Chapter 5

Software Based Methodology to Improve Battery Life of Android Mobile Devices

5.1 INTRODUCTION

Nowadays, the quantity of mobile devices (i.e., smartphones and tablets) have been developing drastically, more than other any computing gadgets. The new mobile devices are rich in information assets, for example, sensors, cameras and rich in User Interfaces (UI) like speakers and bright screens. The Internet connectivity gives their clients the capacity to impart with each other through social networking and online gaming. Also, mobile users can impart their day by day life to their friends and followers by texts, pictures, or video clips. Appropriately, the mobile phones are detectably the biggest givers to social networks (Cisco VNI, 2012). result, propelled mobile devices are required to handle these functionalities, the vast majority of which are known as intensive computing tasks.

There are as of now no less than 45.5 million individuals in the United States who own a mobile phone, and that number is relied upon to increment to 1.5 billion by 2015 (Pathak et al., 2011). Mobile phone users expect and command that their mobile phones should have high performance at the same time consume minimum power. Clients could not care less on the off chance that the hardware is improved for greatest energy efficiency or if the software is expanded for energy effectiveness. When they assess their gadgets, they take a gander at the general battery life of their gadgets. However, because of the battery size imperatives, the measure of energy put away in these gadgets is restricted and is developing by 5% yearly (Miettinen and Nurminen, 2010). Therefore, we have to dissect the energy effectiveness of these mobile gadgets and utilize the lessons learned with a specific end goal to improve the energy consumption and in this manner increment their energy effectiveness.

The previous decade has recognized that there is a gigantic development in the wireless handheld gadgets and internet technologies. The most generally utilized handheld gadgets are Black Berry, Nokia, iPhone and android devices. There are increasing much in fame with extraordinary functionalities and processing abilities which undermines the third party applications, modern smartphones accompanies the mixture of appli-
cation and number of wireless interfaces in particular, Wi-Fi, 2G, 3G and Bluetooth (Vahdat et al., 2000).

Android gadgets have been putting forth various radio frequencies. Google has created an open source operating system called Android. (Android OSP, 2012). Android force Dalvik JVM is streamlined particularly for moderate CPU. A large portion of the handheld gadgets have lithium ion batteries, which can not withstand an entire day in light, of the fact that the vast majority of the applications are made heavier so it devours much battery specifically games which has high end graphics and numerous different wireless applications issues. Now a days numerous applications comes in immense size which performs numerous operations, which expands the majority of the battery. Meanwhile the android operating system has some dynamic energy management strategies that are incorporated in the product that moderates the power (Vallina-Rodriguez and Crowcroft, 2013). However clients still grumble about the battery life. In this chapter we implemented the application totally in the light of java. We assessed our own key stroke so that it can be easily installed in android gadgets without licensing (Schroeder, 2011). We chose this arrangement so that it can help the client to flip their mobile data or Wi-Fi service off with no delay, this application will naturally empower the night mode, there are more features which we will discuss later.

5.1.1 HISTORY OF SMARTPHONES

Mobile devices were begun as voice gadgets in the mid 80’s utilizing analog radio telephone system this is known as the first generation (1G) mobile system. In the mid 90’s, the innovation climbed to the Second Generation mobile system (2G), which reshaped the whole mobile system. The 2G succeeded in executing the digital communication in the mobile system. It was executed in four primary systems: Global System for Mobile Communication (GSM), Time-Division Multiple Access (TDMA), Japan Personal Digital Cellular (PDC) and Code-Division Multiple Access (CDMA one). In this era, information packet services were begun, for example, Short Message Service (SMS) utilizing circle-switch information. The SMS gives a low data rate for short messages. This service speaks to the introduction of data communication not withstanding the voice communication in the mobile systems. After that in mid 2000, the data services were updated by locks in General Packet Radio Service (GPRS), which presents the 2.5 Generation (2.5G). After this upgrade, the Enhanced Data Packet for Global Evolution (EDGE) was locked in. This last upgrade presents, what is called 2.75 Generation (2.75G).

