ANDROID BATTERY SAVER

BY

CHINONSO AQUINAS JAMES

MATRIC NO: 18/4811

A PROJECT WRITTEN AND SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE, COLLEGE OF PURE AND APPLIED SCIENCES (COPAS), IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF BACHELOR OF SCIENCE (B.Sc.) DEGREE IN COMPUTER SCIENCE OF CALEB UNIVERSITY, LAGOS.

JULY, 2021

DECLARATION

I, **CHINONSO AQUINAS JAMES**, do hereby declare that this project is entirely my work and composition. The work embodied in this project has not been submitted in candidature for any degree and is not concurrently being submitted for any other degree. All references made to works of other persons have been duly acknowledged.

Signature……

Date……

CERTIFICATION

We certify that this research work was carried out by **CHINONSO AQUINAS JAMES** in the Department of Computer Science, College of Pure and Applied Sciences, Caleb University, Lagos. The research work is considered adequate in partial fulfilment of the requirements for the award of Bachelor of Science in Computer Science.

_____ _____

Dr. Bukola Olanipekun **Date**

Project Supervisor

_____ _____

Dr. Olumoye Mosud **Date**

Head of Department

_____ _____

Dr Olutola Bob-Soile **Date**

Dean of COPAS

_____ _____

External Examiner **Date**

DEDICATION

I would like to dedicate this project to the Almighty God for grace wisdom, understanding, and the strength he gave me during this period and also to my parents for their support physically, emotionally and financially all through my project.

ABSTRACT

The saving power of Android enabled devices have become a significant issue with 400,000 such devices being activated daily. Android smartphones and tablets offer several power-hungry hardware components and the app developers are exploiting these components at disposal to provide revolutionary user experience. But the battery life has not increased at the same pace to support the power demand. Thus, many projects have been carried out to investigate how to minimize the power consumption in smartphones. This research reports four different project's themes towards the reduction of smartphone power consumption. Efforts have been made to survey Android power saving apps available in Google play Apps store as the basis to find out different power saving approaches, operations and limitations. Then I present four different avenues to prolong the batter life of Android devices. The first two approaches include usage pattern analysis to generate power saving profiles. The advantage being that the power saving profiles are customized to actual user behavior. Integrating a photovoltaic film on top of the smartphone and tablet touch screen to generate electricity is also mentioned. The usage pattern-based power saving profile generation has several privacy concerns which are discussed and countermeasures are proposed. Some emerging security attacks are also briefed.

Keywords - Android power management; usage pattern; SetCPU; JuiceDefender; power monitoring; privacy; security attack.

TABLE OF CONTENT

Title Page	**i**
Declaration	**ii**
Certification	**iii**
Dedication	**iv**
Abstract	**v**
List of Tables	**ix**
List of Figures	**x**
CHAPTER ONE: INTRODUCTION	**1**
1.1 Background of the Study	**1**
1.2 Statement of the Problem	**2**
1.3 Aim and Objectives of the Project	**2**
1.4 Scope of The Study	**3**
1.5 Limitation of the Study	**3**
1.6 Definition of Terms	**3**
CHAPTER TWO: LITERATURE REVIEW	**5**
2.1 Battery Life	**5**
2.1.1 Pathak Et Al	**6**
2.1.2 Xu Et Al	**6**
2.1.3 Woo Et Al	**8**
2.2 Battery Saver Mode	**11**
2.3 Job Scheduling API	**12**

2.4 Battery Application Review **12**

CHAPTER THREE: SYSTEM ANALYSIS AND DESIGN **16**

3.1 Research Methodology **16**

3.2 Observation and Interview Carried Out **16**

3.3 Analysis of System and Use Case Diagram **17**

3.3.1 Sources of Data **18**

3.3.2 System Architecture **19**

3.4 Problem Definition **20**

3.4.1 Accurately Power Consumption of Framework API Calls **21**

3.4.2 Accurately Power Consumption of Sensor Usage Architectures **21**

3.4.3 Accurately Effects of Different Communication Protocols **21**

3.5 Programming Language **22**

3.5.1 Program Module Specifications **23**

3.6 Requirement Specifications **25**

3.6.1 Choice of Programming Language **25**

3.7 Implementation **25**

CHAPTER FOUR: SYSTEM IMPLEMENTATION AND RESULTS **27**

4.1 Scope of Design **27**

4.2 System Design **27**

4.3 Input/ Output Specification and Design **28**

4.4 System Requirement **28**

4.4.1 Hardware Requirements **28**

4.4.2 Software Requirements 29

4.5 Program Design 30

4.5.1 Program Testing and Debugging 30

4.5.2 Program Documentation 30

4.6 Training of Staff / Users 31

4.7 Change over Procedure 31

4.7.1 Pilot Change Over 31

4.7.2 Direct Change Over 31

4.7.3 Parallel Change Over 32

4.8 Documentation 32

CHAPTER FIVE: SUMMARY, CONCLUSION AND RECOMMENDATION 33

5.1 Summary 33

5.2 Conclusion 33

5.3 Recommendation 34

REFERENCES 35

LIST OF TABLES

2.1 List of Test Cases Performed by Each Device **10**

4.1 Input Specifications **28**

4.2 Output Specifications **28**

LIST OF FIGURES

2.1 Energy Savings From Different Email Size and Inbox Size **8**

2.2 Battery Saver Feature on the Nexus 5 **12**

2.3 Home Page of Battery Saver **13**

2.4 Settings Page of Battery Saver **14**

3.1 System Architecture for Battery Saver Application **19**

3.2 Program Diagram **23**

3.3 System flowchart **26**

CHAPTER ONE

INTRODUCTION

1.1 Background of The Study

Smartphone Battery is inherently limited and the applications that consume more power are increasing every day, it is known that most of cell phones application rarely useful if cell phone is not connected to certain network services and saving battery trade off tends against power saving. Currently many applications installed in Smartphone use technologies that may consume battery power quickly, those applications use technology such as GPS, 3G, 3GS and 4G is considered as power drain. A list of applications and services that might consume power such as auto-synch applications, 3G, 4G, GPS, and Wi-Fi can be found in Smartphone. Android system provides an interface where user can see where battery power goes; this interface is located in "Battery use" within about phone panel of android system.

