Modeling and Profiling the Energy Consumption of Mobile Devices

Thesis

Written by Chenglong Cui
a1678254
Supervised by Markus Wagner

Abstract

Power modeling and profiling is an effective way to understand the power behavior of smartphones. However, few tools that effectively combines portability, accuracy and automation have been proposed yet. This report aims to combine the advantages of different types of power profiler and generate the accurate power model for the sub-component of smartphones. To create a precise power model, we firstly consider display, camera, Bluetooth as subcomponents of a smartphone. We conduct experiments for each subcomponent to get the battery discharge curve for these subcomponents.

The results of this thesis could contribute to a deep understanding of power consumption of smartphones. Specifically, the implication of this thesis will contribute to developers to optimize the power consumption for software applications development on the smartphones. Furthermore, users can also benefit from this thesis to guide their usage of the smartphones.

1. Introduction

The usage of smartphones has experienced significant growth in the recent years. By the year 2015, the number of smartphone users has reached about 1.86 billion [1]. Although the popularity of smartphone is increasing dramatically, there is a growing tension between the increasing demand for smartphone performance, e.g. clean responsive design, and the limited power from the battery of smartphones. On one hand, end users expect smartphones to provide more robust hardware and smoother software functionalities that provide a richer user experience, which leads to a rapid growth of the need for energy. On the other hand, battery capacity is severely limited on account of the constraints on either size or weight of the device. As a result, the improvements in the smartphone energy lags far behind the demand for active power management.

However, several challenges make energy optimization difficult to perform. First, application developers usually do not have sufficient understanding of the power consumption of their design, and this will increase 30 to 40% more energy usage [2]. Software developers are always required to understand the power implication of their design when developing an efficient power program. However, a number of application developers including the experienced ones are trained to develop on general PCs and hence have limited experience with power-constrained systems, for example, smartphones. Therefore, our research could help them to build the necessary understanding of energy system first to overcome this lack of knowledge and expertise.

Moreover, designing strategies for performance on mobile devices is different from designing for general purpose PCs and servers because unlike traditional workloads, portable workloads are interaction-centric which may not apply to the general PCs. Thus, we need to help mobile designers to identify the applicable performance metric and understand the performance bottlenecks.

Also, the balancing strategy of the trade-off between power and performance of smartphones is crucial for applications and systems to keep end user satisfied. User experience is always destroyed by applications when designers only consider one aspect, for example, data load speed in their designs. In other words, any power saving techniques are needed to look at how it can affect the performance vice versa.

In summary, optimizing power consumption of these devices is critical for both developers and users. A primary premise of efficient power management is a good understanding of where and how the energy is consumed: how much power the smartphone is used by which parts of the system or sub-component under what circumstances.

This Thesis is structured as follows. Chapter 2 contains more background information regarding some technical terms about power profiling and some general information about sub component of mobile devices are included. Chapter 3 contains the general design of the power consumption measurement experiments of subcomponent of smartphones that described in detail. The results from the experiments conducted are discussed in Chapter 4. Finally, Chapter 5 presents conclusion and future work.

2. Literature Review and Related Work

There are a number of different power measurement and estimation models and the corresponding power optimization strategies in the literature at different levels, for example, system level, thread level and application level. In this chapter, we summarise the literature and techniques in system level. Specially, when we measure power consumption in the system level, we split a mobile device into a set of subcomponents including CPU, display, network, and camera etc., and each subcomponent will be measured individually and build its own model. This chapter summarizes literature that relevant to this project. The review begins with a background energy profiling, In Section 2.1, information on energy profiling is presented. Section 2.2 describes backgrounds knowledge about power model. Finally, Section 2.3 presents information about subcomponents in details.

2.1 Power profiling

Power profiling is defined as the process of creating an power profile that represents the relationship between power consumption value and functionalities of components of a smartphone. This process requires power measurements and power modeling. Chapter 2.1.1 will discuss power measurement, and chapter 2.1.2 will discuss the power model.

2.1.1 Power measurement

Two main types of power measurement methods including hardware based measurement and Software based measurement are commonly used, which described in the [3]. Hardware based measurement measures power consumption with an external equipment such as a digital indicator or other similar instruments. Software-Based measurements mainly focus on making use of data collected from the different software system interfaces to perform calculations to assess the power consumption at application software level.

2.2 Power Model

Power Model is the primary approach for mapping power measurements with functionalities [4]. The general idea for a typical power model is that power consumption is calculated using pre-constructed models for the power consumption of a smartphone’s components. Models are built by observing a component’s states, collecting their corresponding power consumption data, and applying regression on the system. These models are trained beforehand, e.g., in the lab, and are used to attribute prefixed values of power consumption to smartphone functionalities according to some criteria. In this section, we will discuss different smartphone power modeling approaches.

2.2.1 Types of power models

Power Models are mainly divided into three categories based on the kind of input variables that utilized in the model: utilization based models [5], system call based models [6], and code analysis based models.