With the accomplishment of the 2G developments, more requests were expanding on high data rate, network capacity, and frequency bandwidth. These requests drove to the third Generation (3G), which is produced into two systems. The first
system is known as the Universal Mobile Telecommunications Systems (UMTS) and CDMA2000, where the second system is known as High Speed Downlink Packet Access (HSDPA). Today, the achievement of 3G is exceptional since the traffic of 3G data is expanding between 300% to 700% consistently (Brydon and Heath, 2009).

In parallel to mobile system advancement, different wireless technologies, for example, Wireless Local Area Network (WLAN) and Wireless Personal Area Network (WPAN) discovered their route into cell phones. For instance, WLAN gets to be provides prevalent service in numerous zones like homes, coffee shops, campuses, airports and hotels. The fundamental driver for these innovations on mobile devices is that they give higher data rate with much less cost than the expense of mobile system information. Consequently, today numerous spots give free access to their WLAN. Note that, the WLAN was designed as an extension to LAN and it does not include in the mobile system advancement. The essential contrast between mobile systems and other wireless system is that the mobile communication system have scope range in couple of kilometers while the WLAN in couple of meters and WPAN in few meters (De Vriendt et al., 2002).

The development in the mobile device advancements have assumed a part in the development of mobile phones. Notwithstanding the significant advancement in the communicating technology as discussed above, advances in the hardware and software of mobile devices have the significant impacts. The advances in the hardware of the mobile phone include helper highlight to mobile devices, for example, Global Positioning System (GPS). In the same manner, the advances on the semiconductors innovation lead to smaller and lighter mobile device and include more capacities and features. Progresses in the software supports mobile phones with Internet services (e.g., web browsing, gaming, e-mailing and office beneficial applications).

The meeting of the mobile communication systems, which are the Internet, the mobile computing, and the multimedia broadcasting, frames another ubiquitous Information and Communication Technology (ICT) framework that is recognizable these days and will be regular soon. The Fourth Generation (4G) mobile communication system is the most vital stride toward this merging. The 4G guide is to achieve all-IP architecture this decreases the infrastructure cost, expansible, and obliging for future services on the computing, broadcasting, and the Internet. Additionally, the all-IP design bolsters a consistent availability among heterogeneous network called true Internet mobility (Bonomi, 2010). The 4G incorporates every single worldwide networks and every single terminal types in to one mobile environment (Khan et al., 2009). In this advancement, the mobile phones move from voice driven gadgets to Internet Protocol (IP) driven gadgets (De Vriendt et al., 2002; Khan et al., 2009; Sun et al., 2001; Akan and Edemen, 2010).
5.1.2 MOBILE OPERATING SYSTEM

There are numerous mobile operating systems in the business sector, for example, Android, iOS, Blackberry and Windows Phone. As per the factual information, Android holds 81.3% of cell phone market share, iOS possesses 13.4% of smartphone market share, Windows Phone develops to 4.1% of smartphone market share of the overall industry and Blackberry has just 1.0% of the market share (Strategy Analytics, 2013). In this chapter, we concentrate on Android operating system and simply give a few correlations between Android and iOS.

Android operating system depends on Linux kernel. It depends on direct control, which utilizes touch information sources, for example, swipping, tapping and pinching to control on-screen objects. It is connected on to smartphones, tablets, smart TVs and cameras. iOS operating system was created by Apple. It is derived from Mac OS X, which depends on the progression of Unix-based graphical interface operating systems. iOS is additionally taking into manipulation, which utilizes various touch gestures, for example, swipping, tapping and pinching to control on-screen objects. Android smartphone frequently gives an impression of shorter battery perseverance than Apple’s iPhone family with bigger battery packs and more tightly third party software control. Therefore diminishing the power utilization rate through background services is a basic issue for Android based mobile computing devices. One of its mainstream reason is that Google distributes the majority of the source code for Android, including system and telephony stacks, under free and open-source programming licenses.