However, there are tools that provide more information in graphical and visualized form such as power Tutor (powerTutor.org). Power Tutor provide more details about power consuming for power consuming of specific hardware or application in real-time or within certain period of time. Those tools show that hardware such as GPS and Wi-Fi consume more power even though the Smartphone is idle.

Most of the end users have difficulty determining which application designers have an incentive to develop energy efficient smartphones software. Their main barrier is the difficulty of determining the impact of software design decisions on system energy consumption, but that barrier can be overcome.

Many papers have mentioned that the energy consumption has become an important problem in energy management of mobile phones and have their own ways to save energy. We should first know

the energy consumption of the applications on mobile phones. Monitoring the energy consumption of smartphone is very important for saving energy to extend the lifetime of battery.

Battery Viewer that includes both the hardware and software setup. Along with the observations from energy profiling of a mobile phones. The main focus on understanding the energy consumption of real time applications. The most real time applications that are consuming more energy as comparative to other real time applications like Wi-Fi, GPS, LCD, AUDIO, VIDEO and CPU. Our Battery Viewer will generate a Power Saving Profile According to the usage of users.

1.2 Statement of The Problem

The apps are designed to clear the recent apps/running apps that stops using the CPU and thus saving the battery. But then when you open the app again, your phone has to load each and every resource needed to run the app again.

Secondly, the app itself keeps running in background which drains the battery as the phone needs to process data for the app.

Thirdly, the app itself drains your battery percentage dramatically as it run in the background for 24/7 in order to perform app killing process continuously.

1.3 Aim and Objective of The Study

The aim of this study is to develop an android battery saving application that would help users minimize drainage of battery on their android devices. The following objectives have been set in order to accomplish the stated objective:

Analyse the existing battery saving applications.

Develop a battery saving application.

Implementation of the Android battery saver.

Evaluation of the new system.

1.4 Scope of The Study

The app is designed to clear the recent apps/running that stops using the CPU and thus saving the battery. This system uses Android studio as its front end and doesn't use any backend as this type of application doesn't need one since it uses the data from the phone itself and projects to the user.

1.5 Limitation of The Study

The system provides with less information than the phones build in app.

The heavier tasks that need the CPU power is not available in the battery saver app, because they continue to run in background even after you remove them from the recent apps they often requires administrator permission to run.

1.6 Definition of Terms

Battery Life: is pioneer compared to all other battery analysis tools from the App Store. The application will display runtimes, various internal as well as external battery data and keep you informed about the corresponding battery charges.

Smartphones: A smartphone is a mobile device that combines cellular and mobile computing functions into one unit.

Android: Android is a mobile operating system based on a modified version of the Linux kernel and other open-source software, designed primarily for touchscreen mobile devices such as smartphones and tablets.

Application: is a program or group of programs designed for end users. Examples of an application include a word processor, a spreadsheet, an accounting application, a web browser, an email client, a media player, a file viewer, simulators, a console game or a photo editor.

CHAPTER TWO

LITERATURE REVIEW

2.1 Battery Life

One of the most prominent issues with smartphones is battery life, with 37% of user stating it is their biggest problem. As more powerful smartphones are developed, concerns with battery life increase. Users should be able to utilize their device as they wish for a full day before a recharge is required. However, this is often not the case, leading to a change in our activities to preserve battery life. This concern was not evident on desktop computers, as they have a constant source of power. With the rise of smartphone services, it is one of the biggest challenges faced by developers. While research into more efficient batteries is possible, another area of research focuses on improving the efficiency of applications.

For software developers, the solution to preserving battery life is dependent on the efficiency of the application. Each application may have different shortcomings that cause this, varying for each case. However, consistent problems that can affect many applications are no sleep energy bugs. Pathak et al. define no-sleep bugs as energy consuming errors that stem from mismanagement of power control APIs. The components of a smartphone are either off or idle, unless an application explicitly instructs it to remain on. The resulting process requires developers to constantly enable and disable components when developing their applications. This ultimately leads to errors when a component should be disabled but is not turned off. The smartphone's battery will be depleted at an increased rate, unnecessarily powering a component.

A variety of no-sleep bugs have been recorded and categorized into three groups. Pathak et al. note that their list is not definitive, and more bugs can exist. No-sleep code paths define code paths in an

application that wake the component, but do not release it after use. This represents the majority of known no-sleep bugs from the findings of Pathak et al. No-sleep race condition occurs in multi-threaded applications, where one thread switches the component on, and another switches it off. Lastly, no-sleep dilation bugs occur when the awoken component is intended to be put to sleep, but the time required to do so is unnecessarily long.

2.1.1 Pathak Et Al

The solution proposed by Pathak et al. is to create a compile-time dataflow analysis solution that can detect no-sleep energy bugs. Dataflow analysis is defined as a set of techniques that analyze the effects of program properties throughout a given program, managed within a control flow graph. Their solution focuses on the sections where smartphone component power is managed. If all of those sections have end points that turn off the components, the program is free of no-sleep bugs. To test their application, they ran their analysis on 86 different android applications. In addition to the 12 known energy bugs detected, 30 new types of bugs were discovered. Pathak et al. note that this area of research is relatively new, and they are making the first advances towards understanding and detecting no sleep bugs.

2.1.2 Xu Et Al

Focusing on a specific type of application, Xu et al (2014) examined the built-in email clients of Windows Phone and Android to determine areas of improvement. Windows phone uses Microsoft Exchange, while Android uses Gmail. Gmail is one of the most popular applications on a mobile device, with over 50 million users per month. With such a large user base, it is important that Gmail and other email applications are optimized for functionality, accessibility, and power consumption. Unfortunately, functionality and power consumption can be contradicting. Functionality requires the

application to be constantly checking for new messages, but continually syncing is extremely resource intensive. Finding a balance between these two concerns is not only limited to email, and can be practical for other applications.