Utilization-based models collect the power usage data of a subcomponent by correlating the power draw of that component with measured resource usage. For an application or process, its power model includes variables that reflect the resource consumption of all the different subcomponents that are active while running that application.

System-call based models try to solve tail power states that appear in some smartphone components. A tail power state means that a component is still using resources, even though the application that invoked it has completed its tasks. It primarily used to improve the responsiveness of the corresponding components, but their existence mostly affects the process of power consumption measurement. Pathak et al. [6] try to solve this problem by using system call based model. When the information is system-call based, the system calls are tracked, and they provide a clear indication of which hardware components are utilized by a particular application or process and in which way. The duration of the active power state of the corresponding hardware subcomponents depends on the volume of the I/O, which are specified in the parameters of these system calls. When the actual I/O tasks are finished, the tail power can be estimated using close like system calls.

3. Models-Based Code Analysis. The third category of models relies on the analysis of the program code to be executed. The advantage of this approach is that it can estimate power consumption without even running the software on a real system. This method is less frequently used, as the power consumption is often context dependent, which is hard to account for without actually running the program code on a real device.

2.2.3 Survey of Power Profilers

PowerTutor [7], proposes a utilization-based online measurement method which uses a built-in battery voltage sensor and the derived discharge voltage curve. However, they did not take external factors that might affect this curve, such as battery aging, into consideration. They also used an external power measurement device to manually construct their model and compare it with the one produced by their automatic tool. Another example of utilization-based modeling is DevScope [8], which is a self-adaptable autonomous power modeling tool based on the internal Battery Monitoring Unit (BMU). While

DevScope offers power models for most of a smartphone's modules, it still suffers from the low update rate of the BMU, which leads to hidden power states (occurring power states that do not last more than the sampling period). In Eprof [9] a system-call-based power attribution method was introduced, where the power behavior of a smartphone component was modeled as a finite-state machine (FSM). They also attempted to attack the tail power states problem by attributing lingering power (tail) to the last system call triggered (last-trigger policy). Even though the results in power consumption attribution were promising, the complexity of building an FSM for the whole system undermines the practicality of this method. This system was based on an external power measurement device. PowerProf [10] adopts API-call-based information, by utilizing an API to the battery interface to acquire power measurements at the application level. A genetic algorithm is then applied to this data so as to obtain the optimal model parameters. The disadvantage of this method is that it requires additional resources due to the contained genetic algorithm. Table 2.1 provides an overview of all the above approaches. The table shows that no matter the power device and the model typed used, there is always some disadvantage. One approach attempts to overcome the disadvantages of another, but still there is a new issue introduced. A possible conclusion could be that the modeling approach, in general, has inherent and unresolved weaknesses.

Table 2.1 Survey of some typical power profilers

Profiler	Power Device	Model Type	Disadvantages
PowerTutor	Both	Utilization-based	External factors effect missing, tail power states
DevScope	Internal	Utilization-based	Hidden power states, tail power states
PowerProf	Internal	API-call based	Extra resources
Eprof	External	System-call based	FSM complexity

2.3 Smartphone power management

In order to have a deeper understanding of the power consumption of a mobile device, the first step is to analyze the subcomponents of the device. Usually, A mobile device consists of the hardware components, for example, micro CPUs, network including WIFI and cellular network, storage, cameras including front camera and back camera, the display. Among these subcomponents, display, network and camera are known to consume more energy than other components. These hardware components are the actual energy consumers. In this chapter, we survey these crucial components including display, camera and Bluetooth.

2.3.1 OLED Display

According to the research [11][12], the display is known as the largest power-consuming components in the smartphone. Display along with user interactions form graphical user interaction (GUI) sub-system.

OLEDs (organic light-emitting diodes) are new display technology that have been used in many smartphones to replace LCD display. In the OLED display, each pixel can be decomposed in red, green, and blue sub-pixels and there are no backlights in the OLED display. In this case, the power consumption of each pixel will contribute the total power consumption of OLED display. Thus, compared with LCD display, OLED display consumes more power.

2.3.2 Camera

Camera is another subcomponent that may consume large amount of power when users take pictures and shooting videos [13] [14]. When a camera is running, hardware such as display, image sensor, the CPU when recording, and the storage will work together with camera. In this case, the total power consumption will be larger than expectation. This section will explain how the power is consumed by a camera when the camera function is used by users.