It allows users to change their system according to their preference. It permits clients to change their system as indicated by their inclination. iOS does not permit clients to change its UI, which baffles for the individuals who need to customize their smartphones (IOS vs Android, 2013). Interestingly, Android gadgets are more open to clients to fabricate their customized UI. As far as steadiness, iOS is more steady than Android, since iOS just gives clients or applications low needs to control the mobile operating system and does not permit clients to change the operating system with a specific end goal to guarantee the integrity. However, Android gives developers and enthusiasts Android Open Source Project (AOSP) source to build up their customized forms of operating system. Though, Android offers an open platform and clients can redo their operating system, it is conceivable to uncover their private data. Moreover, makers can pick diverse hardware, models and screen size, which can bring about a perfect issue. Likewise, some updated applications won’t have a magnificent performance in each Android phones.
5.2 RELATED WORK

5.2.1 LIMITATIONS OF SMART PHONES

The maximum capacity for smartphones and tablet PCs might be compelled by certain specialized points of confinement such as battery continuance, computational performance, and portability. Current mobile applications own all the more intense capacities yet require bigger calculation and faster edge rate, which consume more battery energy. Throughout the years, battery’s energy thickness has not enhanced as altogether as semiconductor advances. Not at all like conventional cell phones, for which a solitary charge may keep going for a few days, today’s smartphones and other mobile devices scarcely support typical utilization for a day without being charged. Consequently, there are two approaches to drag out battery perseverance: one that builds the vitality limit, i.e., expanding the battery size, and the other that diminishes power utilization rate. Expanding the battery size prompts the expansion in manufacturing cost also. Additionally the gadget size is for all intents and purposes altered for ergonomic thought, we are prone to exchange battery continuance for computational performance, or the other way around, when designing a smartphone or a tablet PC. That is, we either make our gadget predominant in performance however coming up short on battery sooner, or making them withstand longer can conceivably bring down the performance of the device. With the gadget size thought to be altered, we may tradeoff battery strength for computational performance as part of designing a smartphone or a tablet PC.

5.3 TOP 10 ENERGY EFFICIENT PROGRAMMING RULES

There are general rules that software developers ought to receive with a specific end goal to guarantee that their applications are energy efficient. The 10 rules are as per the following:

1. Extend platform sleep duration

In order to build the energy efficiency of a platform, developers ought to keep the platform’s parts dynamic for the briefest conceivable term and maintain a strategic distance from awakening parts unless important. For instance, when creating applications for Windows OS, designers ought to avoid from diminishing the timer resolution interval with a specific end goal to maintain a strategic distance from incessant wake-ups of the platform. Another illustration is utilizing the OS’s API, which can extend sleep duration. Windows gives two APIs, SetWaitableTimerEx and SetCoalescableTime, which can be utilized as a part of request to diminish superfluous wake-ups.

2. Event-driven architecture
Surveying can bring about pointless wake-ups for the platform. Accordingly, developers should utilize event driven interrupts as opposed to surveying for data.

3. Design energy effective user interface (UI)

Energy effective UI comprises of quickening user interaction. At the end of the day, clients ought to have the capacity to get to the required screen utilizing by minimizing the amount of clicks. Moreover, visit screen changes can affect the energy consumption of the display and GPU. In this manner, developers ought to balance the interface experience with the energy efficiency of rendering the updates. Additionally, developers ought to consider whether to display an advancement bar (which continues getting updated) versus a basic occupied marker (which doesn’t require interface update). At last, when developers are thinking about graphically rich interfaces, they ought to weight the effect of their imagination on the energy efficiency of the application. For instance, despite the fact that sprinkle screens can be graphically rich, they can have high energy consumption.

4. Consider data locality

Memory and capacity are high power devouring segments of a mobile device. In this way, developers are urged to work on little information at once so information can stay in cache, keep the rate of memory utilization low, and keep the number of reads from low storage.

5. Efficient multithreading

Multithreading inside applications is a typical practice. In any case, multithreading means simultaneousness. Be that as it may, unless strings are simultaneously running on a multi-center gadget, then they won’t be enhanced as far as energy efficiency. Figure 5.1 envisions the effect of thread execution grouping on the package active duration. In a perfect world, a developer needs to focus to empower the package to enter in to idle sleep state for the longest conceivable span.