(Xu et al 2014) findings outlined five distinct areas that required improvement. The first improvement was reducing the 3G tail time. The tail time is a standby period during data transmission where the device waits for more data before ending a connection. The purpose of tail time is to avoid ending and restarting a connection, which is energy inefficient. However, events received by email are so infrequent, that this process ends up using more energy on average. The second improvement was to decouple data transmission from data processing. The current method will process the current data before receiving the next transmission. Network communication remains open during this time, leading to a waste of energy alongside the 3G tail effect. While this is not an easy process, retrieving all of the transmissions first and closing the connection eliminates the stated problems. The third improvement is to batch data processing requests. As multiple small writes to flash storage is slow and energy inefficient, it is beneficial to batch these requests and process them together. The fourth improvement is to reuse existing network connections to receive emails. Current implementations make it easy and natural to create a new connection for each new email received, but the energy costs are not negligible. The fifth improvement is partitioning the inbox. The energy cost of receiving an email increase when the inbox is larger, and can be attributed to the time is takes to update the metadata. The proposed solution is to partition the inbox into two parts: a small inbox for recently received emails, and a large one for the remainder. As most new messages will interact with recent messages, there is no need to search through old emails. Xu et al. implemented these changes and proceeded to observe the change in energy consumption. Their findings indicated an average energy reduction of 49.9%.

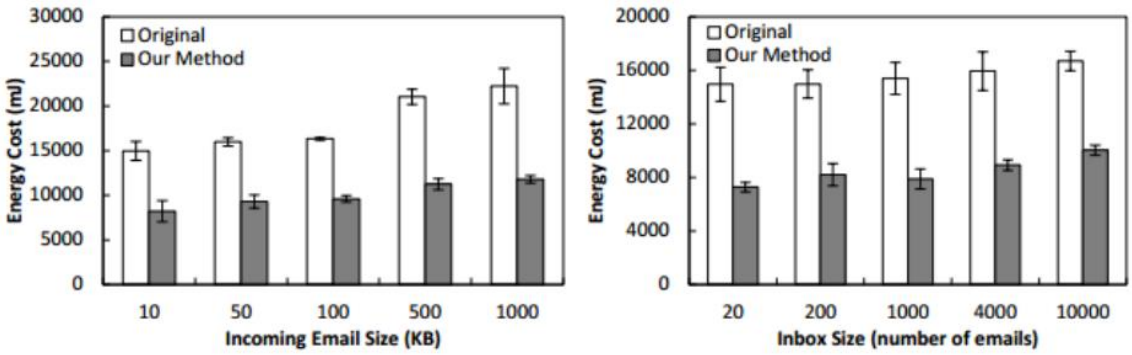

Figure 2.1: Energy saving with different email sizes (left) and energy saving with different inbox sizes (right).

Beyond applications, web browsing can also have a large impact on energy consumption. Most modern webpages are populated with a variety of detail beyond traditional text. Pictures, videos, and animations are placed throughout the site, which require significantly more resources to generate in terms of both bandwidth and energy. This is evident in, a study on the energy and bandwidth costs of web advertisements on smartphones. Another study examines and characterizes resource usage for web browsing as a whole.

2.1.3 Woo Et Al

(Woo et al 2015) examine caching in their study, as minimal work has been done to optimize the content caching in cellular networks. The increasing number of high-speed base stations has made network accessibility more convenient for users. However, the problem surfacing with cellular networks is that all user traffic has to pass a limited number of gateways at core networks before reaching the wired internet. Simply increasing the physical backhaul bandwidth is not feasible for centralized architectures such as this. To circumvent this, optimization strategies must be considered. Their study focuses on three types of caching: conventional web caching, prefix-based web caching,

and TCP-level redundancy elimination. Conventional web caching places information at the Digital Unit Aggregation (DUA) component of the cellular network architecture. However, this approach suffers from two problems. The first problem is that it "…cannot suppress duplicate objects that are uncatchable or that have different URLs (i.e., aliases)". Secondly, it is difficult to handle handovers from the mobile device to the DUA while the content is being delivered. Prefix-based web caching can overcome the first problem that web caching has, suppressing the duplicate and aliased objects. The drawback of this approach is that it's efficiency and rate of false positives was initially unknown, but is addressed by Woo et al. later in the study. Lastly, TCP redundancy elimination can also handle the issues of traditional web caching, but suffers from a complex implementation and high computational overhead.

The first part of the study was to analyse the TCP and application-level characteristics of the traffic, while the second part was comparing the effectiveness of the three types of web caching. Based on the results, 59.4% of the traffic is redundant with TCP-level redundancy elimination if we have infinite cache. In regards to caching options, standard web caching only achieved 21.0-27.1% bandwidth savings with infinite cache, while prefix-based web caching produced 22.4-34.0% bandwidth savings with infinite cache. In addition, TCP-RE achieved the highest bandwidth savings of 26.9-42.0% with only 512 GB of memory cache.

(Li et al 2014) perform an analysis of energy consumption on android smartphones, focusing on how the device is used as opposed to the applications running. To maintain consistency, an additional battery with a fixed voltage source is attached to a smartphone instead of using the traditional lithium-ion battery. A series of test cases are then performed on three different Android devices, and the electric current is measured with a multimeter. In each device's test case, the smartphone was set to 50% brightness, and all applications were closed.

Table 2.1: List of test cases performed by each device. Power consumption values for each scenario are collected.

Test Case
50% brightness screen
Opening GPS
Opening Wi-Fi
Wi-Fi Download (2.55 Mpbs)
GSM Download (35kpbs)
Opening Bluetooth
Bluetooth Searching Devices
Bluetooth sending data
CPU Single thread
CPU multithreads
CPU stress condition
Opening terminal
Calling
Incoming call
Sending a message
Taking a picture
Playing music
Playing video

The power consumption of each test case is recorded, and the results are analysed. In addition, additional test cases are performed with varying screen brightness. With the data collected, energy consumption models for screen brightness are provided. However, models for the other test case modules such as CPU, GPS, Wi-Fi and Bluetooth were unable to be determined. As each module has a variety of states which were rapidly changing, an accurate model could not be calculated for each case. Instead, a general function is provided to approximate the power consumption of any given state. (Hoque at al 2015) present an analysis of the battery in order to examine charging mechanisms, state of charge estimation techniques, battery properties, and the charging behavior of both devices and users, using data collected from the Carat application. Carat is an application that tracks the applications you're using, but does not measure energy consumption directly. The first analysis examines the charging techniques of smartphones and the charging rates. The charging mechanisms, battery voltage and charging rates of the devices are outlined. In addition, two additional charging mechanisms that are variants of the established CC-CV and DLC methods are identified. The second part of the analysis examines battery properties such as the changes in its capacity, temperature when charging, and battery health.