2.3.2.1 Image Sensor

The paper [15] proves that the majority part of power consumption of a camera is a CMOS image sensor. CMOS picture sensors coordinate a few intensifiers specifically into the pixel. This considers a parallel readout design, where every pixel can be tended to independently, or read out in parallel as a gathering. There are two principle sorts of CMOS picture sensor modes, current mode and voltage mode. Voltage mode sensors utilize a readout transistor introduce in the pixel that goes about as a source devotee. The photovoltage is available at the door of the readout transistor, and the voltage read out is a direct capacity of the coordinated photovoltage, to a first request guess. Current mode picture sensors utilize a direct relationship between the door voltage of the readout transistor and the yield current through the transistor to quantify the photocurrent. Compared with CCD(charge-coupled device), a CMOS image sensor have several advantages[16] including low power consumption, low price, high speed imaging and can avoid blooming and smearing effects

2.3.3 Bluetooth

Bluetooth is a widely used short-range communication wireless technology that is integrated on most smartphones nowadays due to its portability. It contains several advantages such as low power consumption [17]. Compared with other wireless technologies, Bluetooth comes with an embedded services discovery mechanisms to support different devices. The transmission rate of Bluetooth devices is generally than 1 Mbps[18].

3. Design of the subcomponent based power consumption measurement experiments

The primary task of this thesis is to determine the relationship between each state variable and power consumption for each relevant hardware component. A possible solution is to use a set of applications to change one variable of a component at a time and meanwhile, keeping all other components constant. This can reduce measurement noise resulting from state transitions by other components. The following chapters provided the experimental design to model display and some details to set the sub-component of the smartphone.

3.1 Experimental Setup

The experiment is conducted in Nexus 6 that runs Android 6.0.1 that provided a particular built-in battery fuel gauge named Maxim MAX17050 Fuel Gauge. With this fuel gauge, we can get the number of power consumption directly by repeatedly reading these two files:

'/sys/class/power_supply/max170xx_battery/charge_counter'

'/sys/class/power_supply/max170xx_battery/charge_counter_ext'

However, we need the super permission of the system to get access to these files. The best way to get the permission is to root the android devices.

3.1.1 Android device rooting

The rooting procedure is done by using the method of [19]. The main two step of rooting is (1) Flashing TWRP and (2) Installing SuperSU. More details can be found in [19].

3.1.2 Controlling subcomponents of mobile devices

In order to derive the power model for power optimization, we measure the subcomponents individually. In other words, to build the power for each subcomponent, we carry out the following experiment for each. Firstly, we hold the power and activity states of all other components constant. Then, we change the activity state to some extreme values for the components that we test, for example, set CPU frequency to its lowest and highest values or configure the GPS state to extreme values by controlling activity and visibility of GPS satellites. For each component, determining the setting that results in extreme power consumption requires some experimentation and knowledge of the component implementation.

3.1.2.1 Controlling CPU Frequency

Different CPU frequency could influence the power consumption of the subcomponent. To change and maintain CPU frequency. We develop an application to control the CPU frequency that shown in figure 3.1.

The first step is changing the CPU governor by modifying the file ‘current_governor’ in the Android Kernel. Next, determining what frequencies are available and setting them. The available frequencies are in the ‘scaling_available_frequencies’ file in the same directory. Finally, Modifying the ‘scaling_setspeed’ to the certain number of frequency and then click the SET

FREQ button. The CPU frequency will be set to the number we expect.

However, the CPU frequency cannot be set less than an absolute number, for example, 149600 in Nexus 6. The reason could be that the kernel is pre-configured to refuse the number that is less than a particular frequency for the CPU. Thus, the minimum number of CPU frequency is 149600 in the experiment.

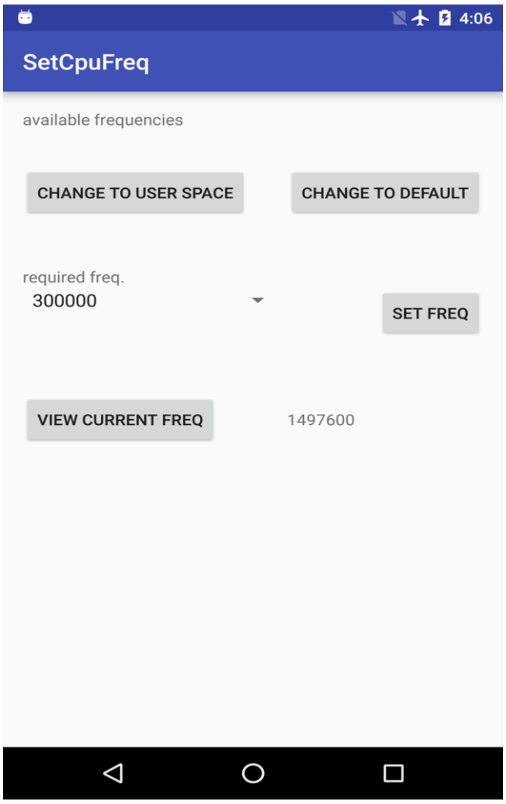


Figure 3.1. Setting CPU Frequency Application

3.1.2.2 Set other sub-components of the smartphone.

The rest component is easier to control than CPU because the super permission is not necessary. Figure 3.2 shows how the sub-components are set in the application.