6. Take advantage of context programming

Context programming makes the applications more quick witted. Designers ought to empower their applications to sense the environment in which they are working and trigger responses in light of the adjustments in nature. Case in point, they ought to have the capacity to sense if the gadget is in AC or DC mode. In the event that the gadget is associated with power, at that point they can trigger reinforcement.

7. Be aware of low upgrade frequency intervals of sensors
Today’s mobile devices are equipped with a few sensors such as an accelerometer and gyroscope. These sensors are managed through APIs that empower designers to change the interim upgrade recurrence. Designers ought to stay away from low upgrade interims of these sensors with a specific end goal to decrease the quantity of wake-ups brought on by the updates and empower the sensors to go to idle sleep states.

8. Coalescent network activities

Each connection with the network devours energy to transmit or get packets notwithstanding the tail power of the system gadget. Subsequently, spacing connections out superfluously can squander significant power. Developers may re-vamp their code with a specific end goal to gather the application’s network connection together.

9. Close network sockets

After an application completes the process of transmitting or accepting information over the system, it does not naturally close the connection. Accordingly, after an interim with no network exercises, the Wi-Fi radio enters an idle rest state until the server will time out and close the socket by sending a FIN packet, which will switch the Wi-Fi radio back to dynamic state. In this manner, it is suggested that developers close network sockets when they are finished with transmitting or accepting data.

10. Avoid high-resolution images

High-resolution images are pretty and may have an additional benefit to the client experience and feel of an application. Be that as it may, these high-determination pictures are immoderate, regarding energy efficiency, to render. In this way, developers need to weight the significance of high-resolution images versus energy efficiency of their applications.

The authors (Park et. al., 2014) implemented the model of Phone2Cloud on Hadoop environment and Android. Two arrangements of test, including scenario experiments and application experiments are intended to assess the framework. The executed results demonstrated that Phone2Cloud could viably spare energy for Android smartphones. Then they dissect the energy consumption and execution time of the applications in their analysis under four diverse factors, including bandwidth, input size, delay-tolerance threshold and CPU workload. They used ZTE V880 smartphone in their analysis. It utilizes Android OS (version 2.2), incorporated with Wi-Fi interface and is enabled of EDGE data connectivity. It has a 256MB memory, a Qualcomm MSM7227-1 CPU.
Figure 5.1 Comparison of the impact of multithreading on package active duration.

with 600MHz frequency and a battery limit of 1250mAh at 3.7 volts. Two arrangements of investigations are directed and the results exhibited that their framework was of awesome viability. With effortlessness, they just utilize a naive approach to foresee execution time and just consider input size and CPU workload. There are numerous optional techniques and elements they can take into thought, and they contrasted those strategies with their methodology in the future work. In addition, as they said prior to their framework is a semi-automatic offloading framework.

5.4 OBJECTIVE

The objective of this chapter is to develop an app to extend the energy of Android mobile devices.

5.5 THE PROPOSED METHODOLOGY

There are numerous approaches to decrease battery consumption, for example, turning off unused applications and components. The background services, which the users may not know, routinely take up the CPU and/or the communication module in mobile devices. These services influence the power consumption essentially even when mobile devices are not in use.

5.5.1 SLEEP TO SAVE

If the mobile task scheduler has no work to execute it is considered as being in an idle state. Sleep states generally waste the battery in handheld gadgets (Flinn and Satyanarayanan, 1999), so in case you are in sleep mode you can be sure to get the battery
decreased status. So we are all that much considering in the status of sleep mode. Another mode is idle state mode, this mode is normally thorough out period (except sleep mode). Secondary thing we are forestalling interrupts. In the midst of both the states, interrupts may happen. Interrupts have two sorts internal and external (Vallina-Rodriguez and Crowcroft, 2013). On the off chance that somewhere in the range of an external or undesirable interrupts comes, the processor module goes back to the running state. The processor saves more energy if it is in sleep state as opposed to idle or running state.

5.5.2 POWER EFFICIENT COMMUNICATIONS (WNIC)

There are also many benefits to choosing to go wireless. Have a look at a few of the many benefits below. As time goes on wireless networks are becoming better and better, they are becoming easier to setup, they are increasing in security and they are getting faster and faster in transfer speeds.