The results indicated a linear relationship between the remaining battery capacity and final voltage, and a decrease in battery temperature over time as the device charged. In addition, the health of the battery did not indicate increases in battery temperature.

2.2 Battery Saver Mode

Android 5.0 also builds in a power saving mode called Battery Saver. This feature is turned on automatically once the battery drops below a threshold value. The saver lowers the processor's speed, reduces animations, dims the screen, and reduces radio usage.

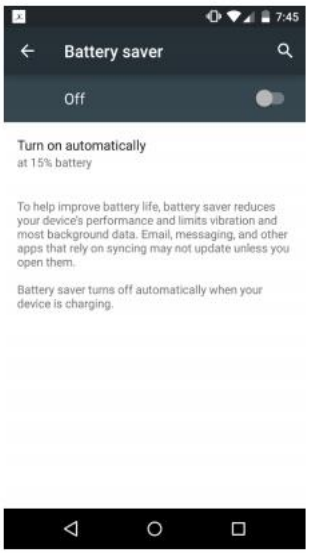

Figure 2.2: Battery Saver feature on the Nexus 5

2.3 Job Scheduler API

In Lollipop, a new API was provided to enable developers to define conditions for background jobs for the system to perform. This allows the OS to batch together multiple tasks and run them together when those conditions are met. This prevents the processor from being woken up in the middle of its sleep for something that can be safely delayed for later, thus reducing overall power consumption. The scheduler is also smart enough not to allow tasks to run when the resource required isn't available, like network updates when there is no network.

2.4 Battery Application Review

The purpose of reviewing the current battery management applications was to determine the functionality that is currently offered, and to check that prediction is not an established approach. DU Battery Saver, Battery Doctor, and Battery Saver applications were retrieved from the Google Play Store, using the search tag battery saver and battery life. They were the highest rated applications, top results from searches, and had a minimum rating of 4.5/5.0. In addition, DU Battery Saver and Battery

Doctor have over 8 million downloads respectively as of May 2018. Battery Saver is no longer available in the Google Play Store as of May 2018, and the number of downloads was not recorded. The DU Battery Saver offers a significant amount of functionality, providing a main page showing the battery percentage, battery remaining, and the temperature of the device as shown in Figure 3. The fix now feature will cause the device to close unused applications that are draining resources to extend the battery life. Within the main page, the mode option allows the user to change the current profile. The smart button reveals a set of options that determine which applications are needlessly using resources, freeing them up to conserve battery. Included here is a whitelist of applications that won't be terminated. A list of profiles is also available, altering the functionality of the device based on the user's needs. A table of switches is provided to quickly enable/disable features such as Wi-Fi, data, display brightness, and ringtones. In addition, the settings page in Figure 5 provides a variety of reminder features. Alongside these features, the application is also visually appealing as well. Icons are used within the main menu, and animations are provided when it is scanning for applications that are using resources.

Figure 2.3: Home page of Battery Saver

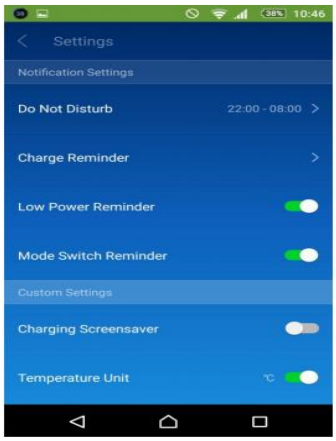

Figure 2.4: Settings page of Battery Saver

The issue with this application comes from the boost and toolbox options on the main menu. The toolbox is a list of advertisements, while boost claims that there is trash on the device, and advertises for another application. While many of the features on the device are beneficial, these components are obstructive and unnecessary. In addition, notifications advertising the other applications were also periodically appearing. Lastly, no prediction-based functions were observed on the application.

The Battery Doctor application lacks the design appeal of DU Battery Saver, and provides similar features. The application monitors the battery consumption, and notifies the user when background applications are consuming excessive battery life. A highlight of application battery usage, power remaining, battery level history, and device temperature are also provided on the main page. Tabs on the bottom of the application lead to charging history, battery profiles, and a consumption page of the applications that are running. A settings page also provided a low power notifications, Wi-Fi toggling, and an ignore list. However, beyond these features, the application was not as appealing as The Battery Doctor. The application would constantly note that the battery was draining fast, as shown in Figure 6, even though the optimize now button was recently used. However, the biggest issue was the amount of advertisements throughout the entire application. In some cases, they blended in with some

of the features, which could confuse users. Figure 7 provides an example of how intrusive the ads on the application were.

CHAPTER THREE

SYSTEM ANALYSIS AND DESIGN

3.1 Research Methodology

The Hidden Markov Model (HMM) can be applied to a system which aims to recognize user states. In this system, Android app battery state (i.e., extracted user contexts through mobile device-based sensors) are seen as inputs. These readings undergo a series of signal processing operations and eventually end up with a classification algorithm in order to provide desirable inferences about user behaviors/profiles for context-aware applications. A required classification algorithm differs in terms of explanation of extracted user context through a specific sensor.

Classification algorithms produce observations (i.e., visible states), ϑ, of HMM. Among observations, only one observation is expected to provide the most likely differentiation in selection of instant user state representation. This observation is marked as instant observation, which also indicates the most recent element of observation sequence of HMM. On the other hand, user states are defined as hidden states, S, of HMM since they are not directly observable but only reachable over visible states. Therefore, each observation has cross probabilities to point any user state. These cross probabilities build an emission matrix, bjk, which basically defines decision probabilities of picking user states from available observations. In addition, a user state might not be stationary since a general user behaviour changes in time. Thus, it is expected from a user state either to transit into another user state or to remain same. These occurrences build a time-variant user state transition matrix, aij, which defines transition probabilities among the user states.

