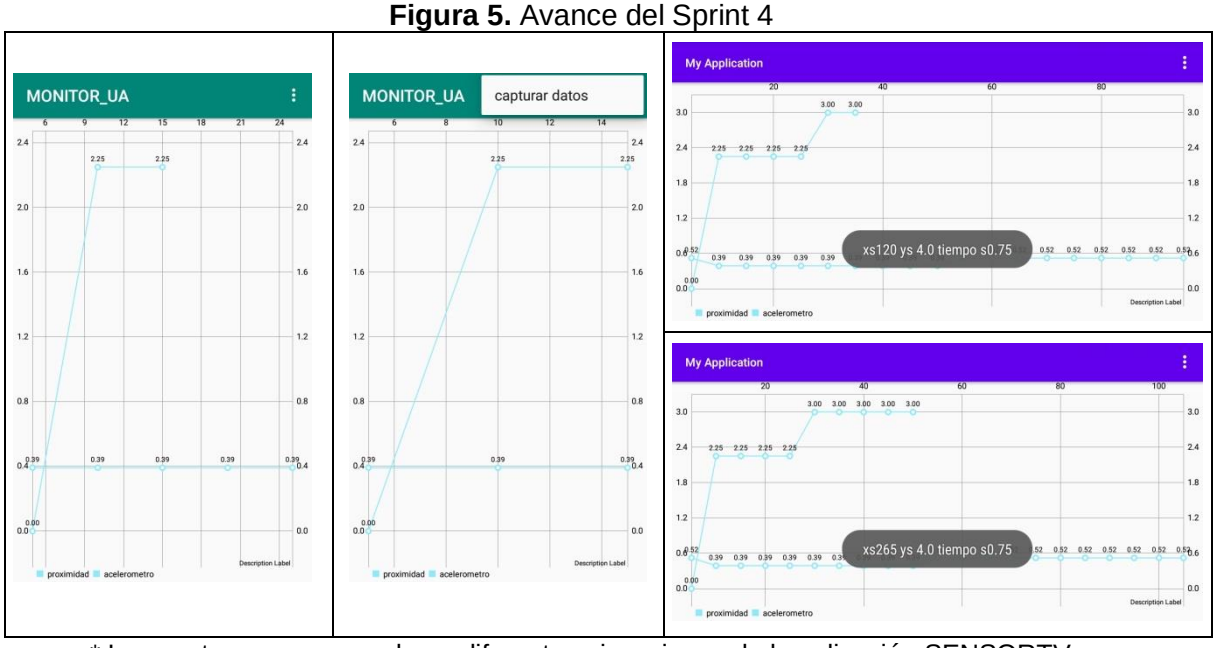
2.4.6 Sprint 4

Avances (Figura 5)

- Se presentó un demo de la aplicación que mostraba los datos de la potencia en periodos de 5 segundos. La aplicación mostraba una gráfica en donde los números expresaban solo un punto decimal para mejorar la estética. Se agregaron los rótulos en sus ejes.

- En la aplicación se incluyó una funcionalidad para guardar los datos que monitoreaba dado un tiempo, estos datos se guardaban en un archivo tipo csv.

Figura 5. Avance del Sprint 4



* Las capturas corresponden a diferentes ejecuciones de la aplicación SENSORTV

Proyecciones

- Se estableció definir una latencia de 3 segundos

- Se sugirió mejorar el archivo que guarda las lecturas, mostrando las capturas individuales por cada sensor además de mostrar información de totales con la sumatoria de la potencia de cada sensor.

- Los datos guardados en el archivo se deben presentar en Miliwatts-minuto.

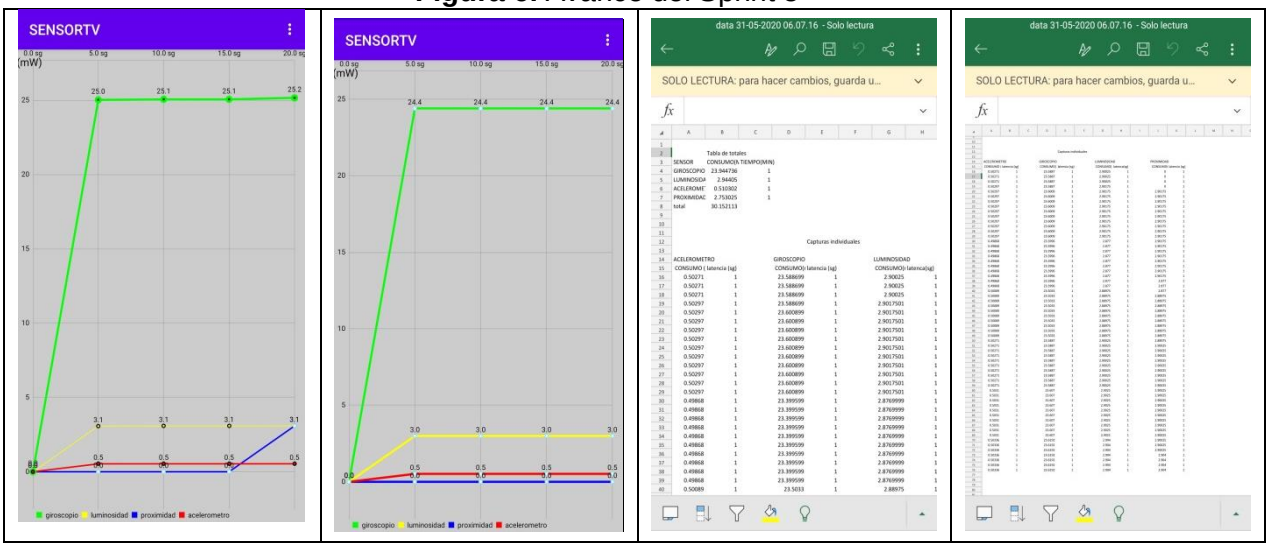
2.4.7 Sprint 5

Avances (Figura 6)

- Se presentó un demo de la aplicación que mostraba los datos de la potencia en periodos de 3 segundos para obtener más precisión en tiempo real.

- Se presentó un archivo de tipo csv que mostraba las capturas individuales de cada sensor con una latencia de un segundo y adicional un informe con los totales del consumo de los sensores en el tiempo seleccionado (Miliwatts-minuto).

Figura 6. Avance del Sprint 5



*Las capturas corresponden a diferentes ejecuciones de la aplicación SENSORTV

Proyecciones

- Se estableció mejorar la estética de la aplicación

3. MARCO DE REFERENCIA

3.1 MARCO TEÓRICO

3.1.1 Teléfono inteligente

Según Carroll & Heiser (2010) y Wong (2010) en Khan *et al*. (2016), un teléfono inteligente es un teléfono móvil moderno de gama alta con capacidades similares a las de un computador. Tiene incorporado un hardware y sistemas operativos que permiten ejecutar aplicaciones sofisticadas y procesar un gran volumen de datos. En este caso, el teléfono es la herramienta que soporta el software de monitoreo de los sensores que se está desarrollando.

3.1.2 Sensores de uso general

Se considera sensor a todo lo que tenga propiedades sensibles a una magnitud del medio y sea capaz de manifestar su presencia y medida. Elemento que puede cambiar una propiedad ante magnitudes físicas o químicas (denominadas variables de instrumentación) y transformarlas en variables eléctricas. Algunas de las principales variables de instrumentación son: la intensidad lumínica, la temperatura, la humedad, la distancia, la aceleración, la inclinación y el desplazamiento (Carletti, online).

Algunos sensores consumen menos y otros consumen mayor energía de la batería. En los teléfonos inteligentes el tamaño y la capacidad de la batería están restringidos al tamaño y peso del dispositivo (Khan *et al*., 2016).

Los sensores sobre los que se va a trabajar en la monitorización son los más comunes y serán el objeto sobre el que gira el presente desarrollo, se mencionan a continuación:

- Giroscopio: monitorea la posición tridimensional x,y,z del teléfono, es decir su ubicación actual (*Rubio,* 2015). Esto permite girar la pantalla y modificar su contenido para que se pueda observar normalmente. Habitualmente es vista en horizontal y vertical, donde la horizontal distribuye el contenido de la pantalla horizontalmente.

- Sensor de proximidad: detecta objetos cercanos al dispositivo (*Jiménez*, (2016). Esta utilidad se ve reflejada al momento de contestar una llamada ya que este es el encargado de apagar la pantalla cuando el teléfono es acercado a la oreja del usuario, esto para evitar que se hagan manipulaciones por error durante una llamada.

- Acelerómetro: mide la aceleración del teléfono respecto a la fuerza de gravedad (*Jiménez*, 2016), necesario para ayudar al giroscopio a girar la pantalla del teléfono y determinar el ángulo y orientación. También se usa para determinar la velocidad o las calorías consumidas.

- Sensor de luminosidad: según *Jiménez*, (2016), permite ajustar el brillo de la pantalla automáticamente de acuerdo al nivel de luz de su entorno. Entre más luz haya, menos brillo tendrá el dispositivo o al contrario.

3.1.3 Application Programming Interface – API

Se entiende por API (interfaz de programación de aplicaciones) a las librerías de programación con el conjunto de procedimientos y funciones que ofrecen. Según Boillot (2012), una API proporciona información de coordenadas y movimiento de un objeto dentro de un campo sensorial, proporciona métodos para detectar una velocidad, desplazamiento, aceleración, o un periodo. También, contiene eventos que permiten detectar actividad de los sensores y un controlador para procesar los eventos capturados. Conjunto de protocolos que se comunican entre sí para solucionar un determinado problema, en este caso es el monitoreo de los sensores de un teléfono inteligente.

Para el desarrollo de la aplicación se trabajará con las siguientes herramientas:

- BatteryManager: clase necesaria para acceder al estado de la batería de un teléfono inteligente (Android, 2019). Esta clase recopila la información de la batería para establecer el tiempo de funcionalidad de esta antes de agotarse.

- Sensor: clase que se utiliza para crear la instancia de un sensor específico.

- SensorEvent: clase que utiliza el sistema para publicar datos del sensor. Incluye los valores de datos de sensores sin procesar, el tipo de sensor, la precisión de los datos y una marca de hora.

- SensorEventListener: tipo interfaz que proporciona métodos de llamada de regreso para recibir avisos del SensorManager cuando los datos o la precisión del sensor han cambiado. Esta interfaz permite la comunicación de la aplicación con los sensores para poder extraer su información y presentarla en el monitor.

- SensorManager: clase que se utiliza para crear una instancia del servicio de sensores; proporciona varios métodos para el acceso a sensores, el registro y la eliminación de registros de las escuchas de eventos de sensores. Esta clase contiene todos los eventos que requiere la manipulación de un sensor de un teléfono inteligente, como es la clase Sensor, SensorEvent, SensorEventLitener, los cuales permiten interactuar con los sensores (Miao, 2019).

3.1.4 Desastre natural

Romero & Maskrey (1983) definen desastre natural como el hecho que ocurran fenómenos naturales peligrosos como: los terremotos, huracanes y maremotos, en ciertas condiciones vulnerables del tipo socioeconómicas y físicas como una economía precaria en la población, viviendas construidas inadecuadamente en ubicaciones riesgosas y condiciones de suelo poco estable.

Adicionalmente, para implementar las funcionalidades de la aplicación y el monitoreo de los sensores acelerómetro, de proximidad, giroscopio y de luminosidad, se aplicaron algunas fórmulas de las leyes mencionadas a continuación:

3.1.5 Ley de Ohm

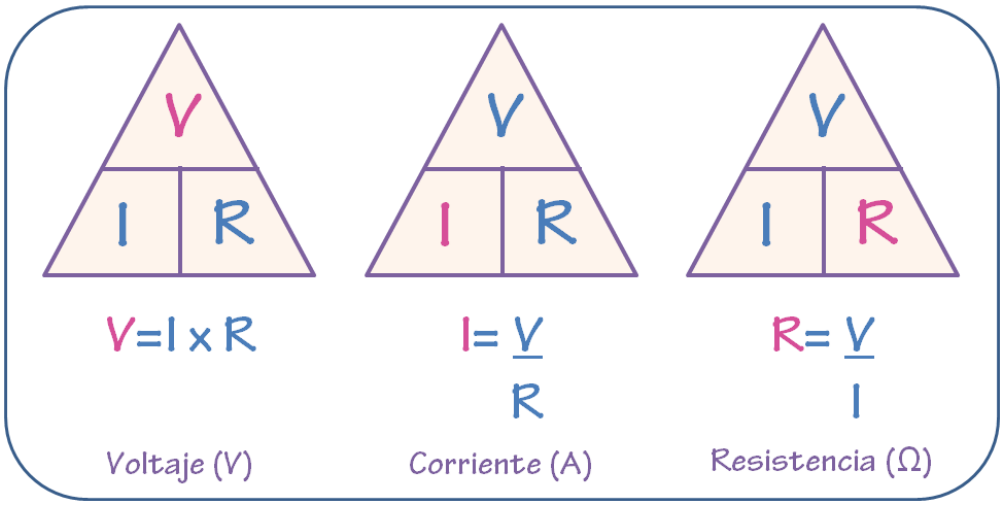
La ley de Ohm (Serway & Jewett, 2009) afirma que en muchos materiales incluyendo metales, existe una relación de densidad de corriente al campo eléctrico, la cual es una constante independiente del campo eléctrico. Su fórmula general es:

V = R ˣ I, donde:

V representa el voltaje en Voltios (V)
R representa la resistencia en Ohmios (Ω)
I representa la intensidad de corriente en Amperios (A)

Así mismo, se tiene el diagrama de la ley de Ohm (conocido como triángulo de la ley de Ohm), el cual ilustra tres maneras en que se pueden relacionar estas magnitudes físicas, como se observa en la Figura 7.