SETUP

Because wireless networks operate wirelessly, they become very easy to setup and change. However, there are still a few things to getting a wireless network setup. To create a functioning wireless network, you need to provide each of the computers with wireless network cards, and also purchase a wireless router.

SECURITY

Previously people were very worried that people would simply hack into their wireless networks, and this was a valid worry. However, today the security that wireless routers and cards offer, is great, if you are willing to just go about setting it up.

CHEAPER

Another major benefit to going wireless over wired, is the fact that wireless is so much cheaper than wired. Although the wireless cards may cost more than a typical computer network card, the benefit in not having to buy network cables far out weighs this.

SPEED

Another reason that people have chosen wired networks in the past, was due to speed. However, with the latest progressions, wireless is quickly becoming just as fast or faster than their wired counterparts.

We are suspecting power proficient. We have to work in the interfaces. Interface is just some essential part that share one thing which has simillar set of attributes. The Wireless Network Interface Controller (WNIC) act as a production in a method to allow
client to perform dynamic power management (Priyantha et al., 2011). A WNIC card does not consume any energy in the power off state (Pathak et al., 2011). While the idle state consumes a lot of energy in the time of transmission in the same sense we are searching for active state light of the energy was used.

5.5.3 GUI DESIGN

GUI is just Graphical User Interface which act as a transitional between the mobile phones and the users. GUI is the most essential part in the process of build PC’s and laptops and even like the longing of mobile (handheld) gadgets for the higher-lower communication from their mobile handheld gadgets (Carroll and Heiser, 2010).

Handheld devices are humbler in size, its use to cooperation among the applications. Generally developers use common attributes on a very basic level developers focus on the human factors and the applications. It has two types 1. Display 2. User Interface

5.6 TOOLS USED

In general MVC architecture can be used for application, in order to allow extraordinary powerful development and diminishing unreasonable coupling between the logic and the UI (Flinn and Satyanarayanan, 2004). Here we actualized methodologies for a decent UI which can’t reduce the use of more battery.

Our application has a decent flexibility and we have used java eclipse and android advancement package which help us to build up a decent application with extraordinary fundamental interface and this interface is further designed by eclipse’s built in layouts and the layout-namely, battery-minder, list-item-icon, mode-selection and widget, all are constrained as xml file. The symbol has been included in res folder in which various drawable folders are made with the objective that it will be easy to switch from platform to platform, As we made the objective platform as version 4.2.2, it can change its symbol size as per default as 64px (Shepard et. al, 2011).

We have built up a widget for our application so that it can be easy to flip our application as enable and disable, as same as in application symbols, we composed a widget so that it can have its size which is dependent to the display of the android devices, the smallest pixel is 16px. We have included android private libraries and android conditions (Ferreira et al., 2011). We have also implemented a time picker management as android slant and built up a package called com.tobyw2cypher.android in which we included utils.java, utilsConstant.java and debug.java, we furthermore added to one more package named as com.tobyw2cypher.liionsaver in which we listed Batteryminder.java, Batteryserviceinfo.java, datatoggler.java, settings.java and togglewidget.java (Vallina-Rodriguez and Crowcroft, 2011). Moreover we added a data switcher package which causes the installer to check which version of android operating system is being used to
be particular, Droid switcher, Gingerbread switcher and Ice Cream Sandwich switcher. Later the eclipse will automatically make a R.java file and buildconfig.java record like:

```java
/** Automatically generated file. DO NOT MODIFY */
package com.tobyw2cypher.LiIonSaver;
public final class BuildConfig {
    public final static boolean DEBUG = true;
}
```

Different icons which are being used as part of the application is stored in resource/Crunch directory under drawable directory as we talked about some time recently, these crunch directory which has various iconic directories with API version specifically hdpi-v11. Besides, Android manifest.xml file created normally in which the application setup and the objective android devices and their version is being incorporated which can be used while distributed or propelling the application by exporting or using the android tools for producing signed or unsigned application. Also we incorporated a gradle wrapper property which gives the information like:

```
#Tue Oct 08 18:04:48 SAST 2013
distributionBase=GRADLE_USER_HOME
distributionPath=wrapper/dists
distributionUrl=http //services.gradle.org/distributions/gradle-1.8-bin.zip
```

Likewise, we tested our application in ant debugger. The performance, validation testing and usability testing is being tried with jubula, it is an eclipse based testing.