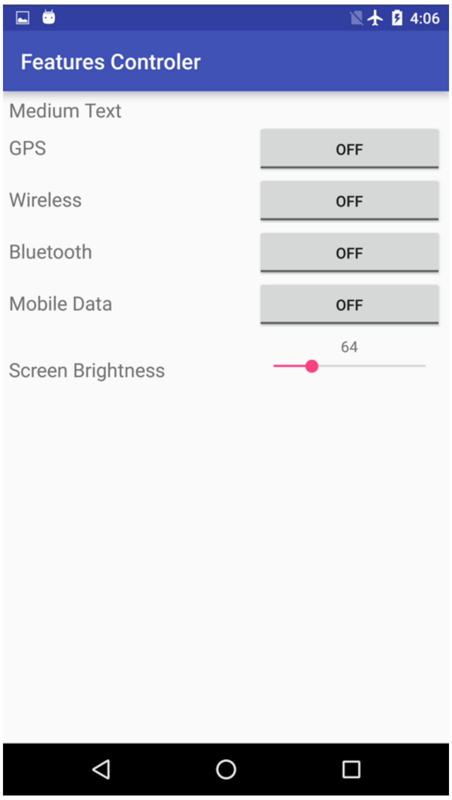


Figure 3.2. Sub-component setting application without CPU

3.2 Display measurement experiment

To evaluate the display power consumption, we set up two experiment. In the first experiment, the display only show white or black color on the entire display using a range of brightness level (0%, 25%, 50%, 75%, 100%). In the second experiment, we set the display with different color (red, green, blue). During the experiment period, the device was in airplane mode and background applications was turned off. The CPU frequency to the lowest or highest values for each experiment. The display power model is derived using an application (figure 3.3) that changes its brightness and turn on/off with discharge duration of 30 minutes for each.

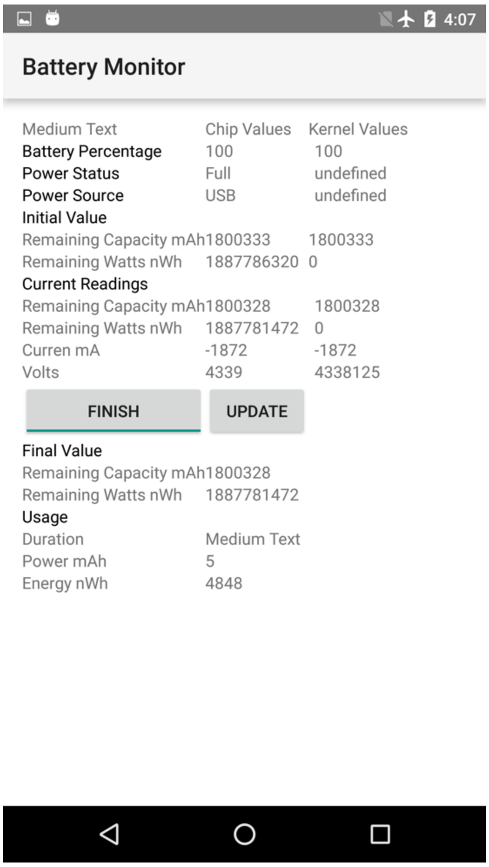


Figure 3.3. The Power consumption measurement application

3.3 Camera measurement experiment

The aim of this experiment is to understand how much power is consumed by the camera when the camera is on and recording a video. we measured different state that can be separate the power drawn by the smartphone into subtasks. The experiment will start from preview mode, and end with recording mode. The experimental program is shown in the figure 3.4.

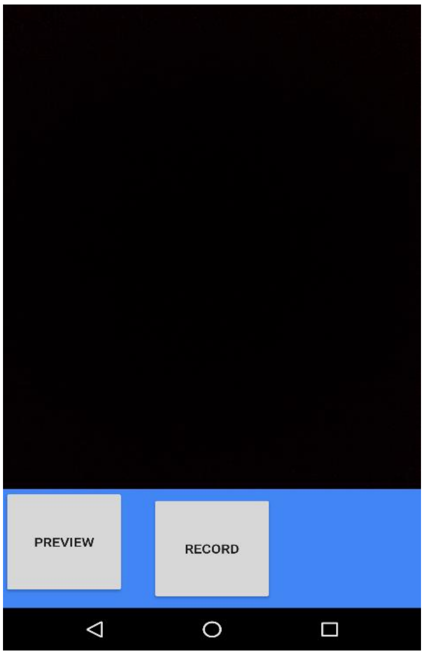
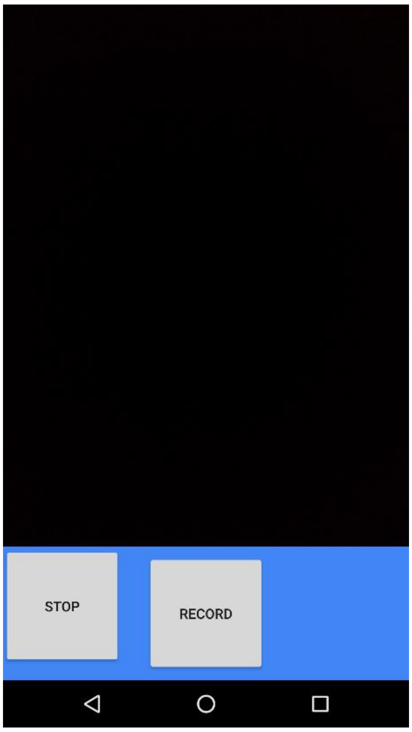