.

3.2 Observation and Interview Carried Out

The main method of data gathering in this research is by internet surfing; I carried out my research on the internet by searching on how GPS, Bluetooth, internet, Brightness etc power consumption can be minimized. Also, I did research on how to implement an Artificial Intelligence method to the app to know when to put off the on the Power saver if the phone is running hot. I also carried out interview by visiting some programming companies to gather primary data.

3.3 Analysis of System and Use Case Diagram

This covers the core of the system functionality and the proposing use case of the system. However, I have detailed some information gathered from the developed application.

• Memory consumers are implemented by dynamically allocating custom objects that wrap byte arrays. To analyze the frequency of garbage collection, a Java WeakReference object is used to inform the garbage collector that they can be reclaimed, despite having active references within running code. The object's finalize() method (which is called immediately before the object is reclaimed by the Android Dalvik virtual machine) is overridden to record the time of garbage collection, thereby allowing developers to analyze the frequency of garbage collection runs. The WeakReference object will thus be reclaimed during every garbage collection run. Due to the limitations of the Android instrumentation API, garbage collection and memory usage must be inferred through analysis of the frequency and duration of garbage collection runs, rather than through direct power consumption measurement. Although this limitation prevents developers from including memory statistics in the data along with CPU, sensor, and network utilization, they can still examine how their design uses memory. Users can also configure the amount of memory and frequency of allocation, as well as supply custom objects (such as WreckWatch's image caches) to use rather than the byte arrays used by default.

• GPS consumers are implemented by code that registers to receive location updates from the GPS receiver at specific intervals.

• Accelerometer consumers are implemented using the configuration specified in the XML file, along with generic setup code to establish a connection to the appropriate hardware.

• Network consumers are implemented as emulation code containing a timer that executes a given networking operation, such as an HTTP operation at a user defined interval.

• Screen drawing agents allow users to specify 3D and 2D graphics to draw on the screen. Developers will specify object contents along with any potential motion or actions.

• Custom code modules allow developers to supply their own code blocks to extend the functionality of SPOT and enhance emulation accuracy as the development cycle progresses by substituting the faux emulation code with actual application logic. SPOT allows developers to supply class files to load into the emulation application dynamically, as well as method “hooks” to allow the emulation code to interact with the custom code properly.

3.3.1 Sources of Data

The data that were collected in this research were from the primary sources. Observations were made while gathering my data on operations Vehicle breakdown assistance finder, I did research on the most suitable programming language to use. Other sources of data were of the secondary source, which includes gathering information from journals and magazines, materials from internet, Microsoft Encarta premium, seminars, lectures and personal researches.

3.3.2 System Architecture

To analyse the energy consumption of the application on mobile phones, we designed Battery Viewer System. First, we will start the application. Second it will check the energy consumption of the application on mobile devices according to the data it collects. Then third, it will show on the screen according to the select of tab that will show on the application. Last one is it shows the graphical view of energy consumption of applications. Battery Viewer is used to check the information of the battery and application, especially the energy consumption information. It draws the real time graphical view of energy consumption

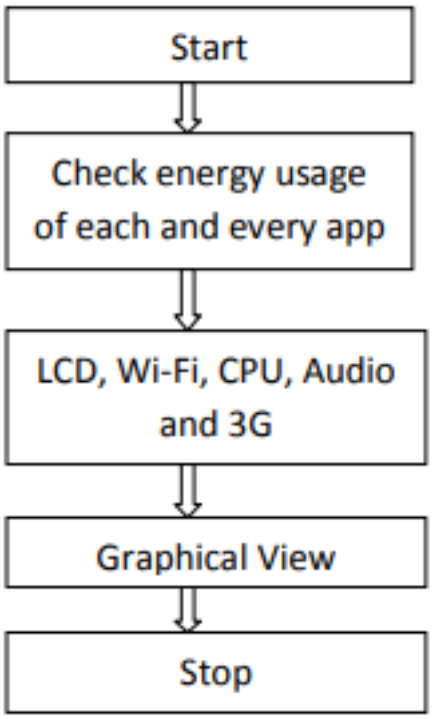

Figure 3.1: System architecture for battery saver application

(Software Development Kit) by using Eclipse IDE (Integrated Development Environment) with the ADT (Android Development Tools). First, we implemented a data collection application for analyzing user usage pattern. Our application was written in java and runs on any Android based platform. It

periodically records the following information to a log file. Currently it is scheduled to store the information every second. 1. Screen Status: -Whether the screen status is in the “ON” state or “OFF” state. 2. Wi-Fi Status: -Whether the Wi-Fi status is in the “ON” state or “OFF” state. 3. GPS, 3G and Audio Status: -Whether the GPS, 3G and Audio status is in the “ON” state or “OFF” state..

3.4 Problem Definition

3.4.1 Accurately Predicting Power Consumption of Framework API Calls

Each line of code executed results in a specific amount of power consumed by the hardware. In the simplest case, this power consumption results from a small series of CPU operations, such as reading from memory or adding numbers. In some cases, however, a single line of code can result in a chain of hardware interactions, such as activation of the GPS receiver and increasing the rate at which the screen is redrawn. Moreover, although the higher levels of abstraction provided by modern smartphone SDKs make it easier for developers to implement mobile application software, they also complicate predictions of the effects on the hardware.

For example, WreckWatch heavily utilizes the Google Maps API and the “MyLocation” map overlay, which provides end users with a marker indicating their current GPS location. The use of the “MyLocation” is typically accomplished with fewer than 10 lines of code, but results in substantial power consumption. This is because the overlay is redrawn at a high rate to achieve a “flashing” effect, and because the overlay enables and heavily utilizes the GPS receiver on the device, which further increases power expenditure. It is hard to predict how using arbitrary API calls, such as this overlay, will affect application power consumption without implementing a particular design and testing it on a particular target device. This abstraction in code makes power consumption analysis on

arbitrary segments of code hard. Predicting power usage from high-level design abstractions, such as a UML diagram, is even harder.