## 5.7 FEATURES INCORPORATED

Initially, there are a couple of potential elements which we had hoped to execute, and may still be considered on. We had actualized various features with respect to saving the battery life over the mobile data and Wi-Fi which includes:

- Battery Minder
- Setting
- Battery service Information
- Data Toggle
- General Receiver
- Main Process
- Mode Select
- Benchmark and Comparison
5.7.1 BATTERY MINDER

This module incorporates kernel wake lock and CPU details and Figure 5.2 demonstrates the screen shot of the module:

The above module named battery minder is works with various packages/classes which fuse ByteArrayOutputStream, Timer, Content Intent, Resolve Info and Android OS Bundle. Few user described packages developed such as Preference Manager, Log Manager and Activity base which oversees widgets.

In this module we set the default esteem for wake lock as 60*30 by means of seconds in a definite way, which is being connected from battery INFO which will simply demonstrate the wake lock of 30 minutes. In this module we designed a spare instance state where the CPU state is being saved to a log and showed by means of a list to the user as 'Time spend at various CPU speed'. The set 'substance perspective' is designed as a format to show the total battery status and the present time is shown using the built-in function System.currentTimeMillis() * 1000 which displays the raw real time, if it fails in demonstrating the local time it uses a try function and shows by executing from uptime by spiltting as Getting the CPU stats is as clear as getting the time where it just gets the string as cslog and displays the CPU stats.
CPU stat is built with the following features:

- CPU frequency
- Time Format
- CPU stat message
- CPU Sleep
- System Uptime
- Wake Lock Info
- Battery Usage
- Error Report
- Power Usage Info
- Power Usage Intent.

The CPU status is being appeared with RAW organization by executing the built-in time-in-state procedures and portraying these RAW data as Long data type by passing the information by multiplying it with 100001 and augmenting it with uptime by dividing it with 1000f (Frequency) which is formatted as Mega Hertz with time format as appeared in Figure 5.2 battery minder. The elapse time appears by getting the value from system uptime.

“Wakelock” is an mechanism of power management service in Android OS, which can be used to keep CPU awake (Partial wakelock) and keep the screen on (Full wakelock) (Roy et al., 2011). A wake lock is essentially used to lock the device in an “awake” state, in which the CPU will be on, and the screen may or may not be on. It is unreasonable to do these errands without a wakelock if the phone is in sleep mode otherwise, as then the CPU is similarly in sleep mode. Regardless, if the user is using the device for something else, and client’s application is in background, you can do these errands without a wakelock. Skirting on everything user doing is battery concentrated (sensors, Wi-Fi, wakelock) and user should not ever do it so that the user don’t degrade the user’s battery life.

Getting the status of wakelock is insignificant, tricky which needs a root approval from the third party tool example super user which gets the wakelock access. The Figure 5.2 exhibits the data of android which is not rooted where it doesn’t demonstrate the wakelock details. While in default this application will simply indicate stuff that is wakelocked for more than 30 minutes. Along these lines the wakelock is used here to
prevent the CPU from sleeping. The time configuration is appeared by giving back the qualities as micro seconds (microsecs - 500)/(1000*1000).

The time presentations is a String which attaches days by simple logic as seconds / (60*60*24) and the use as hour*60*60. By suffixing it with hours for minutes it does the same by excluding hours and just multiplying it with 60 seconds and adding it as minutes.

5.7.2 SETTING

This module allows the user to enable and disable the application and other time picker management tools which helps us to disable and enable the needed options for the devices for later use.

The Setting module is being actualized by importing the list of packages such as Preference manager, Shared Preferences, Context and Toast which is appeared in Figure 5.3. Before executing the general settings we actualize few center components, for example, setting the default values to investigate night mode as 'false'. Likewise, Night mode core settings with two primary function of it as begin and end. Minimum awake time is arranged as last with the objective that it starts at start up. The few core elements for display property is being set as awake time, sleep time, default awake period, default
sleep period and run version.