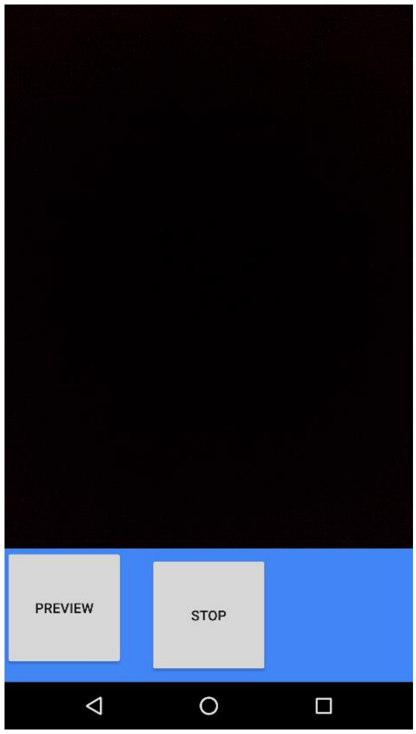
Figure 3.4. The camera experimental application

In this experiment, we divided the camera power consumption experiment into two part, preview mode and recording mode. The preview mode is will switch the camera on but not yet recording video. It causes the camera to focus on the object being shot and display it on the screen. The camera internal hardware is effectively being used in addition to the application that controls camera operations. The recording mode is the time when we record a video. the power drawn by the processing and storing of the video content to the file system is added in this mode.

During the measurement, there were no background applications running and the device was in airplane mode. We chose to display black because black display consumes minimum power but white display consumes maximum power.



(a)



(b)

Figure 3.5. The camera experimental application when recording
(a) Preview Mode. (b) Recording Mode

3.4 Bluetooth measurement experiment

In order to find out how the power consumption of Bluetooth compares to the with a smartphone, we set up the experiment to get measurement results from two Android smartphones. We categories different Bluetooth communication states into six classes to get a precise power measurement results. We set up an experimental application that sequentially uses both kinds of Bluetooth and recorded the power consumption when two devices send or receive a 133MB file when using that application.

(1) Bluetooth switching off. This is the power consumption of the smartphone when we switch off the Bluetooth function, which representing the benchmark of the power consumption. Since the device will remain airplane mode, this state will get the idle power consumption of the device.

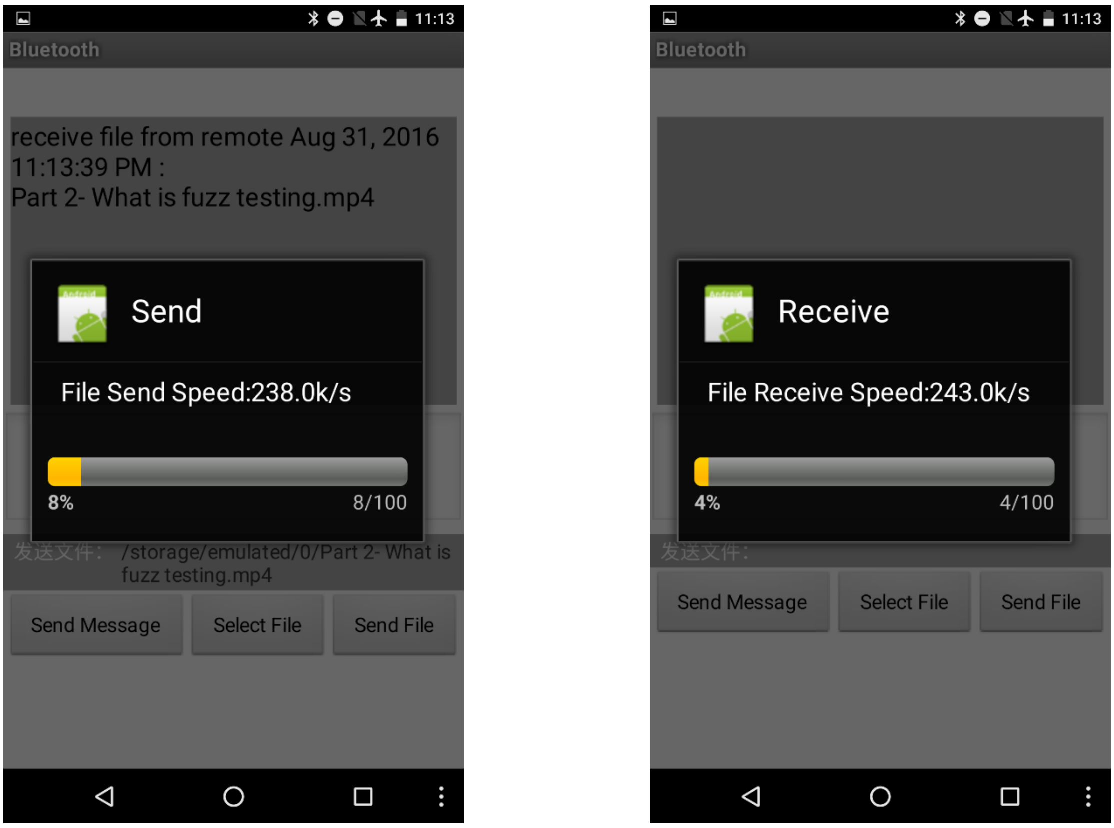
(2) Bluetooth switching on. In this state, we measured the power consumption when Bluetooth is switched on.