Figura 7. Triángulo de la ley de Ohm



Fuente: elaboración propia con base en Gouveia (online)

3.1.6 Ley de Watt

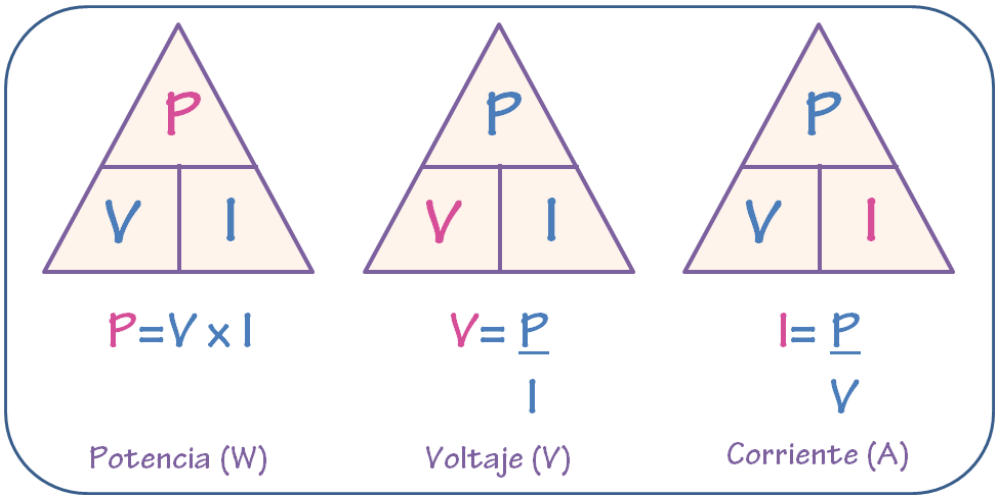
La ley de Watt (Zetina-M. & Zetina-C., 2004) afirma que la potencia de un circuito es directamente proporcional al producto del voltaje y la intensidad de corriente que circula por el circuito, esto indica que entre más alto sea el voltaje aplicado o más intensa sea la corriente, mayor será la potencia eléctrica. Su fórmula general es:

P = V ˣ I, donde:

P representa la potencia eléctrica en Watts (W)
V representa el voltaje en Voltios (V)
I representa la intensidad de corriente en Amperios (A)

De igual forma, se tiene el triángulo de la ley de Watt, el cual muestra la forma de despejar las ecuaciones para interactuar con estas magnitudes físicas, como se observa en la Figura 8.

Figura 8. Triángulo de la ley de Watt



Fuente: elaboración propia con base en Gouveia (online)

3.2 ANTECEDENTES O ESTADO DEL ARTE

El consumo de batería es un aspecto fundamental cuando se desarrollan aplicaciones móviles y puede convertirse en un grave problema a combatir debido a que los dispositivos modernos consumen más batería y se pueden presentar fugas espontáneas de energía (Khan *et al*., 2016).

En el año 2010 se desarrolló el sistema de estimación de energía “PowerTutor” (Zhang *et al*., 2010), el cual permitía realizar estimaciones en tiempo real para algunos componentes del hardware, entre ellos el GPS, el Wi-Fi, el audio e interfaces celulares. Este sistema se implementó para teléfonos inteligentes funcionales con sistema operativo Android.

Khan *et al*. (2016) consideraron importante evaluar los sensores en los teléfonos inteligentes que monitorean actividades cotidianas y su impacto en la vida útil de la batería. Con este fin, desarrollaron la aplicación “EM3S” para teléfonos inteligentes que operaban con Android Ice Cream Sandwich (4.0.3), con el propósito de estimar el consumo de energía de los sensores sobre la batería, encontrando que los sensores consumen niveles variables de batería dependiendo de las actividades cotidianas del usuario.

Los teléfonos inteligentes actuales están equipados de robustos sistemas y requisitos de autonomía, lo cual ocasiona un mayor consumo de batería. Corral *et al*. (2013) desarrollaron la aplicación “CharM” capaz de analizar y caracterizar el ciclo de descarga. Esta aplicación de Android registra el nivel de batería y captura datos del sistema operativo para analizar las funcionalidades con mayor impacto en la curva de descarga de la batería.

Tawalbeh *et al*. (2016) realizaron una comparación de consumo de batería de los componentes OLED, CPU, WiFi y las unidades GPS, en dos teléfonos inteligentes (marcas Samsung y Sony) que funcionan con el sistema Android. Las mediciones las realizaron utilizando las aplicaciones “PowerTutor” y “AmobiSense" y concluyeron que el componente que mayor energía consume es el WiFi, los componentes 3G y la OLED también consumen una cantidad importante de batería.

Tuysuz *et al*. (2019) estimaron como componente principal a la tarjeta de interfaz de red inalámbrica considerando que tiene un consumo significativo de energía. Con base en esto, desarrollaron una aplicación para teléfonos inteligentes que funcionen con sistema Android denominada “PowerProfiler”, la cual opera como generador de perfiles de consumo de energía de red en tiempo real. Esta aplicación advierte el nivel de consumo de batería de interfaces de red inalámbrica y celular mediante la medición de paquetes reales y cálculos precisos, permitiendo al usuario gestionar varias operaciones mediante la aplicación.

“GreenDroid” es una aplicación concebida para examinar el consumo de batería de aplicaciones Android y descubrir si hay fugas de energía en el código fuente (Couto *et al*., 2015). Esta aplicación opera estimando el nivel de energía que consume determinada aplicación al ejecutarse, instrumentando el código fuente e ilustrando a manera de representaciones el análisis ejecutado.

En la Tabla 1 se consolida la información asociada a las características de cada una de las aplicaciones mencionadas anteriormente.

Tabla 1. Información general de aplicaciones mencionadas en antecedentes.

Aplicación	Componentes que aplica	Funcionalidad	Dispositivo	Sistema operativo*	Fuente
PowerTutor	GPS, el Wi-Fi, el audio e interfaces celulares	Realiza estimaciones en tiempo real	Teléfonos inteligentes (probado en HTC Dream y HTC Magic)	Android	Zhang *et al*. (2010)
CharM	CPU, OLED, Wi-Fi, GPS, reproducción de video, modo normal y modo avión	Analiza las funcionalidades con mayor impacto en la curva de descarga de la batería	Teléfonos y tabletas	Android	Corral *et al*. (2013)
GreenDroid	Aplicaciones Android	Estima el nivel de energía que consume determinada aplicación al ejecutarse y descubre si hay fugas de energía en el código fuente	Teléfonos inteligentes	Android	Couto *et al*. (2015)
PowerTutor AmobiSense	OLED, CPU, WiFi y las unidades GPS	Comparación de consumo de batería en dos teléfonos inteligentes con sistema Android	Galaxy Note3 y Sony Xperia Z2	Android	Tawalbeh *et al*. (2016)
EM3S	Sensores	Estima el consumo de energía de los sensores	Teléfonos inteligentes (probado en QMobile A12)	Android (Ice Cream Sandwich 4.0.3)	Khan *et al*. (2016)
PowerProfiler	Tarjeta de interfaz de red inalámbrica (WNIC)	Genera perfiles de consumo de energía de red en tiempo real	Estaciones inalámbricas o celulares IEEE 802.11	Android	Tuysuz *et al*. (2019)

Fuente: elaboración propia

*El sistema operativo de las aplicaciones se refiere a la versión que estaba vigente en el año de la publicación.

Como se ilustra en la Tabla 1, actualmente no se dispone con una aplicación que monitoree el nivel de energía de los sensores: giroscopio, de proximidad, acelerómetro y de luminosidad, en la versión de Android que se encuentra vigente a la fecha. Con la propuesta que se plantea en este trabajo, se pretende desarrollar una aplicación con la cual sea posible monitorear el consumo de energía de los sensores mencionados anteriormente, en teléfonos inteligentes compatibles con el sistema operativo Android actual (desde la versión 8.0 en adelante) y genere estimaciones para que el usuario conozca este consumo y pueda administrar de mejor manera su dispositivo.

3.3 MARCO LEGAL

3.3.1 Leyes de derechos de autor

Estas leyes tratan sobre los derechos morales y patrimoniales que se conceden a los autores por la creación de obras (inédita o publicada) de aspecto literario, artístico, musical, científica y didáctica. Los derechos de autor se encuentran regulados en Colombia por la Ley 23 de 1982 (Congreso de la República, Ley 23 de 1982), la Decisión Andina 351 de 1993 (Congreso de la República, Decisión Andina 351 de 1993) y la Ley 1915 de 2018 (Congreso de la República, Ley 1915 de 2018) que modifica la ley 23 de 1982. En el presente proyecto todo el desarrollo realizado estará amparado bajo las leyes vigentes de derechos de autor.

3.3.2 Ley de protección de datos personales

Estas leyes tratan sobre el reconocimiento y la protección del derecho que tiene una persona a conocer, actualizar y rectificar su información en bases de datos o archivos susceptibles a tratamiento de entidades públicas o privadas. La protección de datos personales se encuentra regulada en Colombia por la Ley Estatutaria 1581 de 2012 (Congreso de la República, Ley Estatutaria 1581 de 2012) y el Decreto número 1317 de 2013 (Congreso de la República, Decreto número 1317 de 2013). En el presente proyecto toda la información recopilada de los teléfonos inteligentes de los usuarios de la aplicación será manejada bajo las leyes vigentes de protección de datos personales.

4. DESARROLLO DEL PROYECTO

En esta sección se desarrollan los procedimientos necesarios para dar cumplimiento a los objetivos específicos planteados desde el anteproyecto, aunque para la metodología SCRUM no se considere necesaria la documentación de algunos de ellos.

4.1 HISTORIAS DE USUARIO

Se describieron las siguientes historias de usuario de acuerdo a las funcionalidades planteadas para la aplicación. Estas descripciones están contempladas como la interacción que puede tener el usuario con la aplicación SENSORTV. Se estimaron los siguientes puntos: 3 puntos (importancia baja); 5 puntos (importancia alta).

Historia de usuario 01

Historia de Usuario	
Número: HU1	Usuario: usuario
Nombre historia: consultar consumo en tiempo real	
Prioridad en negocio: Alta	Riesgo en desarrollo: Bajo
Puntos estimados: 5	
Descripción: Yo como usuario quiero consultar el consumo de los sensores en tiempo real para conocer el gasto de energía producido por los sensores	
Validación: caso de prueba 01, caso de prueba 02	

Historia de usuario 02

Historia de Usuario	
Número: HU2	Usuario: usuario
Nombre historia: consultar consumo en determinado tiempo	
Prioridad en negocio: Alta	Riesgo en desarrollo: Bajo
Puntos estimados: 5	
Descripción: Yo como usuario quiero obtener un informe del consumo de potencia miliwatt-minuto, dado un tiempo en minutos y almacenado en un archivo de tipo csv	
Validación: caso de prueba 04, caso de prueba 05	

4.2 DEFINICIÓN DE REQUERIMIENTOS

Teniendo en cuenta la información ofrecida por el grupo del proyecto de investigación "Sistema de comunicación para sobrevivientes de un desastre basado en una red ad hoc de teléfonos inteligentes", se definieron los siguientes requerimientos para la aplicación:

4.2.1 Requerimientos funcionales

No.	Requerimiento	Descripción del requerimiento
RF1	Monitorear sensores	La aplicación deberá monitorizar el consumo de energía en Miliwatts (Mw) de los cuatro sensores en el teléfono inteligente.
RF2	Generar reporte gráfico	La aplicación deberá generar un reporte gráfico del consumo de energía en tiempo real de los cuatro sensores
RF3	Generar reporte de datos	La aplicación deberá generar un reporte (archivo csv) con los datos en Miliwatts-Minuto, cuando el usuario ingrese un tiempo en minutos.

4.2.2 Requerimientos no funcionales

No.	Requerimiento	Descripción del requerimiento
RNF1	Espacio en disco	Para la instalación y el posterior funcionamiento de la aplicación, se necesita mínimo 10 megabytes de almacenamiento disponible en el teléfono inteligente.
RNF2	Portabilidad	La aplicación debe ser fácil de instalar
RNF3	Usabilidad	La aplicación debe ser fácil de usar, su interfaz debe ser amigable e intuitiva.
RNF4	Compatibilidad	La aplicación debe poder ejecutarse en dispositivos de diferente fabricante, teniendo en cuenta que cada fabricante añade características propias al sistema operativo.
RNF5	Eficiencia	La aplicación debe mostrar sus resultados en el tiempo de respuesta esperado.

4.2.3 Requerimientos de software

No.	Requerimiento	Descripción del requerimiento
RS1	IDE Android Studio	Para la implementación de la aplicación se hará uso del entorno de desarrollo IDE Android Studio versión 3.6.1, la clase BatteryManager y la interfaz SensorEventListener.

4.2.4 Requerimientos de hardware

No.	Requerimiento	Descripción del requerimiento
RH1	Computador con S.O. Windows 10	Para el desarrollo de la aplicación móvil se requiere un computador con sistema operativo Windows 10. Se eligió este sistema debido a las características de software.
RH2	Teléfono inteligente (Smartphone) con S.O. Android 8 (o superior)	Debido a que es una aplicación móvil se requiere un celular con sistema operativo Android 8 (o superior) para realizar las pruebas necesarias y para su funcionamiento. Se eligió este sistema debido al alcance del proyecto.

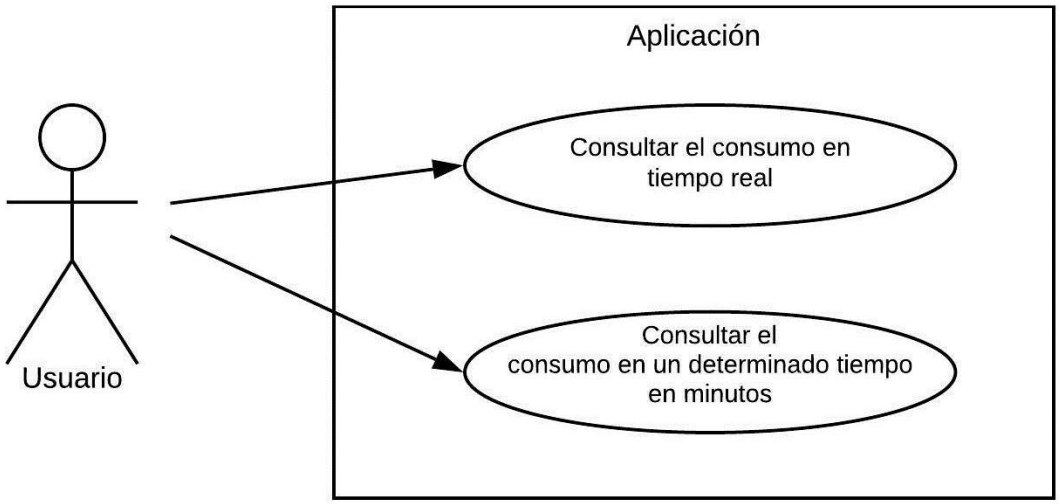
4.3 DIAGRAMACIÓN UML

El lenguaje UML (Lenguaje Unificado de Modelado) permite representar de manera visual: el diseño, la arquitectura, la implementación y el comportamiento de un software, mediante diferentes diagramas que abordan cada uno un aspecto diferente del sistema. En este proyecto se trabajó con tres de estos diagramas.