Here it uses a HashMap class to deterministic emphasize. HashMap is an execution of Map. Each and every optional operations are maintained. All parts are permitted as keys or values, including null. The iteration order for HashMap is non-deterministic. If you require deterministic iteration, use LinkedHashMap.

The execution of HashMap is not synchronized. If one thread of several threads of an instance modifies the map essentially, access to the map ought to be synchronized. A structural change is an operation that incorporates or removes an entry. Changes in the estimation of an entry are not structural changes. The Iterator made by calling the iterator method may throw a Concurrent Modification Exception, if the map is basically changed while an iterator is used to rehash over the parts. Simply the remove method that is given by the iterator allows to clearing of parts during iteration. It is unreasonable to guarantee that this framework satisfies the desires in all cases of unsynchronized concurrent change. It should simply be used for debugging purposes.

To flip the widget there are few packages that has been used to take up with the android center components such as Pending intent, Context, Intent, Shared Preferences, Remote Views and Widget Providers. At first setting the parameters for widget with essential flip by inheriting the App widget supplier which does not appear in the setting menu, this is the reason it uses remote views.

5.7.3 BATTERY SERVICE INFORMATION

This module gives the data about the present state of battery. The module uses few built-in and user defined packages/classes such as Map as we gave the brief depiction about mapping in the setting section, Timer, IBinder, Sparse Array. Here we built few features which deals with the data in the stats including the already spared information and also include only the running after the last time the device was unplugged in the stats.

It also offers the user with the major component as screen on sort where the screen brightness is default which is set to lessen to save the energy moreover offers the user with options such as dark, dim, medium, light and bright. The usage of sparse array in this application is to get the User Id status and display it to the user in a precise way by throwing it as exceptions. This feature also include Battery Uptime where the application processes Real-time battery stats and screen on time and discharge time and the level of the android device.

5.7.4 DATA TOGGLE

This offers a smooth and easy way to tweak alerts, Notifications, Broadcasting, Wi-Fi Manager, Async Talk, Mobile Data toggles and screen service modes. Switches
between auto synchronization decisions in android, checks the screen services on the off chance that it is being slaughtered. The main functionality is to turn the data on when the system wakes up, it skips to turn on the data if the service is in Airplane mode and it automatically disables the data service. Its usefulness to run the notice depends on to summoning the user to update the status of the present screen time, type and mode which is being initiated. It can similarly handle the broadcasting features from the contraption and enables either the Wi-Fi or mobile data and saves the present state by clearing the past notification. It sets the flag if any synchronization should run while checking for network associations. Once it is associated it begins the force sync features of android contraption. Various android users do not trust android contraptions to dependably inform of the data connection, so we present a manual check choice with appropriate timeouts.

5.7.5 GENERAL RECEIVER

General receiver gets the broadcast, checks for the wake or sleep of the screen. If screen on data is engaged, switch data off when screen goes to sleep, indenting to switch off the data with no delay. It uses a timer to switch off the data and it also has the ability to arrange the sleep ararm and tell the screen is on and if the data is enabled. It also has features like the popup message at whatever point the android system or device is booted up, it moreover tells the user when the network is being changed or connectivity is changed with options in it as start sync with timer or sync just once. General receiver notifies when the contraptions being plugged with AC power charger or USB charger and prompts if it is removed from the charger.

5.7.6 MAIN PROCESS

The thought about this module is alluded from one of greatest battery saving application “greenify”. The basic functionary of this essential module is to handle the different limits and features of the application. This essential strategy consolidates various utilities such as Data Format, Alarm Manager, Notification Manager, Pending Intents, Resolver, URI, APN switcher, Mobile Data Switcher and other implicit bundles. It generally starts with general receiver as a data condition of period 24 hours.

As of late it starts the scheduling procedure by using an android scheduler, and scratches off each one of the synchronizations and let the widget understand that the application has been started and engages the Li Ion saver application. If the user has disabled fiddling with the mobile data, it reestablishes APN type else it enables the mobile data and switch the APN. The main process sets up the default flags such as mode selections, alarm managers. For the Night mode there is no prerequisite, for ordinary data alerts. It confirms whether the data is on. It stops the scheduling procedure
and re-awakens the data and does the same process again.