(3) Bluetooth discovery. In this state, we measure the power consumption when the mobile device keeps discovering other devices. The amount of time of this procedure varies based on the number of the devices within the search range.

(4) Bluetooth connected but remaining idle. In this state, the mobile device will remain the connected state to another device but no data transmission.

(6) Bluetooth sending: In this state, we measure the power consumption of the smartphone sending data to another smartphone.

(5) Bluetooth receiving: In this state, we measure the power consumption of the smartphone receiving data to another smartphone.



(a) (b)

Figure 3.6. Bluetooth power consumption measurement experiment.
(a) Sending files (b) Receiving files

4. Result and Discussion

This chapter describes the results and some analysis of ten experiments for five brightness level and two CPU frequency.

4.1 Display Power Consumption

As mentioned above, two display experiments are designed to measure different types of power consumption. for display, we are interested in the energy consumed when users are not actively interacting with their smartphones.

4.1.1 Display experiment

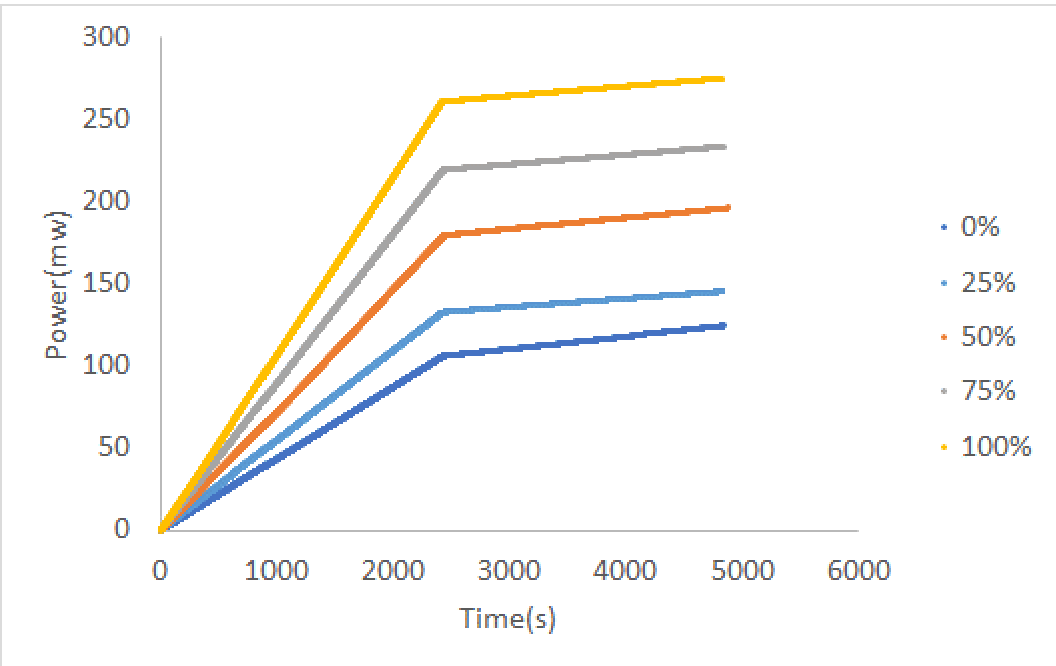


Figure 4.1. Power Consumption for minimum CPU frequency

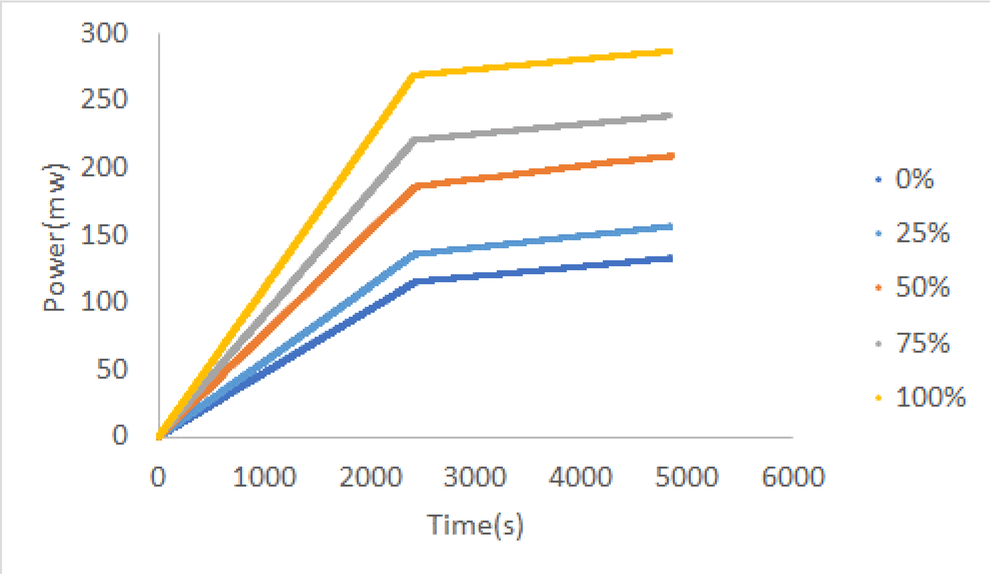


Figure 4.2. Power Consumption for Maximum CPU frequency

Figure 4.1 and 4.2 display the discharge curve (solid line) after executing a routine by different brightness level, which set the CPU frequency at minimum and maximum respectively. The x-axis represents the time in seconds, and the left y-axis is power consumption. The break point of the figure represents the time point between the screen on and screen off.

The result shown in figure 4.1 and 4.2 proves that both the brightness level and screen status(on/off) affects the power consumption of the display. This indicates that when we model the display, we can use brightness level and screen status as the variables.

Figure 4.3 and figure 4.4 provide the total power consumption by different brightness level for the screen on and off. The screen on group remain the constant linear growth of power consumption, but the screen off group seems more irregular.