4.3.1 Diagrama de casos de uso

En este diagrama de comportamiento se representa la interacción que puede tener el usuario con la aplicación. Este diagrama se realizó con base en las historias de usuario como se puede observar en la Figura 9.

Figura 9. Diagrama de casos de uso



Fuente: elaboración propia. Herramienta usada: StarUML versión 3.2

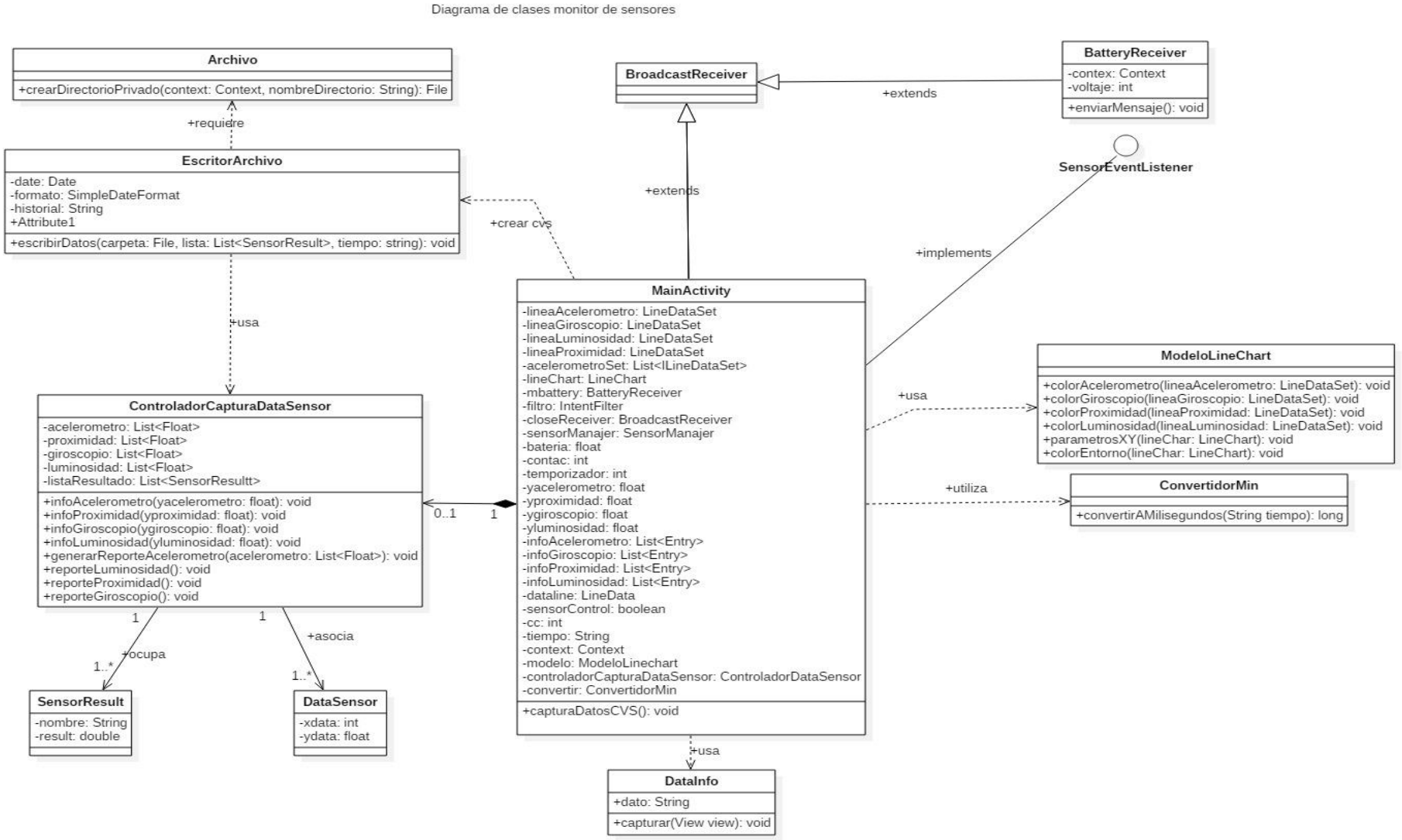
4.3.2 Diagrama de clases

En este diagrama de estructura se representan los elementos que debe contener el sistema, mediante clases, atributos, operaciones y las relaciones entre clases y objetos, como se puede observar en la Figura 10.

4.3.3 Diagrama de secuencia

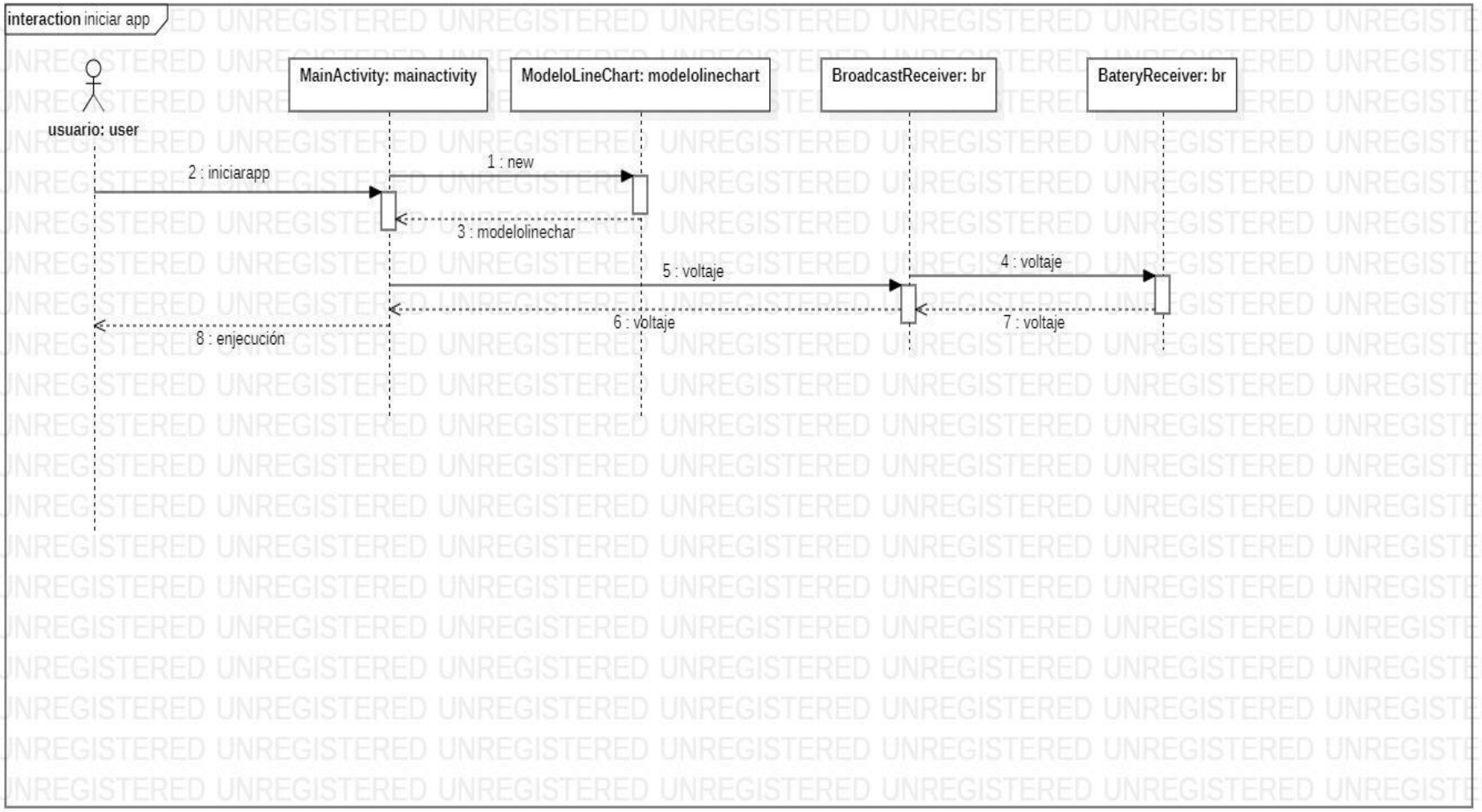
En este diagrama de comportamiento se representa mediante líneas la interacción de los procesos y objetos, el intercambio de mensajes para ejecutar una función o completar un proceso. Permite realizar una modelación de la lógica de una función, describiendo el cómo y el orden en el cual determinados objetos deben funcionar de manera conjunta, como se observa en las Figuras 11 y 12.

Figura 10. Diagrama de clases



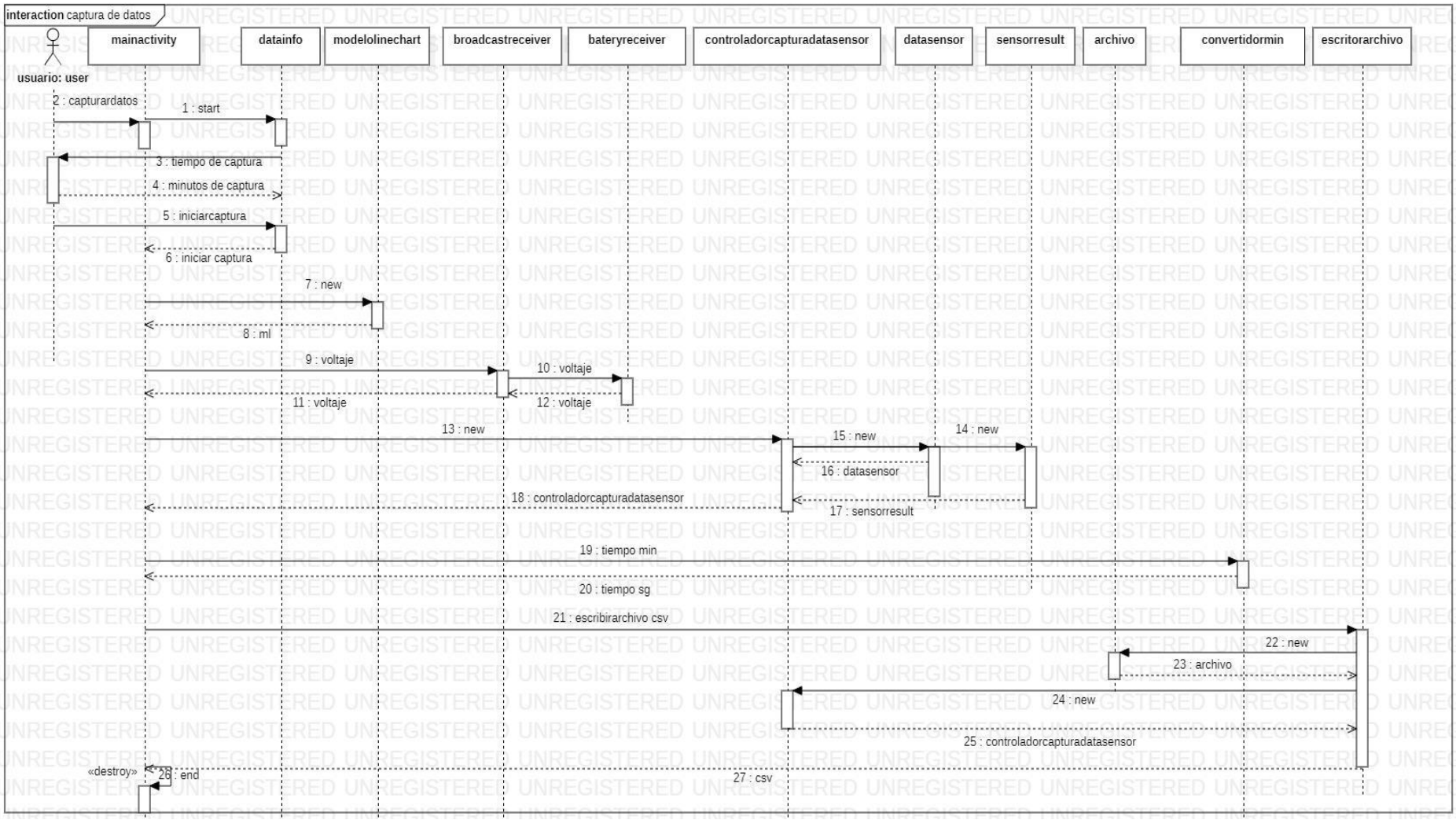
Fuente: elaboración propia. Herramienta usada: StarUML versión 3.2

Figura 11. Diagrama de secuencia de la inicialización de la aplicación



Fuente: elaboración propia. Herramienta usada: StarUML versión 3.2

Figura 12. Diagrama de secuencia de la captura de datos



Fuente: elaboración propia. Herramienta usada: StarUML versión 3.2

4.4 IMPLEMENTACIÓN

Inicialmente para la implementación de la aplicación, fue fundamental el uso de las siguientes herramientas que permiten obtener información, trabajar los datos y diseñar gráficas.

- BatteryManager[1], es una clase que se utiliza para el control y extracción de datos de la batería de un teléfono Android, permite tener acceso a las propiedades de la batería del teléfono.

- SensorEventManager[2], librería que permite el acceso a los sensores a utilizar.

- SensorEventListener[3], es una interfaz que se utiliza para recibir notificaciones del SensorManager cuando hay nuevos datos de un sensor.

- SensorManager[4], es una clase abstracta que permite acceder los sensores del teléfono inteligente

- MPAndroidChart[5], es una librería para crear gráficas lineales representadas en un plano cartesiano.

Para implementar las funcionalidades de la aplicación y el monitoreo de los sensores acelerómetro, de proximidad, giroscopio y de luminosidad, se aplicaron algunas fórmulas de la ley de Ohm y la ley de Watt, mencionadas en la sección del marco de referencia. Teniendo en cuenta estos fundamentos se realizaron los siguientes procedimientos:

4.4.1 Obtener el voltaje total de la batería del teléfono inteligente

Inicialmente es necesario conocer el voltaje total de la batería, debido a que este valor será la base para realizar posteriores cálculos. Para obtener este voltaje (Ley de Ohm) se utiliza la clase BatteryManager, la cual tiene acceso a esta información mediante sus funciones internas ya definidas, de esta forma se extrae el voltaje. Debido a que el valor del voltaje se genera en Voltios (V), se debe hacer una conversión a Milivoltios (mV), esto se hace multiplicando el valor del voltaje por 1000 (1 V = 1000 mV). Estos datos se envían al Main Activity para ser utilizados posteriormente.