3.4.2 Accurately Predicting Power Consumption of Sensor Usage Architectures

In applications utilizing sensor data, the most accurate sensor data is obtained by sampling as often as possible. Sampling at high rates, however, incurs high power consumption (Krause et al., 2005) by not allowing sensors to enter low power modes and by increasing the amount of data processed by applications. Reducing the sample rate can decrease application power consumption considerably, but also reduces accuracy. The trade-offs between power consumption and accuracy at a given sampling rate are hard to determine without empirical tests on a target device due to the high-degree of layering in modern smartphone APIs and system architectures. For example, WreckWatch was originally designed to collect GPS data every 500 milliseconds and consume accelerometer data at Android's predefined NORMAL rate setting. During the development of WreckWatch, it was clear that reducing WreckWatch's GPS sampling rate would reduce overall power consumption, but it was unclear to what degree. Moreover, it was hard to predict what sample rate would provide sufficient accuracy and still allow the phone to operate for days between charges. Section 4 describes how we use automatic code generation to create emulated applications that accurately analyze the power consumption of a candidate sensor utilization architecture without incurring the substantial time and effort to manually implement the architecture.

3.4.3 Accurately Assessing the Effects of Different Communication Protocols

Each application and network communication protocol has a specific overhead associated with it and can result in significant power consumption (Heinzelman et al., 2000). Certain protocols require more development overhead to implement, but have low runtime overhead (e.g. bandwidth consumption,

message processing time, etc.). It is hard to determine early (e.g., at design time) in an applications lifecycle, however, how this overhead will affect power consumption and whether the number of messages transmitted will be substantial enough to impact power consumption significantly. This challenge is exacerbated if certain network operations consume more power than others, e.g., receiving data often consumes more power than transmitting data (Wang et al., 2006). For example, to provide the most accurate situational awareness to first responders—and provide the most accurate congestion information to motorists— the WreckWatch application must periodically request wreck information from the central web server. These updates must be done periodically and were originally intended to run over HTTP. Using HTTP results in a significantly less developer effort but results in a considerable amount of communication overhead from the underlying TCP and HTTP protocols, which ultimately transmits substantial amounts of data that have no relevance to the application. It is hard to determine at design time if/how this additional data transmission will significantly impact power consumption. Section 4 shows how we used MDE code generation to implement and analyze potential communication protocols rapidly.

3.5 Programming Language

Android Studio is the official integrated development environment for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development.

USER INTERFACE DESIGN: It is a system that permits the interaction between human beings and the computer. The project has a user interface design because of the level of interaction where the input and output of the system will be carried out.

PROCEDURAL DESIGN: This involves the design of an efficient algorithm that will satisfy the functional description of the various sub systems.

3.5.1 Program Module Specifications

These are the program module:

Homepage –

Registration page

Login page

Synchronize phone to app

On Battery saver mode

Apply

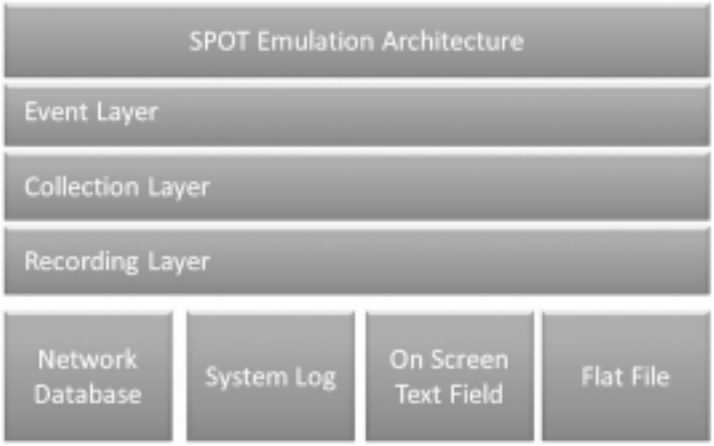

Figure 3.2: Program Diagram

The instrumentation system are either Collectors, Recorders, or Event Handlers. Collectors interface directly with the specific power API on the system and pass collected data to Recorders, which persist the data collected by Collectors by writing it to a file or other storage medium. Event Handlers respond to the events fired by entering or leaving emulation code blocks. These components are dynamically loaded via Java reflection to ensure extensibility and flexibility. For instance, developers can implement a custom Collector to monitor which components are in memory at any given time.

Alternatively, developers could define Recorders to log power consumption information to another data storage medium, such as a local or network database rather than a flat file. To analyze an architecture effectively, SPOT records battery state information over time to allow developers to pinpoint specific locations in their application's execution path that demand the most power. To collect power consumption information accurately, SPOT uses an event-driven architecture that precisely identifies the occurrence of each major application-state change, such as registering or unregistering a GPS listener and SPOT takes a "snapshot" of the power consumption when the application performs these operations. This event-driven architecture allows developers to understand the power consumption of the application before, during, and after key power-intensive components. In addition to event-triggered power snapshots, SPOT also periodically collects power consumption information. This information allows developers to trace overall power consumption or power consumption within a given block. The power information Collector that collects snapshots and periodic samples can be configured to run in a separate process to prevent contamination of the data. To accomplish event-driven power profiling, SPOT fires events immediately before an application enters a component that was defined in the model and immediately after it leaves a model-defined component. These events work in conjunction with the periodic power consumption updates to provide developers with a complete description of how their software architecture elements consume power. SPOT's event driven model of collecting power consumption data also allows developers to identify precisely what the application was doing when key power consumption spikes occur, further helping them optimize their designs.

3.6 Requirement Specifications

System requirements are a description of the needs, and devices a user needs for an information system. Here, unique requirements of a system are identified as user requirement. These are the installed program to carry out development and operation of the system.

3.6.1 Choice of Programming Language

This project is implemented with the programming languages.

Flutter: Flutter is an open-source UI software development kit created by Google. It is used to develop applications for Android, iOS, Linux, Mac, Windows, Google Fuchsia and the web from a single codebase. The first version of Flutter was known as codename "Sky" and ran on the Android operating system.

Firebase: Firebase is a platform developed by Google for creating mobile and web applications. It was originally an independent company founded in 2011. In 2014, Google acquired the platform and it is now their flagship offering for app development.