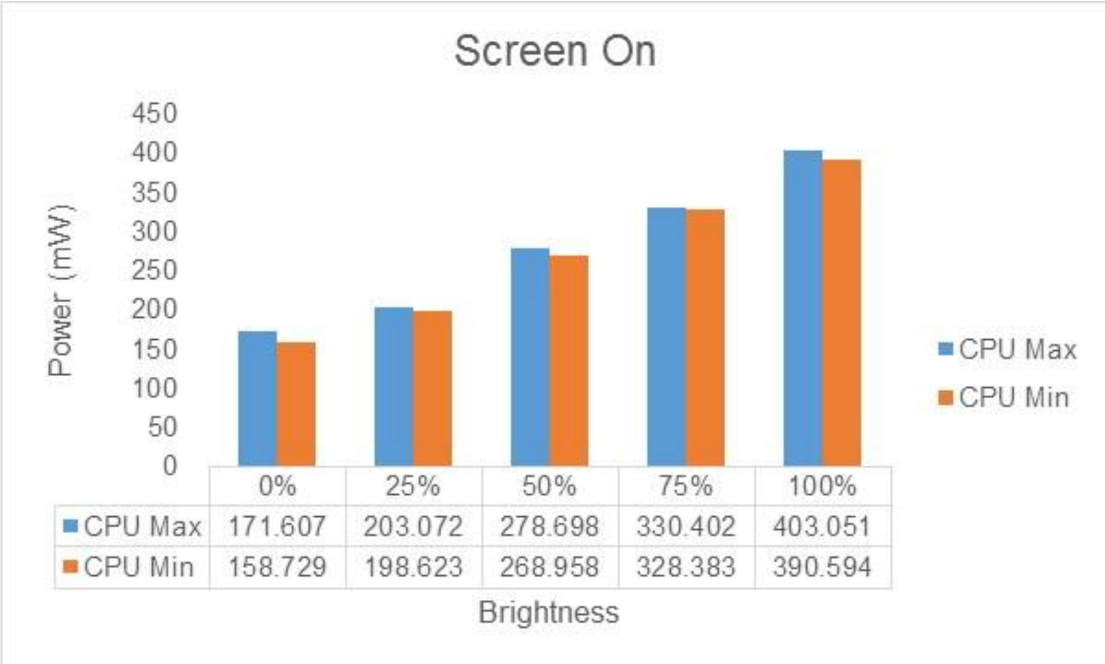


Figure 4.3. Total Power consumption for screen on

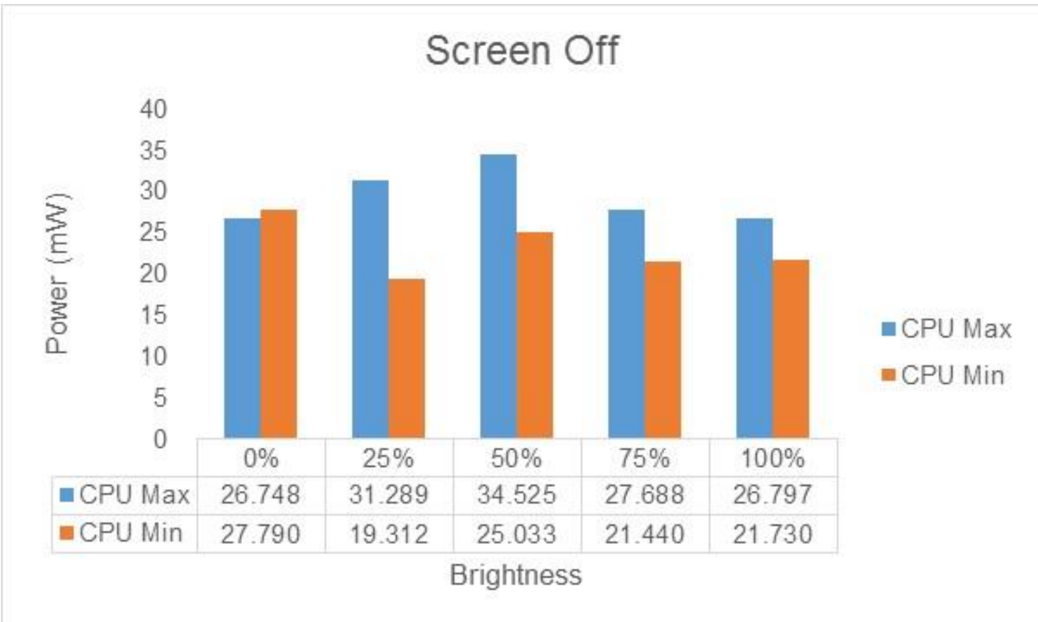


Figure 4.4. Total Power consumption for screen off

Figure 4.3 proves the hypothesis that the higher brightness level, the more power consumption. Moreover, figure 4.4 shows some impressive results that when the screen goes off, the screen is still consuming power. However, power consumption shown in figure 4.4 stays