4.4.2 Obtener la intensidad de cada uno de los sensores

Para obtener estos valores se utiliza la interfaz SensorEventListener en donde se crea la clase SensorManager, la cual incluye unos parámetros propios definidos que permiten habilitar los sensores y realizar lecturas sobre ellos. Después se utiliza el método onSensorChanged con el parámetro SensorEvent, en el cual se generan diferentes datos para cada uno de los sensores que han sido previamente inicializados (como el nombre del fabricante, el nombre del sensor, el estado, la Intensidad (I), la latencia propia, entre otros). De aquí se obtiene la Intensidad (I) de cada sensor en Miliamperios (mA).

[1] https://developer.android.com/reference/android/os/BatteryManager
[2] https://developer.android.com/reference/android/hardware/SensorEvent
[3] https://developer.android.com/reference/android/hardware/SensorEventListener
[4] https://developer.android.com/reference/android/hardware/SensorManager
[5] https://github.com/PhilJay/MPAndroidChart

4.4.3 Obtener la potencia de cada uno de los sensores

Para obtener esta potencia se requiere conocer el Voltaje de la batería en Milivoltios (mV) y la Intensidad (I) en Miliamperios (mA) de cada sensor, estos datos se obtuvieron en los procedimientos anteriores. Con estos datos se multiplica el Voltaje de la batería por la Intensidad (I) de cada uno de los sensores (Ley de Watt), de esta manera se obtiene la potencia en Miliwatts (mW) para cada sensor, estos valores son el resultado que arroja la aplicación y serán visualizados posteriormente en el eje *y* de la gráfica.

4.4.4 Monitoreo en el tiempo

En cuanto al tiempo se utiliza el método onSensorChanged, el cual genera una latencia (la latencia se refiere a los ciclos de muestreo cada cierto tiempo) de 200 Milisegundos[6]. Es importante tener en cuenta que esta latencia puede variar según el tipo de sensor (por ejemplo, el sensor acelerómetro por definición presenta una latencia de 200 Milisegundos, sin embargo, puede presentar una latencia ligeramente menor) e igualmente puede variar en algunos teléfonos inteligentes, en la documentación del método se informa que esto puede deberse al hardware, al comportamiento (aunque tiende a ser el mismo, no todos los teléfonos siguen este parámetro) y al entorno en que se encuentren (normalmente los sensores están definidos en 200 Milisegundos, pero si se estimulan mayormente pueden presentar una frecuencia de muestreo más pequeña).

Cada 200 Milisegundos este método onSensorChanged entrega los datos de la Intensidad (I) de los sensores que están activados. Luego de esto, la clase abstracta de tipo CountDownTimer captura los datos del método onSensorChanged con una latencia de 3 segundos, obteniendo con esto datos la potencia y los envía al graficador EmpyLineChart para posteriormente presentarlos en una gráfica lineal. Se programó 3 segundos de latencia para aumentar la exactitud en tiempo real e igualmente visualizar la gráfica de mejor manera.

4.4.5 Representación en la gráfica

Para la creación de la gráfica se requiere la librería LineChart, la cual permite crear un gráfico de líneas (en este proyecto se mostrarán los resultados asemejando un plano cartesiano). Por cada sensor que sea instanciado e inicializado se creará una lista que captura los datos del eje *x* y el eje *y*. Con estas listas y teniendo en cuenta que esta librería acepta listas con parámetros de tipo Entry, los cuales hacen referencia los ejes del plano cartesiano, se agregan los datos de los ejes, donde el eje *y* representa la potencia en Miliwatts para cada uno de los sensores (resultado que se obtuvo en procesos anteriores) y el eje *x* representa una variable contador se definió para que emulara 3 segundos de tiempo de cada ciclo, de esta manera se crea un archivo de tipo LineData el cual debe contener agregadas todas las listas de los sensores con su información y se agrega a un nuevo tipo de dato lineChart, que se encarga de ejecutar todas las listas y proyectarlas mediante una gráfica. La gráfica resultante se muestra al usuario.

4.4.6 Capturar datos

Para realizar la captura de datos se utiliza un método en el cual se utiliza una clase abstracta de tipo CountDownTimer, esta clase permite al usuario ejecutar una tarea en un

[6] https://developer.android.com/guide/topics/sensors/sensors_overview?hl=es#sensors-monitor

tiempo determinado (el usuario define este tiempo en minutos) durante una latencia determinada (se programó una latencia de un segundo). Cuando el usuario ingresa el tiempo en minutos, se inicia la clase controladorCapturaDataSensor que se encarga de capturar los datos que se estén presentando cada segundo de cada uno de los ejes y de los sensores, hasta que el tiempo ingresado termine, mostrando un mensaje indicando que la captura de datos ha finalizado. Luego de esto, la clase ContDownTimer implementa un segundo método llamado onFinish en donde se termina la captura de datos. En este método onFinish se utiliza la clase controladorCapturaDataSensor para generar todos los reportes de los sensores durante el tiempo que el usuario ingresó, obteniendo datos de tipo Miliwatts-Minuto. Posteriormente, se genera el reporte para los sensores, se crea un directorio donde se crea un archivo nombrado con la fecha y hora de la captura, el cual contiene la lista de resultados.

Para almacenar los datos en un archivo tipo csv es necesario realizar unas cuantas conversiones con el fin de entregar los datos finales en Miliwatts-Minuto (mW-m); cuando el usuario ingresa un tiempo en minutos en la interfaz de capturar datos, este tiempo es agregado a un temporizador de tipo CountDownTimer el cual requiere de un tiempo máximo y una latencia equivalente a 1 segundo, con esto queda establecido el periodo de tiempo y su latencia.

En cada periodo de latencia se captura toda la información de Intensidad (I) dada por los sensores en ese instante y es almacenada en una lista por cada sensor. Al terminar el proceso, se llama al método onFinish el cual permite realizar tareas después de haber terminado el tiempo dado por el usuario.

En este espacio se hacen las conversiones de los datos en Miliwatts previamente obtenidos, para ello es necesario convertir los datos de Miliwatts (mW) a Watts (W) y la latencia de 1 segundo convertirla a horas para así volver a la ecuación básica de potencia.

Para convertir Miliwatts (mW) a Watts (W) se debe dividir por 1000 todos los datos de las capturas, a su vez sabemos que un segundo equivale a 0,0002777777778 horas aproximadamente, esto se puede obtener planteando una regla de 3 donde una hora equivale a 3600 segundos:

Unidad hora (h)	**Equivalencia segundos (sg)**
1	3600
x	1

$$= \frac{\mathbf{1\ h*1\ sg}}{3600\ sg} = \frac{\mathbf{1}}{3600}$$

Dado que la latencia es la misma para todas las capturas, se toma este número como una constante, hasta este punto se tienen los siguientes datos: Watts y Horas.

Con estos datos se utiliza la Ley de Watt para obtener la potencia en el tiempo, multiplicando los Watts por la cantidad de horas de actividad:

W-h=w*h

Esta fórmula se aplica a todas las capturas. Luego de esto, se procede a sumar todos los resultados por sensor, obteniendo así un total de consumo en Watts-hora para cada uno de los sensores. Finalmente se toma este total por sensor y se convierte a Miliwatts-Minuto, para ello se multiplica los totales por 1000 para convertir los Watts a Miliwatts,

luego para convertir las horas a minutos se multiplica por 60, por lo que el nuevo total será en Miliwatts-Minuto (mW-m), a continuación se ilustra un ejemplo con el sensor acelerómetro:

Datos acelerómetro

Potencia mW	Latencia horas
0.13	1/3600
0.13	1/3600

Miliwatts a Watts	Watts
0.13/1000	1.3x10^-4
0.13/1000	1.3x10^-4

Watts	Latencia	Watts*latencia	Miliwatts-Minuto	**Resultado Miliwatts-Minuto**
1.3x10^-4	1/3600	3.61x10^-8	(3.61x10^-8)*60*1000	13/6000
1.3x10^-4	1/3600	3.61x10^-8	(3.61x10^-8)*60*1000	13/6000

4.4.7 Creación del fichero

Mediante la clase EscritorArchivo se genera un archivo de tipo csv dentro de una carpeta llamada “LACSERInfo”. Este archivo csv contiene los registros de los sensores a los que previamente se les realizó la captura de datos, mostrando el nombre del sensor, el tiempo que se le estuvo haciendo el monitoreo y el registro en Miliwatts-Minuto.

Inicialmente se contempló la posibilidad de guardar el archivo csv en una carpeta como mis documentos, descargas, imágenes, entre otras, ya que estas son habituales, de fácil acceso y el usuario tiene mayor interacción con ellas. Sin embargo, se encontró que el método que permitía guardar archivos en un directorio externo fue descontinuado por Android, debido a que viola la privacidad de las aplicaciones y del usuario, por lo que se generó un método en su reemplazo el cual permite guardar archivos de forma segura en el directorio de la propia aplicación, aislados de carpetas públicas como las mencionadas anteriormente y el usuario puede acceder libremente a estos archivos posteriormente en el teléfono inteligente[7]. Adicionalmente, la información generada en el archivo csv es propia de la aplicación SENSORTV y no debe dejarse abierta a ser editada desde otras aplicaciones.

Adicionalmente, en este trabajo se intentó realizar el monitoreo de los siguientes sensores:

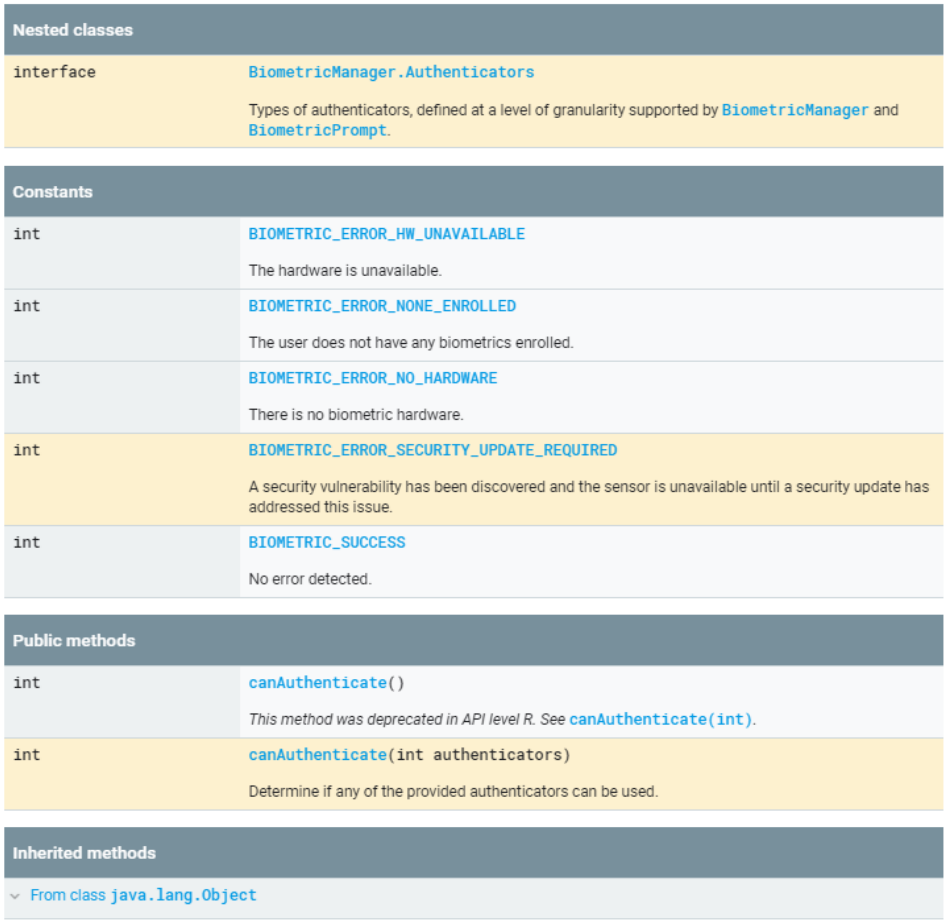
4.4.8 Sensores biométricos

Inicialmente en la propuesta se tenía contemplado realizar monitoreo al sensor biométrico junto con los otros sensores, sin embargo, durante el desarrollo se encontró que la clase BiometricManager que proporciona Android la cual controla este sensor, no cuenta con un método que permita acceder a información como su Intensidad (I) como se realizó con los otros cuatro sensores, es decir, para el sensor biométrico no es posible conocer la cantidad de energía que consume. Esta información se encuentra en la documentación

[7] documentación de Android sobre almacenamiento de archivos disponible en https://developer.android.com/training/data-storage/files?hl=es-419#WriteExternalStorage

oficial de Android[8], la cual ilustra las posibles funcionalidades e implementaciones de la clase BiometricManager, como se observa en la Figura 13.

Figura 13. Clases, constantes y métodos que ofrece la clase BiometricManager



Nested classes	
interface	BiometricManager.Authenticators Types of authenticators, defined at a level of granularity supported by BiometricManager and BiometricPrompt.

Constants	
int	BIOMETRIC_ERROR_HW_UNAVAILABLE The hardware is unavailable.
int	BIOMETRIC_ERROR_NONE_ENROLLED The user does not have any biometrics enrolled.
int	BIOMETRIC_ERROR_NO_HARDWARE There is no biometric hardware.
int	BIOMETRIC_ERROR_SECURITY_UPDATE_REQUIRED A security vulnerability has been discovered and the sensor is unavailable until a security update has addressed this issue.
int	BIOMETRIC_SUCCESS No error detected.

Public methods	
int	canAuthenticate() *This method was deprecated in API level R. See* canAuthenticate(int).
int	canAuthenticate(int authenticators) Determine if any of the provided authenticators can be used.

Inherited methods
From class java.lang.Object

Fuente: imagen tomada de la documentación de Android para la clase BiometricManager[9]

Adicionalmente, cuando se planteó la propuesta se contempló trabajar con la interfaz SensorEventListener para el desarrollo de este proyecto, sin embargo, según la guía de sensores disponible en la documentación oficial de Android[10], se encontró que este sensor biométrico no está incluido en la lista de tipos de sensores compatibles con la plataforma de Android, lo que indica que no es considerado un sensor, por lo tanto no es posible conocer su potencia ya que no puede trabajarse dentro de la interfaz SensorEventListener la cual controla los eventos para los sensores compatibles[11]. En la Figura 14 se muestra el tipo de sensores compatibles, donde se observa que no se menciona al sensor biométrico.