Flutter is reliable mobile application programming language with the right modules to build a user-friendly mobile application and easily adapts with other third parties and works very well with firebase.

3.7 Implementation

Based on the design described above I develop a model of our Battery Viewer System. This model is programmed by the Java program language. It is developed on Android SDK

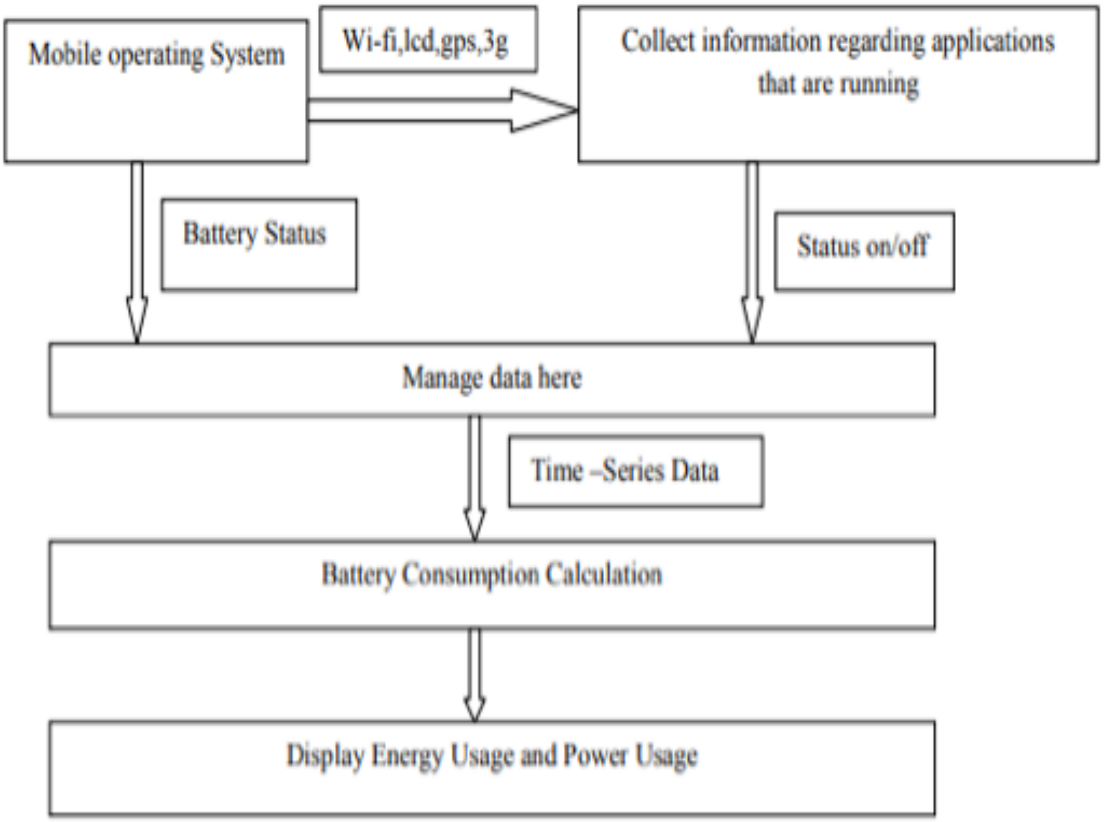

Fig 3.3: System flowchart

CHAPTER FOUR

SYSTEM DESIGN AND IMPLEMENTATION

4.1 Scope of Design

Android battery saver is an Android power optimization application that is developed to assist users save power consumption on their android phone. However, the system is covering the following modules:

Mobile Application

Dart

Flutter

4.2 System Design

System design is a logical and creative process consisting of an integrated collection of tools and specifications of input, output etc of the system. The purpose is usually to propose a specification which will enable the complete and accurate implementation of this new system. The result of such an analysis will give rise to proper documentation of events into a document or a set of documents which will enable the programmer or software engineer to start work

4.3 Input/ Output Specification and Design

The input to the new system consists of the user's credentials like: username, email and password. The samples are labeled according to the types of the input and also verify to determine its validity. The outputs are the processed information which includes the user's record. The output will be design in such a way that it will display the battery's percentage.

Table 4.1: Input Specifications

User's email and phone number
User's Android phone utility software power consumption data

Table 4.2: Output Specification

Retained power
Power consumed

4.4 System Requirement

The requirement of the new system is divided into two

1. Hardware

2. Software

Front End: Android

Backend: Dart and Firestore

4.4.1 Hardware Requirement

i3 Processor Based Computer

4GB-RAM

500 GB Hard Disk

Monitor

Internet Connection

Android Device

4.4.2 Software Requirement

Windows 7 or higher

Android Studio

Flutter

Advantages:

User can gets optimized power consumption on their mobile phone.

Users can have reports on how power is consumed.

Disadvantages:

Requires active internet connection for registration.

System may provide inaccurate results if data not entered properly.

Applications:

The system can only be used by Android user

System implementation also includes creating a new system and getting it into working order. For this to be achieved it must be done in a procedure involving how to start and stop the system, entering of information and data must be documented. Although some users already have a good idea of the system while others need to be taught about it in order to be conversant with it. Here, the entire program is tested using different data and system platform. During this phase, the developer documents clearly the user trimly and instructions on the changes the new system will introduce in the area of information management and decision making.

4.5 Program Design

The program is designed to take care Hof Android power saver system with mobile application respective task that constitutes an Android implementation. The program design is based on the input and output specification to facilitate easy coding, testing, debugging and it is in module.

4.5.1 Program Testing and Debugging

Dart programming language has distinct modes for developing a mobile application base system. Before the program is implemented, the user must test and run it and if bug is found in the program, the developer debugs the program so that the program can run without bugs when the program is free of bugs it can then be implemented fully.

4.5.2 Program Documentation

This is an important part of system development it uses the techniques and tools of system analysis and design to record and communicates the activities and result of each stay of information system development.

The hardware has all features required to best utilize the software. The software can handle the data processing assignment easily without major modification, and the language it is written in a programming language that is commonly used by mobile app developers and user.