When the alarm goes off, we have to broadcast an Intent to our Broadcast Receiver. Here we make an Intent with an explicit class name to have our own receiver (which has been distributed in AndroidManifest.xml) instantiated and called, and then make an Intent Sender to have the intent executed as a broadcast, this Intent Sender is designed to allow itself to be sent multiple times. It offers the user with various important features such as setting the sleep time period, booking the wake alert, setting up the night mode alarms which will be designed as a basic data format which has been added unequivocally to the application during designing it.

Mode Select, package is designed with the help of Android studio with open source symbols. It shows the user a list of modes in a list view as exhibited in Figure 5.4. In this the user can select the modes like standard and night mode as in Figure 5.5. In night mode the application will automatically toggle off the Wi-fi and mobile data services and we also offered the sleek setting GUI so that the user can easily view the current status of their battery and also the restart period of the device.

5.8 BENCH MARK AND COMPARISON

This section is to give the better results of performance and energy efficiency of the android contraption with and without Li Ion saver application installed in it. In Figure 5.6 it clearly demonstrates the assessed battery life with the monitored results. It demon-
Figure 5.5 Night Mode Select.
strates the overall average standby time elapsed by both in 100% and current state of the battery. It furthermore demonstrates the estimated time for the battery getting drained using 3D, 2D Games, Online and offline videos, Internet over Wi-Fi, Internet over mobile networks, other broadcasts. It dependably changes in time when the mode is being picked such as night or travel mode. It also includes battery minder as we have given the complete information about the use and the feature of the battery minder utility in the past section.

In the comparison section we tested our android software based methodology in various devices. Figure 5.7 and Figure 5.8 demonstrates the result of this tested in Asus Zenfone and comparison of various gadgets as appeared in Table 5.1.

In disabled mode the battery gets exhausted early, from the Figure 5.7 battery saver mode disabled it clearly demonstrates the estimated time of the energy. It incorporates Global estimation as some hour followed with minutes and also demonstrates the average times by means of percentage.

From the Figure 5.8 battery saver mode enabled which demonstrates the time estimations of present and past services or history of last battery spared states and measures the general battery life for a specific time or days or week tops, example from 23:00 to 14:00, it saved few hours of battery life representing it in a graph with date and time embellish within a user interface. The detailed result analysis of capacity of mobile device with Battery Saver App enabled and disabled is represented in Figure 5.9.
Table 5.1 Extending Energy of different Android mobile devices using Battery Saver Mode Enabled

<table>
<thead>
<tr>
<th>Mobile Phone</th>
<th>Usage when Battery Saver Mode Disabled (in Hours)</th>
<th>Usage when Battery Saver Mode Enabled (in Hours)</th>
<th>Extended Hours (in Hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>6.36</td>
<td>9.72</td>
<td>3.36</td>
</tr>
<tr>
<td>Model 2</td>
<td>5.78</td>
<td>8.41</td>
<td>2.63</td>
</tr>
<tr>
<td>Model 3</td>
<td>7.28</td>
<td>10.93</td>
<td>3.65</td>
</tr>
<tr>
<td>Model 4</td>
<td>6.93</td>
<td>10.03</td>
<td>3.10</td>
</tr>
<tr>
<td>Model 5</td>
<td>7.18</td>
<td>9.93</td>
<td>2.75</td>
</tr>
</tbody>
</table>

Figure 5.7 Battery Saver Mode Disabled.
Figure 5.8 Battery Saver Mode Enabled.

Figure 5.9 Battery Capacity with BSM Enabled and Disabled.
5.9 SUMMARY

Final conclusion of this chapter is the undesirable battery utilization for the mobile phone users (handheld devices). A considerable lot of the organizations concentrate on hardware implementation like size of the battery expanded. Furthermore hardware designs are complicated because of that we have produced the solution in software methodology. The application works based on the user time synchronizing. So auto process conveys the work because of that a few activities are working beyond that and the battery life is increased. We implemented power synchronization method which automatically syncs an online application in particular WhatsApp, other email applications, and so forth.