[8] https://developer.android.com/
[9] disponible en https://developer.android.com/reference/android/hardware/biometrics/BiometricManager
[10] https://developer.android.com/guide/topics/sensors/sensors_overview
[11] documentación de Android para la interfaz SensorEventListener
https://developer.android.com/reference/android/hardware/SensorEventListener

Figura 14. Tipos de sensores compatibles con la plataforma de Android

Sensor
TYPE_ACCELEROMETER
TYPE_AMBIENT_TEMPERATURE
TYPE_GRAVITY
TYPE_GYROSCOPE
TYPE_LIGHT
TYPE_LINEAR_ACCELERATION
TYPE_MAGNETIC_FIELD
TYPE_ORIENTATION
TYPE_PRESSURE
TYPE_PROXIMITY
TYPE_RELATIVE_HUMIDITY
TYPE_ROTATION_VECTOR
TYPE_TEMPERATURE

Fuente: imagen tomada de la documentación de Android sobre la guía de los sensores[12]

Con base en lo anterior, teniendo en cuenta que no es posible obtener datos de su Intensidad (I) y para Android no es considerado un sensor, el sensor biométrico fue descartado de la propuesta.

4.4.9 GPS

En cuanto al sensor GPS, aunque no estaba inicialmente contemplado en la propuesta, se intentó realizar monitoreo sobre este. Después de realizar las investigaciones sobre este sensor, no fue posible encontrar un método o una librería que permita obtener datos de su Intensidad (I), sin embargo, Android proporciona datos predeterminados del consumo de intensidad dependiendo de la actividad que esté desarrollando del GPS, el escenario en que se encuentre o su estado (activo/inactivo), estos son datos aproximados al valor real de su consumo[13]. Esto quiere decir que no es posible detectar el consumo en tiempo real del GPS debido a que los datos a los que se tiene acceso sobre este sensor son valores estáticos.

En otros trabajos que se han realizado sobre lecturas a este sensor, como la aplicación PowerTutor (Zhang *et al*., 2010), los autores utilizan estos valores estáticos proporcionados por Android para realizar su experimentación, lo cual se puede observar en el repositorio público de la plataforma Github donde se encuentra el código de esta aplicación[14], tener diferentes valores estáticos permite que se obtenga diferentes resultados en sus pruebas dado que estas investigaciones están enfocadas en la

[12] disponible en https://developer.android.com/guide/topics/sensors/sensors_overview
[13] disponible en el recurso https://android.googlesource.com/platform/frameworks/base/+/master/core/res/res/xml/power_profile.xml
[14] disponible en el repositorio https://github.com/msg555/PowerTutor/commit/dd56456c9780576206ae45e55904bdcc2d87cc1b

capacidad de la batería como variante de los sensores, además de presentar resultados a modo de porcentajes se puede obtener datos de gran variación, respecto al GPS o a cualquier sensor.

En este proyecto se descartó el GPS debido a que no fue posible obtener datos de su Intensidad (I) y no se utilizaron los datos predeterminados que proporciona Android debido a que son valores aproximados que no significan consumo en tiempo real, lo cual es el alcance de este trabajo, además del hecho de que no se maneja porcentajes respecto a la capacidad actual de la batería, como si lo hacen en el ya mencionado artículo.

4.5 PRUEBAS

Para la aplicación SENSORTV se describieron las siguientes pruebas de funcionalidad, estas pruebas se realizaron en condiciones normales, es decir en un escenario donde no se realizaron estimulaciones a los sensores, activar otras aplicaciones o activar el modo avión.

4.5.1 Caso de prueba 01

Caso de prueba								
Id	**Nombre caso**				**Historia asociada**			
01	Validar sensores				HU1			
Módulo	**Fecha de prueba**				**Responsable:**			
Lectura de sensores	30-05-2020				Mónica Parra			
Descripción del caso de prueba		***Caso de prueba***						
		Paso	***X***	***Fallo***		***No ejecutó***		
Verificar la inicialización de los sensores que contiene el teléfono		**Prioridad:**						
		Alta	**X**	Media		Baja		

Requisitos		
No.	**Prerrequisitos**	**Datos de prueba**
01	Tener instalada la aplicación en un teléfono inteligente con sistema operativo Android 8.0 o versiones superiores	Samsung Galaxy J7 Pro, sensor giroscopio, sensor de proximidad, sensor de luminosidad, sensor acelerómetro

Resultados			
No.	**Detalles del paso**	**Resultado esperado**	**Estado**
01	Iniciar la aplicación SENSORTV y observar los sensores presentes en el gráfico	Visualizar los sensores giroscopio, de proximidad, de luminosidad, acelerómetro	Pasó
02	Iniciar la aplicación SENSORTV y observar las convenciones de los sensores presentes en el gráfico	Visualizar el mismo número de convenciones de los sensores y de líneas presentes en el gráfico, las cuales representan a los sensores giroscopio, de proximidad, de luminosidad, acelerómetro	Pasó
Ver captura en la Figura 15			

4.5.2 Caso de prueba 02

Caso de prueba								
Id	**Nombre caso**				**Historia asociada**			
02	Sensor de proximidad				HU1			
Módulo	**Fecha de prueba**				**Responsable:**			
Lectura de sensor de proximidad	30-05-2020				Ermes Cerón			
Descripción del caso de prueba		***Caso de prueba***						
		Paso	***X***	***Fallo***		***No ejecutó***		
Verificar el cambio de potencia del sensor de proximidad al estimularlo		**Prioridad:**						
		Alta	**X**	Media		Baja		
Requisitos								
No.	**Prerrequisitos**				**Datos de prueba**			
01	Tener instalada la aplicación en un teléfono inteligente con sistema operativo Android 8.0 o versiones superiores				Samsung Galaxy J5 Prime, sensor de proximidad			
Resultados								
No.	**Detalles del paso**	**Resultado esperado**						**Estado**
01	Iniciar la aplicación SENSORTV y estimular el sensor de proximidad	Se debe visualizar cambios en la línea que representa la potencia del sensor de proximidad en el gráfico						Pasó
Ver captura en la Figura 16								

4.5.3 Caso de prueba 03

Caso de prueba								
Id	**Nombre caso**				**Historia asociada**			
03	Sensor de proximidad				HU1			
Módulo	**Fecha de prueba**				**Responsable:**			
Lectura de sensor de proximidad	30-05-2020				Ermes Cerón			
Descripción del caso de prueba		***Caso de prueba***						
		Paso	***X***	***Fallo***		***No ejecutó***		
Verificar el cambio de potencia del sensor de proximidad al estimularlo		**Prioridad:**						
		Alta	**X**	Media		Baja		
Requisitos								
No.	**Prerrequisitos**				**Datos de prueba**			
01	Tener instalada la aplicación en un teléfono inteligente con sistema operativo Android 8.0 o versiones superiores				Samsung Galaxy J5 Prime, sensor de proximidad			
Resultados								
No.	**Detalles del paso**	**Resultado esperado**						**Estado**
01	Iniciar la aplicación SENSORTV y estimular el sensor de proximidad	Se debe visualizar cambios en la línea que representa la potencia del sensor de proximidad en el gráfico						Pasó
Ver captura en la Figura 17								

4.5.4 Caso de prueba 04

Caso de prueba								
Id	**Nombre caso**				**Historia asociada**			
04	Obtención de datos				HU2			
Módulo	**Fecha de prueba**				**Responsable:**			
Capturar datos	31-05-2020				Mónica Parra			
Descripción del caso de prueba		***Caso de prueba***						
		Paso	***X***	***Fallo***		***No ejecutó***		
Verificar que se guarden los datos luego de realizar una captura de datos		**Prioridad:**						
		Alta	**X**	Media		Baja		
Requisitos								
No.	**Prerrequisitos**				**Datos de prueba**			
01	Tener instalada la aplicación en un teléfono inteligente con sistema operativo Android 8.0 o versiones superiores				Samsung Galaxy J7 Pro, captura de datos por un minuto			
Resultados								
No.	**Detalles del paso**	**Resultado esperado**						**Estado**
01	Iniciar la aplicación SENSORTV y ejecutar la funcionalidad Capturar datos	Se debe encontrar un archivo tipo csv en la ubicación que menciona la aplicación SENSORTV						Pasó
Ver captura en la Figura 18								

4.5.5 Caso de prueba 05

Caso de prueba								
Id	**Nombre caso**				**Historia asociada**			
05	Visualización de datos				HU2			
Módulo	**Fecha de prueba**				**Responsable:**			
Capturar datos	31-05-2020				Mónica Parra			
Descripción del caso de prueba		***Caso de prueba***						
		Paso	***X***	***Fallo***		***No ejecutó***		
Visualizar los datos que se han guardado en el archivo tipo csv		**Prioridad:**						
		Alta	**X**	Media		Baja		
Requisitos								
No.	**Prerrequisitos**				**Datos de prueba**			
01	Tener instalada la aplicación en un teléfono inteligente con sistema operativo Android 8.0 o versiones superiores				Samsung Galaxy J7 Pro, captura de datos por un minuto			
Resultados								
No.	**Detalles del paso**	**Resultado esperado**						**Estado**
01	Iniciar la aplicación SENSORTV y ejecutar la funcionalidad Capturar datos	Se debe crear archivo tipo csv que contenga 60 registros de un segundo de las capturas individuales como también un registro del consumo de potencia total de cada sensor						Pasó
Ver captura en la Figura 19								

Figura 15. Captura del caso de prueba 01

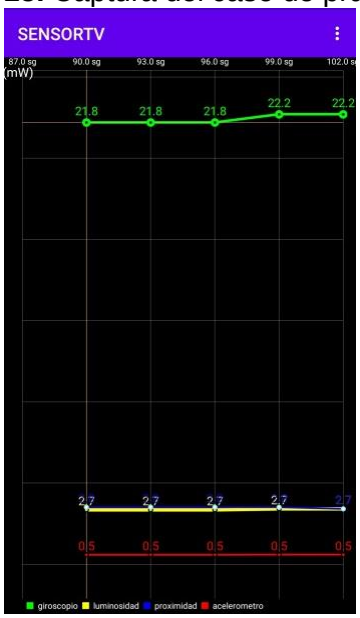


Figura 16. Captura del caso de prueba 02

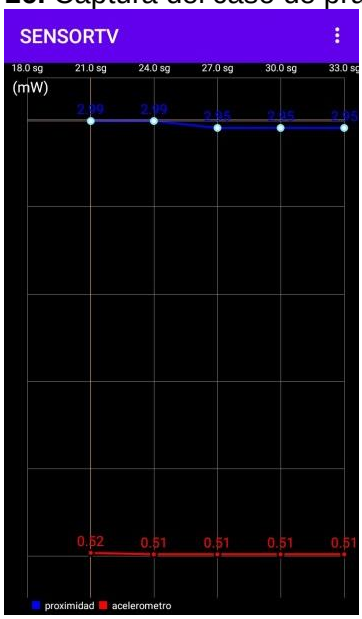


Figura 17. Captura del caso de prueba 03

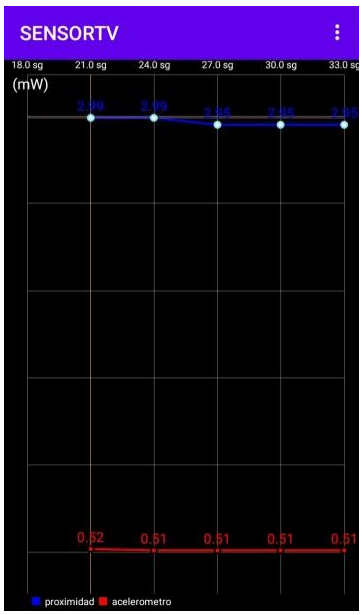
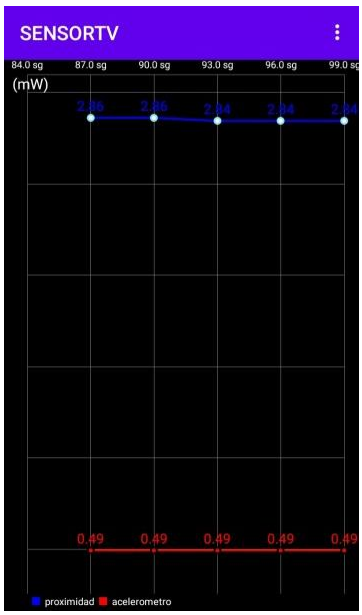
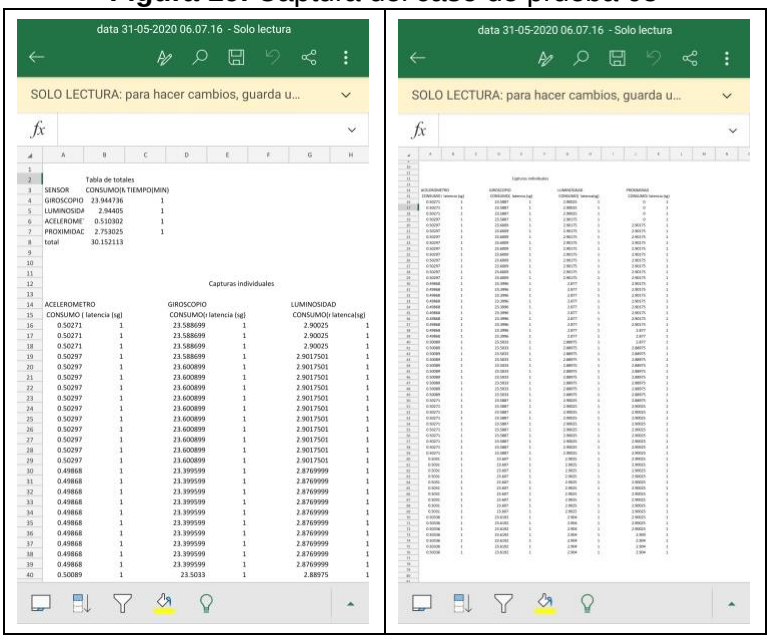
 Ejecución 1 (sin estimulación)	 Ejecución 2 (con estimulación, escenario: se aumentó el brillo al nivel máximo, se abrieron dos aplicaciones simultáneamente, se giró el teléfono repetidamente)

Figura 18. Captura del caso de prueba 04

Figura 19. Captura del caso de prueba 05



Adicionalmente, para este proyecto se describió una serie de pruebas tomando como referencia al conjunto de normas ISO/IEC 25000 denominada SQuaRE, específicamente en la norma ISO/IEC 25010 calidad del producto software[15], en donde se medirán las siguientes características:

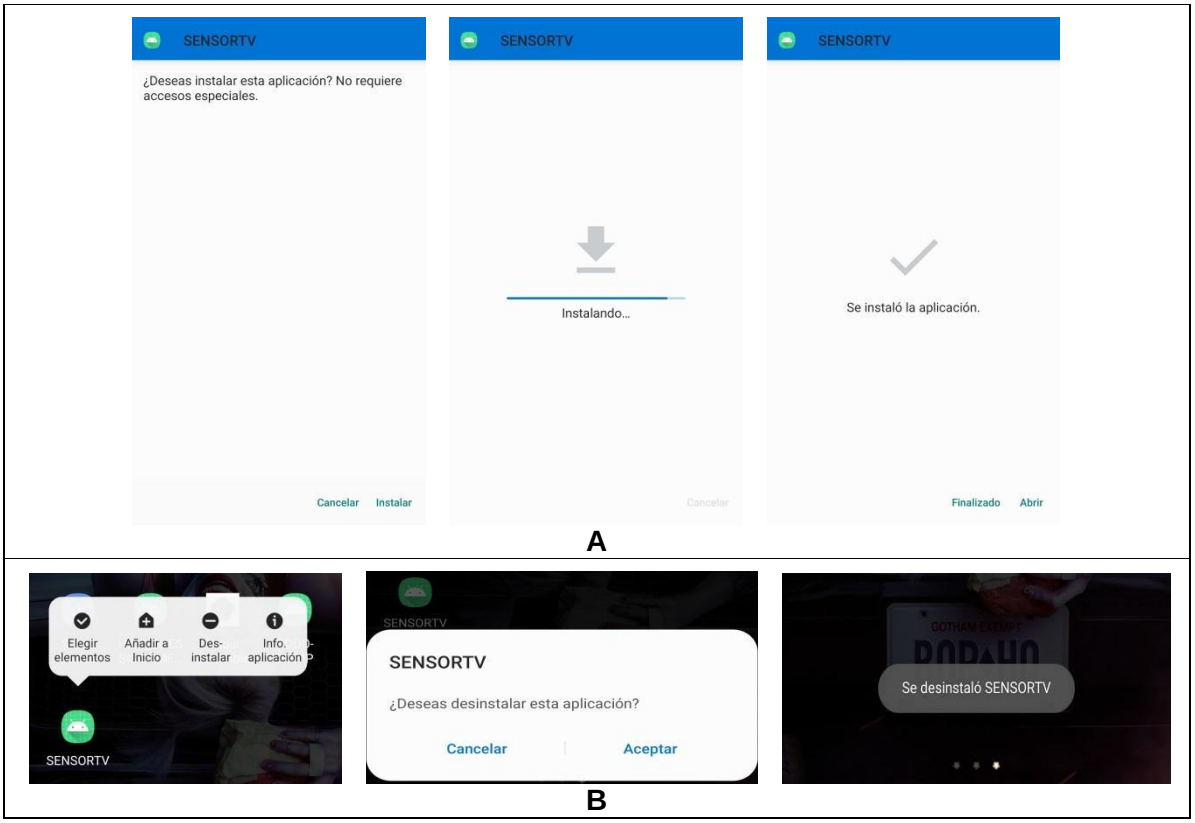
4.5.6 Portabilidad

Es la capacidad de transferir el producto de un entorno hardware o software a otro de forma efectiva y eficiente. Se medirán las subcaracterísticas **adaptabilidad** y **capacidad para ser instalado**, mediante su instalación y desinstalación en dispositivos de distintos fabricantes (marcas, modelos), todos con sistema operativo Android.

Resultado de la prueba: la aplicación SENSORTV se instaló y desinstaló en repetidas ocasiones en diferentes teléfonos de marcas Samsung y Motorola, no se presentó fallo alguno (Figura 20).

[15] disponible en https://iso25000.com/index.php/normas-iso-25000/iso-25010

Figura 20. Proceso de instalación y desinstalación de la aplicación SENSORTV. **A.** Instalación. **B.** Desinstalación

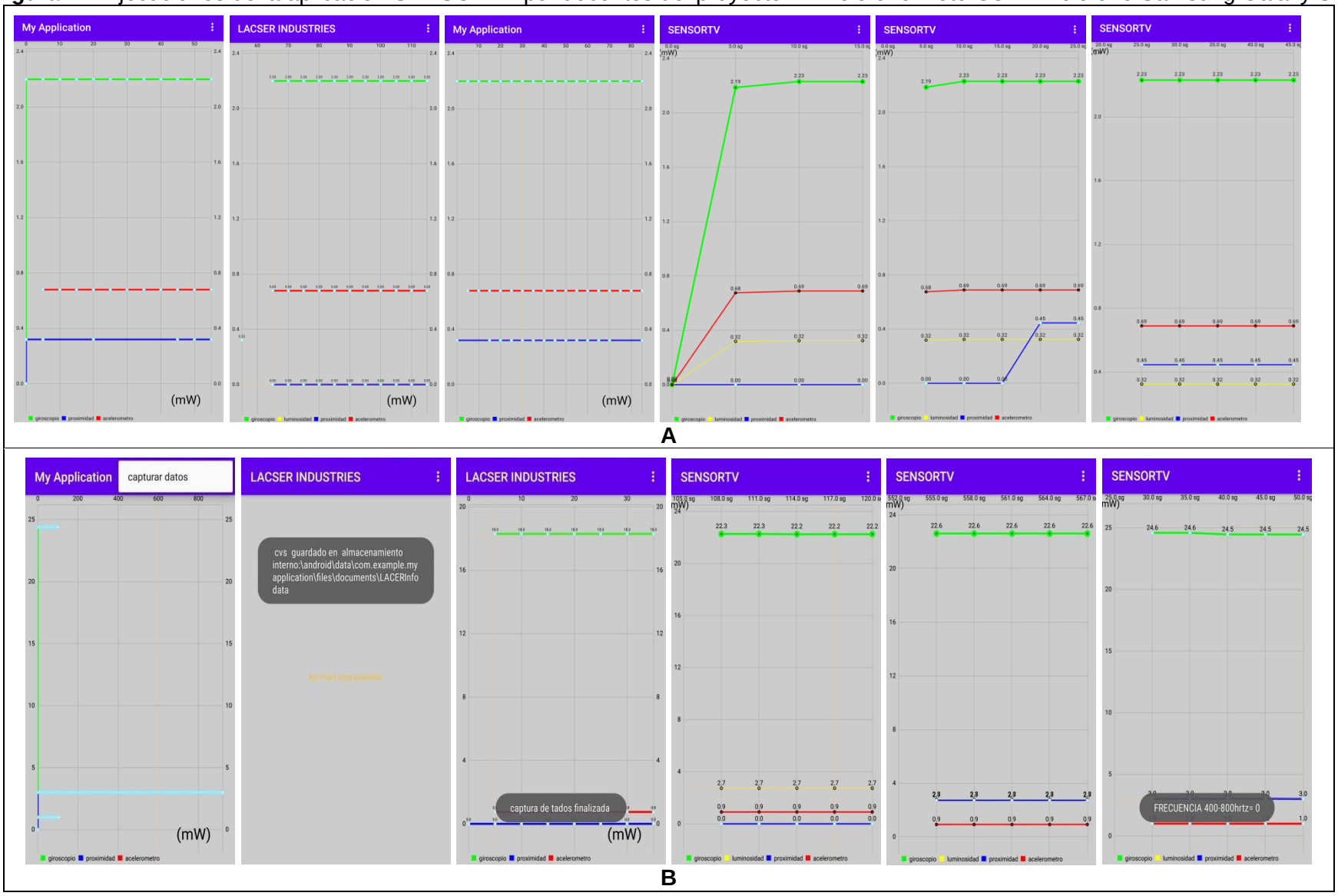


4.5.7 Usabilidad

Es la capacidad del software para ser entendido, aprendido y usado. Se medirán las subcaracterísticas **capacidad de aprendizaje, capacidad para ser usado, estética de la interfaz de usuario, accesibilidad**, cuando docentes del grupo de investigación ejecuten la aplicación.

Resultado de la prueba: los docentes involucrados en este proyecto ejecutaron la aplicación conforme se iba diseñando, sus resultados se pueden observar en la Figura 21.

Figura 21. Ejecuciones de la aplicación SENSORTV por docentes del proyecto. **A.** Teléfono Moto G8. **B.** Teléfono Samsung Galaxy S7



4.5.8 Compatibilidad

Es la capacidad del sistema o componentes de realizar sus funciones al compartir el entorno hardware o software. Se medirá la subcaracterística **coexistencia**, mediante la ejecución de la aplicación en dispositivos de diferente fabricante (marca, modelo) y con el sistema operativo Android.

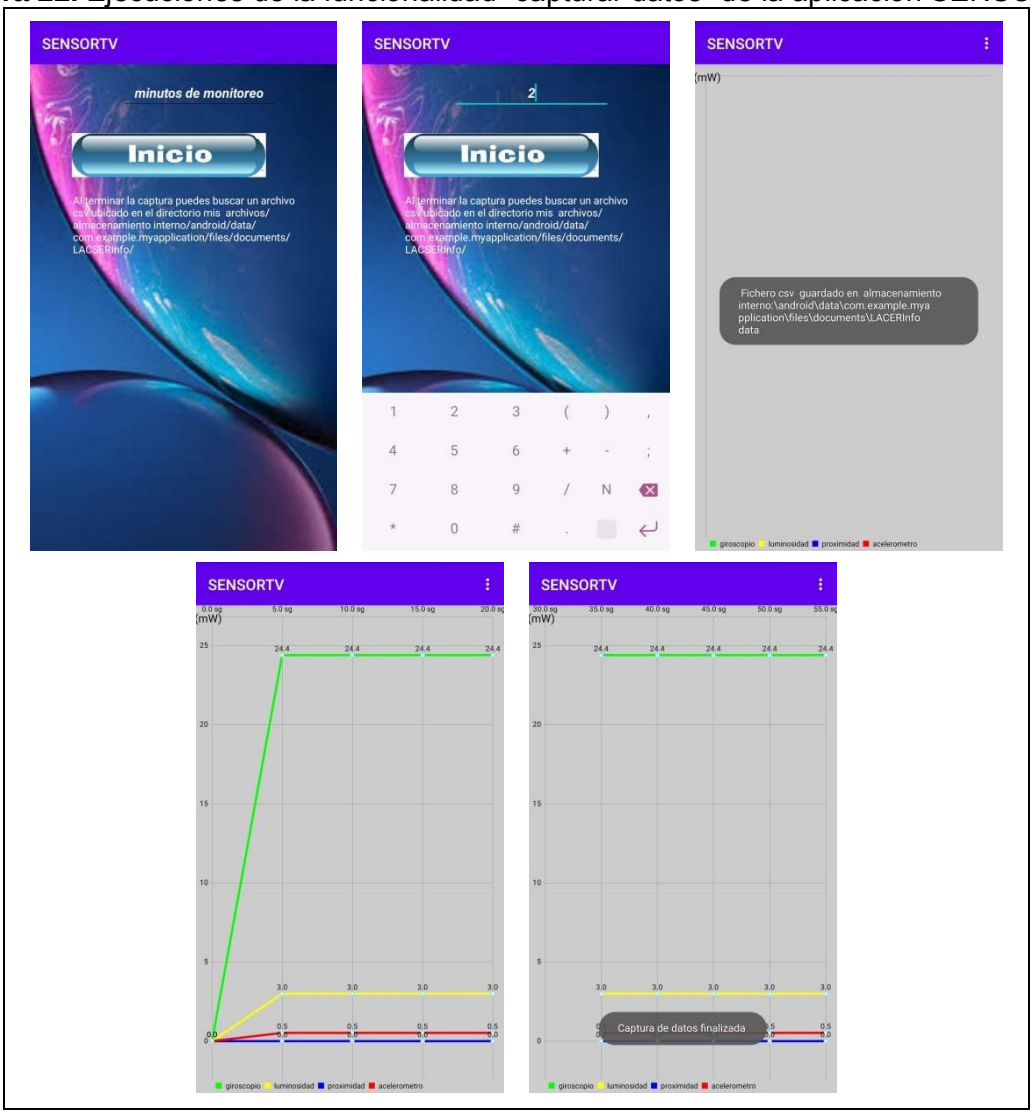
Resultado de la prueba: la aplicación SENSORTV fue ejecutada en los celulares Moto G8, Samsung Galaxy S7, Samsung Galaxy J5 Prime y Samsung Galaxy J7 Pro, en todos generó la gráfica en la cual se observa la potencia en tiempo real y se obtuvo resultados similares a las Figuras 21 y 26.

4.5.9 Eficiencia de desempeño

Es el desempeño relativo a la cantidad de recursos utilizados en ciertas condiciones. Se medirá la subcaracterística **comportamiento temporal**, cuando se ejecute la aplicación en modo captura de datos, se ingrese una cantidad de minutos y genere un archivo con los datos del monitoreo solo cuando termine el tiempo ingresado.

Resultado de la prueba: cuando se ejecuta la funcionalidad de captura de datos, luego de que el usuario ingresa una cantidad en minutos, la aplicación SENSORTV comienza a guardar las lecturas de cada sensor mientras muestra la gráfica de consumo en tiempo real, una vez termina el tiempo de captura la aplicación muestra un mensaje informando que la captura de datos ha finalizado y genera un archivo de tipo csv que puede ser consultado posteriormente, la ubicación del archivo se muestra en la interfaz de captura de datos y en un mensaje cuando inicia la captura (Figura 22).