System Evaluation and Maintenance

Maintenance of the system means keeping the system running without a problem that may cause the system to breakdown. The steps outlined below can help prevent the system:

No usage of outside source floppy disk, flash drives or CD-Rom which may contain program that can rearrange files in the system.

No external bodies should be allowed to access the system

Anti-virus package most be installed and updated annually.

The system should be out of reach to unauthorized users in the organization

Faulty material should be changed immediately.

4.6 Training of Staff / Users

The staff / Users of the new system will be given adequate training, it involves prove decision on how well the system work, handbooks are needed for this purpose to provide detailed description on how the job is done. On rare occasions depending on the complexity of the system and the level of skill currently available courses should be organized periodically to update the staff/user knowledge. Also job aid are needed to assist staff /user to carryout instruction while performing the job. Effort should be made to work on the psychology of staff/user in order to disabuse any wrong motion or mindset they have about the new system.

4.7 Change over Procedure

This is the implementation process which the changes are made from the existing system to the newly designed one. There are three basic methods

4.7.1 Pilot Change Over

This involves changing part of the system using parable or indirect change over procedure. This procedure is a trial system implementation in a subset of the overall operation as an office geographical area.

4.7.2 Direct Change Over

This is termed immediate or crash change over using this method, the old system is discontinued and the new system is put into operation at once. For a large organization. This system is risky should there be a system failure. This approach is feasible only the timing problem becomes greater as the scale of operation increase with regard to me new system an individual can easily adapt. This system without much course to regret

4.7.3 Parallel Change Over

This is a changeover procedure where the old system is run in parallel with the new system and comparison made for a whole. One merit is that, there is room for proper evaluation of the new system as to ascertain errors in any of the process. This method has been formed to be operational effective despite the fact that procedure is not cost-effective.

4.8 Documentation

Documentation is very important since it is a method of recording and communicating the activities of the system development and the outcomes of each stage in the entire process. Documentation of any system will minimize during the problem that may be encountered during modification or maintenance of the system. Documentation serves the following purpose:

Eliminates duplicates and reducing of effort in the system development process

It minimizes ambiguity in the procedures.

It prevents loss of key information vital to proper implementation of the system.

It provides adequate instruction for proper use of the system.

The complexity of the system documentation will normally depend on the extent of the system being developed. For instance, a system being developed for a temporary use would not require a detailed documentation as the system being developed for permanent use

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 Summary

Android operating system is an open platform which is becoming very popular operating system. Its open-source code is easily handled by the users to get and use new contents and applications on their handsets. Android is based on Linux kernel. Android device are being activated per day and power management of these device is becoming an issue. The one problem is common that is Low battery life of the device. It is not a common thing in smartphones. Now days, more powerful with power consuming technologies like GPS, 3G and 3GS.In this approach we do not require the client server architecture. we will include a learning engine. This learning will monitor the user behavior in terms of apps used, battery consumption and contexts, and then we will collect the information for a period of time and the fed to a learning engine. Artificial neural can be used to implement the learning engine. This pattern-based learning engine intelligently decide, how to control the smartphones features. This will build after the learning of user behavior, would not include any statically define power saving profile.

5.2 Conclusion

Energy consumption in android devices has become more important issue to save energy it is critical to monitor the energy consumption of application on android device. This paper described online power estimation. The Battery Viewer main focus on more energy consuming components likes CPU, Wi-Fi, LCD, and GPS etc. Now, at the end of our project, we have reached our goal of almost fully. We have shown that, by learning the resource utilization of user's activities we have been able to predict and supply the minimum required resources and thereby successfully save energy on mobile device. In this project, we proposed unique power management frameworks which leverage the user's

behavior to save energy. Additionally, we created an accurate power estimation model, which allow estimating power consumption of individual. components and applications in real time. Validation tests indicate that the model is highly accurate. The estimation model can be used for estimating current consumption and forecasting the future resource availability.

5.3 Recommendation

Future researchers can add more components to check battery consumptions like Video, Gaming Application etc. Another possible future work could be developing a software tool that measures the power consumes by main components and application of a smartphone on a very detailed level.

The research is recommended for future researches on Mobile phone Micro controller, the operating system., etc

REFERENCES

Adama D.W (2016) “Context-aware wireless sensor networks for assisted living and residential monitoring.” In IEEE Network, vol. 22, issue 4, 2008.

Fredrick, J. K (2017). “Energy-efficient rate adaptive GPSbased positioning for smartphones.” In Proc. Of ACM MobiSys’10, San Francisco, California, 2010, pp. 299-314.

Jeffery, K.J,(2016) “Android stack integration in embedded systems,” in International Conference on Emerging Trends in Computer & Information Technology, Coimbatore, India, 2016

Kang, K.M (2011) “Usage pattern analysis of smartphones.” In 13th Asia-Pacific Network Operations and Management Symposium, 2011, pp. 1-8.

Lui Z.W. “Accurate online power estimation and automatic battery behavior based power model generation for smartphones.” In Proc. Of ACM CODES+ISSS’10, Arizona, USA, 2010, pp. 105-114.

Neoma. Ravi, J. Scott, L. Han and l. Iftode (2008). “Context-aware battery management. for mobile phones.” In 6th Annual IEEE International Conference on Pervasive Computing and Communications, 2008, pp. 224-223.

Retrieved 17/05/2021 https://play.google.com/store/apps/details?id=com.mhuang.overclocking&hl=en

Retrieved 17/05/2021 http://ziyang.eecs.umich.edu/projects/powertutor/index.html.

Retrieved 17/05/2021 www.cs.uwc.ac.za/~mmotlhabi/apm2.pdf

Retrieved 17/05/2021 http://www.google.com/events/io/2011/index-live.html

Xu, K. (2014)“Improving energy efficiency of location sensing on smartphones.” In Proc. Of ACM MobiSys’10, San Francisco, California, 2010, pp. 315-329.

Yuwan. P.L (2015). “Improving energy efficiency of wi-fi sensing on smartphone.” In Proc. Of IEEE INFOCOM, 2011, pp. 2930-2938.

Zhang. P.K (2015) “Where is the energy spent inside my app? Fine grained energy accounting on smartphones with eprof.” In Proc. of ACM EruoSys’12, Bern, Switzerland, 2012.