Figura 22. Ejecuciones de la funcionalidad “capturar datos” de la aplicación SENSORTV

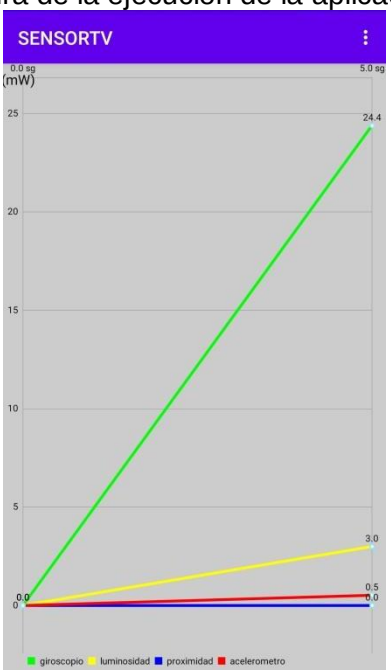


5. RESULTADOS OBTENIDOS

5.1 FUNCIONALIDAD

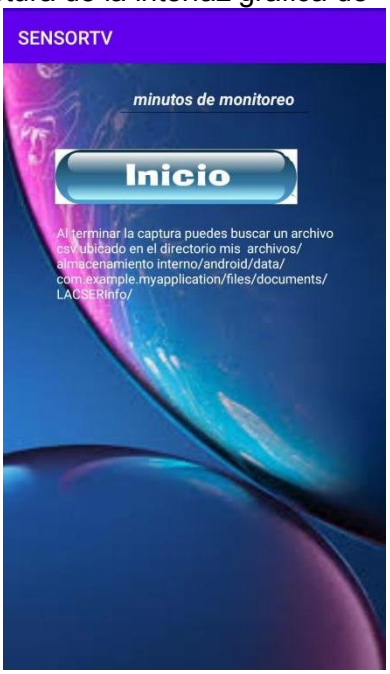
En este trabajo se desarrolló una aplicación denominada SENSORTV capaz de monitorear el consumo de energía en Miliwatts (Mw) de los sensores giroscopio, de proximidad, acelerómetro y de luminosidad. Su funcionamiento inicia cuando el usuario la ejecuta, comienza de forma automática y constante en tiempo real, muestra los sensores que se encuentren activos de los cuatro (giroscopio, de proximidad, acelerómetro, de luminosidad), teniendo en cuenta que se eligieron estos cuatro por ser los más comunes en los teléfonos inteligentes, aunque puede que alguno no esté disponible (depende de la gama del teléfono inteligente). La aplicación SENSORTV empieza a realizar lecturas en tiempo real de los sensores que haya leído, sin importar que no estén todos los cuatro sensores activos. Realiza la lectura cada 3 segundos aproximadamente. SensorEventListener informa que la latencia puede cambiar dependiente al ambiente en que se encuentre, si los sensores reciben un mayor estímulo es posible que generen más datos). La aplicación SENSORTV muestra en una gráfica de plano cartesiano el consumo instantáneo de los sensores en Miliwatts (Mw), cada sensor es representado por una línea de color distintivo; de luminosidad color amarillo, de proximidad color azul, acelerómetro color rojo y giroscopio color verde (Figura 23). En el eje *y* se ilustra de forma dinámica la potencia consumida por el sensor en Miliwatts (Mw) y el eje *x* se presenta el tiempo que está transcurriendo en intervalos de 3 segundos que se van sumando. La aplicación SENSORTV descarta los datos anteriores (más antiguos) y continúa con los actuales (más recientes) que son los que se van mostrando en la gráfica.

Figura 23. Captura de la ejecución de la aplicación SENSORTV



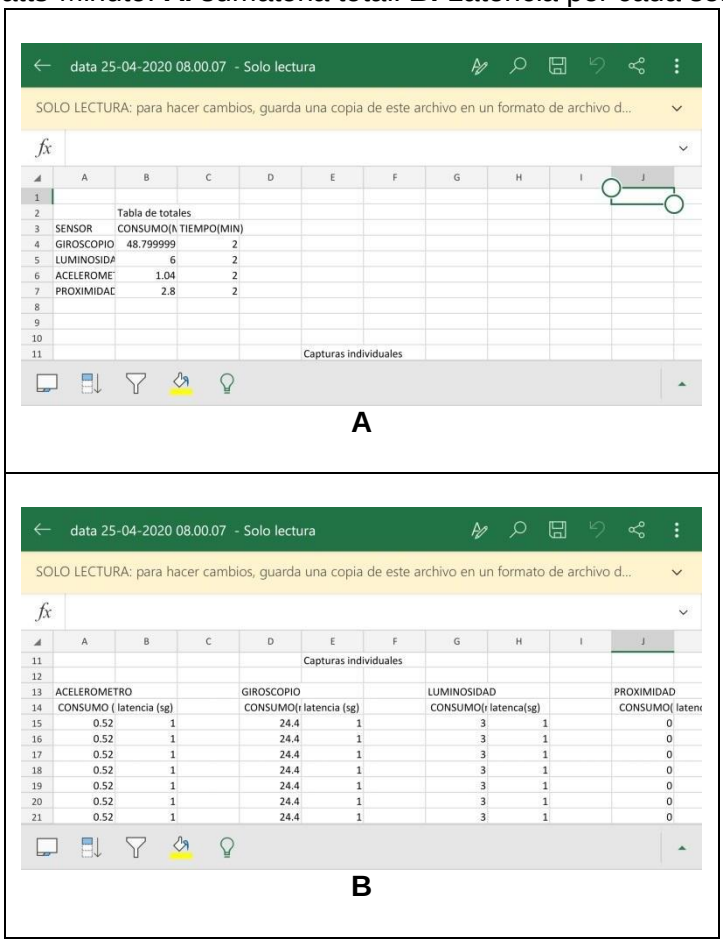
En la parte de capturar datos, la aplicación SENSORTV cuenta con un menú en el cual es posible seleccionar la opción de captura de datos. Cuando se ingresa en esta opción, se muestra una interfaz en la cual se puede ingresar el número de minutos en los cuales la aplicación SENSORTV realizará la captura de datos (Figura 24). En esta parte se requiere que el usuario ingrese un número (entero), que será el tiempo en minutos que la aplicación SENSORTV realice la lectura o recopilado de datos de consumo de los sensores en Miliwatts (Ms) por minuto.

Figura 24. Captura de la interfaz gráfica de “capturar datos”



Una vez se seleccione el botón iniciar captura, la aplicación SENSORTV regresa de nuevo a la ejecución principal, donde se muestra el gráfico de potencia en Miliwatts (Ms) de los sensores activos, esta vez guardando los datos que está capturando en intervalos de un segundo. Se eligió que los intervalos de captura fueran de un segundo para tener una curva aceptable de información y con un intervalo pequeño de tiempo se adquiere una mayor precisión. Cuando el tiempo de la captura ingresado por el usuario finaliza, la aplicación SENSORTV muestra un mensaje informando que la captura de datos ha terminado junto con la ubicación donde ha guardado el archivo tipo csv (directorio_mis archivos/almacenamiento_interno/android/data/com.example.myapplication/files/documents/LACSERInfo/). El archivo tipo csv es nombrado con fecha y hora de finalización de la captura, la información se muestra en una tabla con el nombre del sensor, tiempo de captura de datos, la potencia en Miliwatts-Minutos cada sensor y la sumatoria final (Figura 25). Después de terminar con la captura de datos, la aplicación SENSORTV sigue mostrando la gráfica del consumo en tiempo real. La aplicación SENSORTV se detiene cuando el usuario sale de ella.

Figura 25. Captura del archivo tipo csv que contiene la información de potencia en Miliwatts-Minuto. **A.** sumatoria total. **B.** Latencia por cada sensor.

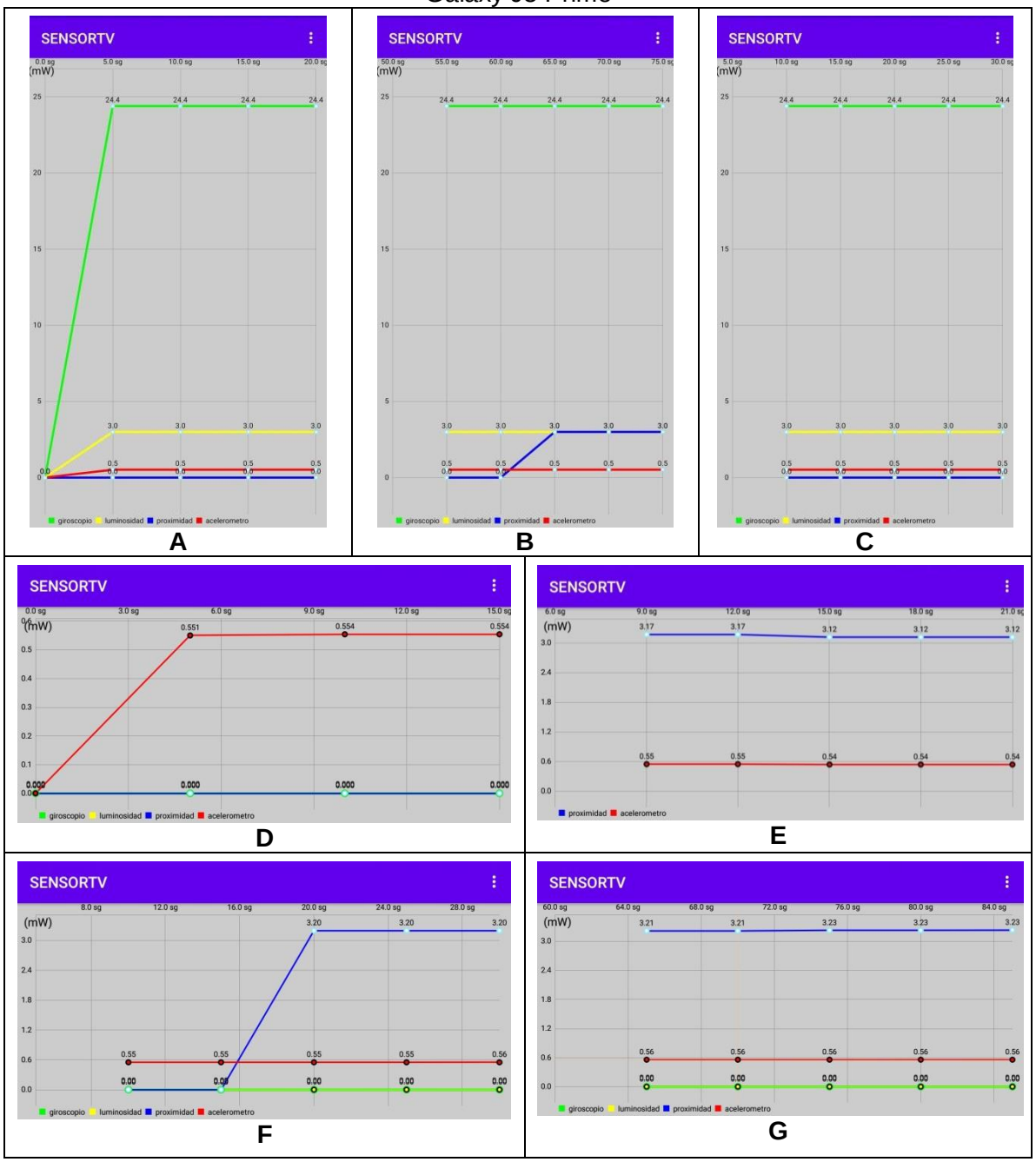


Es importante tener en cuenta que cuando se está ejecutando una captura de datos, la aplicación SENSORTV no permite que el usuario inicie otra, es decir que solo se permite una captura de datos por vez.

Esta aplicación SENSORTV permite hacer lecturas y capturas en segundo plano, mientras que el usuario no cierre la aplicación, si esto ocurre se detendrán las lecturas y capturas.

En la Figura 26 se muestra la gráfica resultante al ejecutar la aplicación SENSORTV en tiempo real.

Figura 26. Resultados de la ejecución de la aplicación SENSORTV. **A, B y C** ejecución en el teléfono Samsung Galaxy J7 Pro. **D, E, F y G** ejecución en el teléfono Samsung Galaxy J5 Prime



En estos resultados se puede observar lo siguiente:

Sensor de proximidad: inicia en un valor cero y debe ser estimulado para observar su variación, como se observa en la Figuras 26B y 26F. Esto se debe a que este sensor está clasificado dentro del grupo de sensores de posición[16] y es un sensor de activación (wake-up sensors), el cual solo activa un evento de informe cuando son estimulados, así despiertan al AP (aplicación proceso) e informan en ese instante de tiempo

[16] https://developer.android.com/guide/topics/sensors/sensors_position?hl=es

independientemente de su latencia máxima, en la Figura 27 se observa un fragmento tomado de la documentación oficial de Android sobre wake-up sensors[17].

Figura 27. Fragmento de la documentación de Android sobre sensores de activación

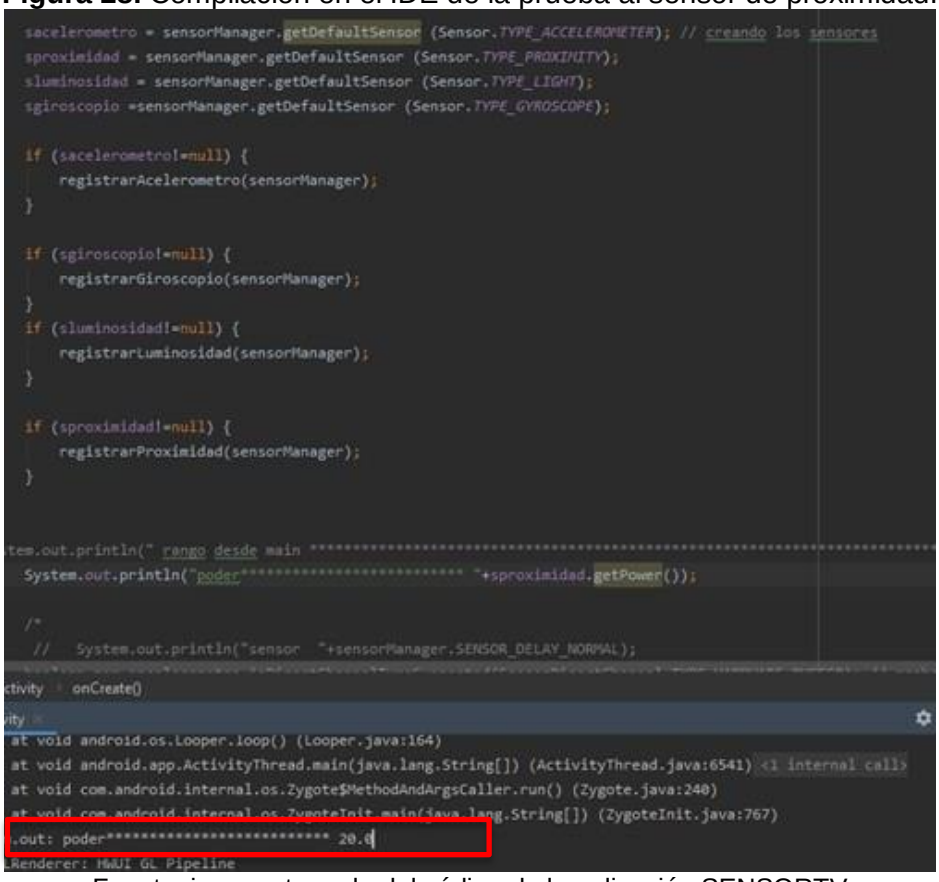
Wake-up sensors

In opposition to non-wake-up sensors, wake-up sensors ensure that their data is delivered independently of the state of the AP. While the AP is awake, the wake-up sensors behave like non-wake-up-sensors. When the AP is asleep, wake-up sensors wake up the AP to deliver events. That is, the AP will wake up and the sensor will deliver the events before the maximum reporting latency is elapsed or the hardware FIFO gets full. See `SensorManager#registerListener(SensorEventListener, Sensor, int, int)` for more details.

Fuente: imagen tomada de la documentación de Android sobre sensores de activación[18]

Para conocer el estado de este sensor antes de una estimulación, se realizó una prueba al sensor de proximidad. En el resultado se observa que el sensor presenta un estado de actividad antes de su estimulación, esto se logra mediante la captura de su Intensidad (I) antes de iniciar un evento. En la Figura 28, se puede observar una intensidad de 20.0, en ese estado el sensor no ha sido activado y tampoco se ha generado un evento del sensor.

Figura 28. Compilación en el IDE de la prueba al sensor de proximidad.



Fuente: imagen tomada del código de la aplicación SENSORTV

[17] https://developer.android.com/reference/android/hardware/Sensor#TYPE_PROXIMITY y https://developer.android.com/reference/android/hardware/Sensor#isWakeUpSensor()

[18] disponible en https://developer.android.com/reference/android/hardware/Sensor#isWakeUpSensor()

Sensores acelerómetro y giroscopio: estos sensores pertenecen al grupo de sensores de movimiento[19], guardan información de los eventos en una lista tipo FIFO (first in, first out), estos eventos transcurren mientras el AP (aplicación proceso) está en modo suspensión (en este modo consume muy poca energía), cuando transcurre la máxima latencia establecida en el AP, despierta e informa de los eventos que se encuentran en la lista[20].

Sensor de luminosidad: pertenece al grupo de sensores de ambiente[21], este sensor es utilizado por la mayoría de fabricantes de dispositivos para controlar el brillo de la pantalla, es de aclarar que este sensor de luminosidad no está disponible en todos los teléfonos inteligentes.

Estos resultados que arroja la aplicación SENSORTV son valores que tienden a mostrar pequeñas variaciones durante el monitoreo como se puede observar en la Figura 26, esto se debe a que para calcular la potencia de cada uno de los sensores en Miliwatts (Mw) se utilizaron dos valores: el primero es el voltaje total de la batería del teléfono inteligente, los dispositivos móviles generalmente incluyen una batería de iones de Litio (li-ion) la cual tiene unos límites de variación desde ~ 2.9 Voltios hasta 4.2 Voltios, nominalmente ~ 3.6 Voltios (Le *et al*., 2013), el segundo es la Intensidad (I) de cada uno de los sensores. Por esta razón la potencia resultante se observa con pequeñas variaciones durante el monitoreo, debido a la variación del voltaje.

En el desarrollo realizado en la aplicación PowerProfiler (Tuysuz *et al*., 2019), los autores utilizan la capacidad actual de batería para realizar sus lecturas, es decir que utilizan la carga (o nivel) de batería que tiene el teléfono inteligente al momento de ejecutar la aplicación, de este modo presentan sus resultados como un porcentaje de consumo respecto a la capacidad de batería (duración de la batería). Esto puede indicar que se presenta una variación en sus resultados debido a que el nivel de batería cambia continuamente; aumenta cuando ha sido cargada y disminuye a medida que se descarga por uso de otras aplicaciones o paso del tiempo.

Si comparamos los resultados de este proyecto con los de la aplicación PowerProfiler (Tuysuz *et al*., 2019), se observa que sus resultados no son constantes debido a que uno de los datos que utilizaron (nivel de batería) son valores continuamente cambiantes. En los resultados obtenidos en este trabajo con la aplicación SENSORTV, se observa que la gráfica tiende a pequeñas variaciones, esto se debe a que se trabajó con el voltaje total de la batería, valor fundamental para obtener la potencia de los sensores en unidades de Miliwatts (Mw), la cual se muestra en tiempo real en la gráfica y en Miliwatts-Minuto en el archivo tipo csv cuando se realiza captura de datos, de esta manera se cumple con los requerimientos funcionales definidos para el proyecto.

Al comparar los resultados de este proyecto con los de la aplicación EM3S (Energy Monitoring System for Smartphones Sensors, Khan *et al*., 2016), se puede observar que sus resultados se reflejan de manera porcentual respecto a la batería, mientras que SENSORTV muestra los resultados de la potencia en unidades de Miliwatts (Mw). También, los autores de EM3S informan que se debe realizar la exportación de los datos a un PC para poder realizar un análisis más avanzado haciendo uso de herramientas estadísticas, es decir que la aplicación EM3S no guarda ningún tipo de información, SENSORTV guarda información de la captura de datos, la cual se visualiza en un archivo de tipo csv, es decir que no se utiliza ningún tipo de conversión de terceros.

[19] https://developer.android.com/guide/topics/sensors/sensors_motion?hl=es
[20] https://developer.android.com/reference/android/hardware/Sensor#isWakeUpSensor()
[21] https://developer.android.com/guide/topics/sensors/sensors_environment?hl=es

6. CONCLUSIONES

Del desarrollo de este trabajo se derivan las siguientes conclusiones:

Se logró cumplir con los objetivos específicos planteados en este proyecto, como se evidencia en la sección Desarrollo, al elaborar cada objetivo se consiguió desarrollar la aplicación SENSORTV, la cual es capaz de monitorear la potencia de los sensores acelerómetro, de proximidad, giroscopio y de luminosidad.

Al utilizar una latencia corta (1-3 segundos) se puede lograr mayor precisión al trabajar tiempo real.

El patrón biométrico no está reconocido por Android como un sensor.

Las lecturas que se han realizado para el GPS no corresponden a estimaciones en tiempo real, por cuanto a que se utilizan valores predeterminados.

Cuando se trabaja con datos de potencia, los valores resultantes en algunos sensores tienden a ser levemente variables, debido a que se utiliza el voltaje total de la batería y la intensidad, en donde se presentan valores con límites de variación.

La aplicación SENSORTV incluye una funcionalidad que permite guardar los datos capturados en un tiempo establecido, lo que permite al usuario visualizar estos datos posteriormente. Esta funcionalidad es una herramienta novedosa y da respuesta a uno de los requerimientos establecidos para este trabajo.

Al trabajar con la metodología SCRUM es necesario que el equipo de desarrollo tenga adaptabilidad al cambio y pensamiento versátil, para manejar los escenarios cambiantes que se presentan.

Trabajo futuro

La aplicación cuenta con una modelación de datos UML como son diagramas de clase y los casos de uso, esta información puede ser útil en futuras investigaciones relacionadas al consumo de energía de los sensores de un teléfono inteligente.

Esta investigación puede ampliarse para incluir compatibilidad para otros dispositivos que funcionen con sistema operativo Android, como relojes y tabletas.

Ampliar su versionamiento a versiones inferiores a Android 8.0 (Oreo)

Incluir una compatibilidad para que pueda ser aplicado en otros sistemas operativos como iOS y Windows Phone.

Este desarrollo puede ser utilizado como referencia para otras investigaciones similares que se apliquen al campo de la ingeniería.

Esta investigación puede ser utilizada para complementar otras aplicaciones que requieran monitorizar el consumo de energía de uno o varios sensores específicamente.

7. GLOSARIO DE TÉRMINOS

Captura: monitorear y guardar el consumo de energía de uno o varios sensores.

Clase abstracta: clase que no se puede instanciar. Su nombre es reservado por lo tanto no es posible utilizar el nombre de esta clase para declarar otras variables.

Directorio: contenedor virtual en donde se almacenan subdirectorios y/o diferentes archivos.

Fichero: archivo almacenado en el dispositivo.

Latencia: ciclos de muestreo cada 200 Milisegundos (ms).

8. BIBLIOGRAFÍA

Admin. (2019). Intel® Software. Retrieved from https://software.intel.com/

Android Developers. (n.d.). Retrieved from https://developer.android.com/

Boillot M.A. (2012). Patente de Estados Unidos Nº 8.312.479 . Washington, DC: Oficina de Patentes y Marcas de los Estados Unidos.

Borja C.T. & Bueno A.G. (2006). Sistemas Biométricos. *Recopilado de: https://www. dsi. uclm. es/personal/MiguelFGraciani/mikicurri/Docencia/Bioinformatica/web_BIO/Documentacion/Trabajos/Biometria/Trabajo% 20Biometria. pdf*.

Carletti E.J. (online). Sensores - Conceptos generales. Descripción y funcionamiento. Disponible en http://robots-argentina.com.ar/Sensores_general.htm

Carroll A. & Heiser G. An Analysis of Power Consumption in a Smartphone. En USENIX annual technical conference. 2010.

Congreso de la República, Ley 23 de 1982 disponible en http://derechodeautor.gov.co/documents/10181/182597/23.pdf/a97b8750-8451-4529-ab87-bb82160dd226 (acceso octubre 2019).

Congreso de la República, Decisión Andina 351 de 1993 disponible en http://derechodeautor.gov.co/decision-andina (acceso octubre 2019).

Congreso de la República, Ley 1915 de 2018 disponible en http://es.presidencia.gov.co/normativa/normativa/LEY%201915%20DEL%2012%20DE%20JULIO%20DE%202018.pdf (acceso octubre 2019).

Congreso de la República, Ley Estatutaria 1581 de 2012 disponible en http://www.secretariasenado.gov.co/senado/basedoc/ley_1581_2012.html (acceso octubre 2019).

Congreso de la República, Decreto número 1317 de 2013 disponible en https://www.mintic.gov.co/portal/604/articles-4274_documento.pdf (acceso octubre 2019).

Corral L., Georgiev A.B., Sillitti A., Succi G. 2013. A Method for Characterizing Energy Consumption in Android Smartphones. 2nd International Workshop on Green and Sustainable Software (GREENS). IEEE. 38-45 pp. Doi 10.1109/GREENS.2013.6606420

Couto M., Cunha J., Fernandes J.P., Pereira R. & Saraiva J. (2015). GreenDroid: A Tool for Analysing Power Consumption in the Android Ecosystem. IEEE 13th International Scientific Conference on Informatics: 73-78. Slovakia.

Gouveia, R. (online). Ley de Ohm. Disponible en https://www.todamateria.com/ley-de-ohm/

Jiménez J. & Stalin J. (2016). *Monitorización distribuida en tiempo real de monóxido de carbono a través de smartphones* (Bachelor's thesis, Quito: Universidad de las Américas, 2016.).

Khan I., Khusro S., Ali S. & Ahmad J. (2016). Sensors are Power Hungry: An Investigation of Smartphone Sensors Impact on Battery Power from Lifelogging Perspective. Bahria University Journal of Information & Communication Technologies 9(2): 8-19.

Laínez J.R. (2015). Desarrollo de software ágil. Extreme programming y Scrum: 137-143 pp. IT Campus Academy.

Le H., Crossley J., Sanders SR & Alon E. (2013). Un regulador de voltaje de condensador conmutado conectado a la batería totalmente integrado con respuesta sub-ns que entrega 0.19W / mm2 con un 73% de eficiencia: 372-373 pp. Conferencia Internacional de Circuitos de Estado Sólido IEEE Recopilación de documentos técnicos, San Francisco, CA.

Miao (2019). Intel® Software. Retrieved from https://software.intel.com

Ramírez J.C. (in prep.). An app for the real-time monitoring of energy consumption in Android.

Romero G. & Maskrey A. (1983). Documento de Estudio No. 1. Como entender los desastres naturales: 1-7 pp. PREDES.

Rubio F.G. (2015). Sensorconomy: Gestión de los sensores de un smartphone para la creación de contenidos audiovisuales. *El nuevo diálogo social: Organizaciones, Públicos y Ciudadanos*, 437-450.

Serway R. A. & Jewett, J. W. (2009). Corriente y resistencia: 752-774 pp. En: Física para ciencias e ingeniería con Física Moderna. Volumen 2 séptima edición. Editorial CENGAGE Learning.

Tawalbeh M., Eardley A. & Tawalbeh L. (2016). Studying the Energy Consumption in Mobile Devices. The 13th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2016). Procedia Computer Science 94: 183 – 189.

Trigas Gallego M. (2012). Metodología scrum. 55 pp.

Tuysuz M.F., Ucan M. & Trestian R. (2019). A real-time power monitoring and energy-efficient network/interface selection tool for android smartphones. Journal of Network and Computer Applications 127: 107–121.

Wong K.Y. (2010). Cell phones as mobile computing devices. IT professional, pp. 40-45.

Zetina-M., A. & Zetina-C., A. (2004). Ley de Watt: 313-316 pp. En: Electrónica básica. Editorial Limusa.

Zhang L., Tiwana B., Qian Z., Wang Z., Dick R.P., Mao Z.M., & Yang L. (2010). Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In Proceedings of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis: 105-114. ACM.

Urls: Documentación oficial de Android disponible en https://developer.android.com/
Documentación de Android para la clase BiometricManager disponible en https://developer.android.com/reference/android/hardware/biometrics/BiometricManager
Documentación de Android sobre la guía de los sensores disponible en https://developer.android.com/guide/topics/sensors/sensors_overview
Documentación de Android para la clase BatteryManager disponible en https://developer.android.com/reference/android/os/BatteryManager
Documentación de Android para la clase SensorEvent disponible en https://developer.android.com/reference/android/hardware/SensorEvent
Documentación de Android para la interfaz SensorEventListener disponible en https://developer.android.com/reference/android/hardware/SensorEventListener
Documentación de Android para la clase SensorManager disponible en https://developer.android.com/reference/android/hardware/SensorManager
Documentación de Android sobre almacenamiento de archivos disponible en https://developer.android.com/training/data-storage/files?hl=es-419#WriteExternalStorage
Documentación de Android sobre sensores de posición disponible en https://developer.android.com/guide/topics/sensors/sensors_position?hl=es
Documentación de Android sobre sensores de activación (wake-up sensors) disponible en https://developer.android.com/reference/android/hardware/Sensor#TYPE_PROXIMITY, https://developer.android.com/reference/android/hardware/Sensor#isWakeUpSensor()
Documentación de Android sobre sensores de movimiento disponible en https://developer.android.com/guide/topics/sensors/sensors_motion?hl=es
Documentación de Android sobre sensores de ambiente disponible en https://developer.android.com/guide/topics/sensors/sensors_environment?hl=es
Documentación de Android sobre eventos de sensores disponible en https://developer.android.com/guide/topics/sensors/sensors_overview?hl=es#sensors-monitor