less constant as we can see that CPU Max consumes more power than CPU Min in 0 level group. Also, there is no trend we can get on both the brightness level and CPU frequency direction. This indicates that when the screen is off, the screen will contribute much less power consumption than CPU or memory.

4.1.2 Display color experiment

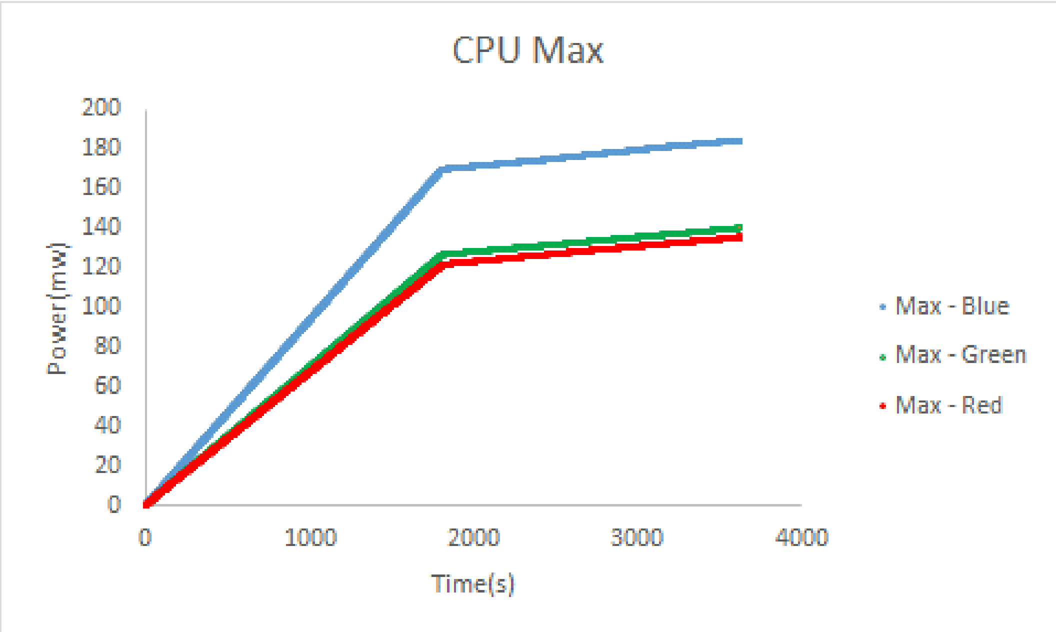


Figure 4.5. Total Power consumption for different color in max CPU Frequency

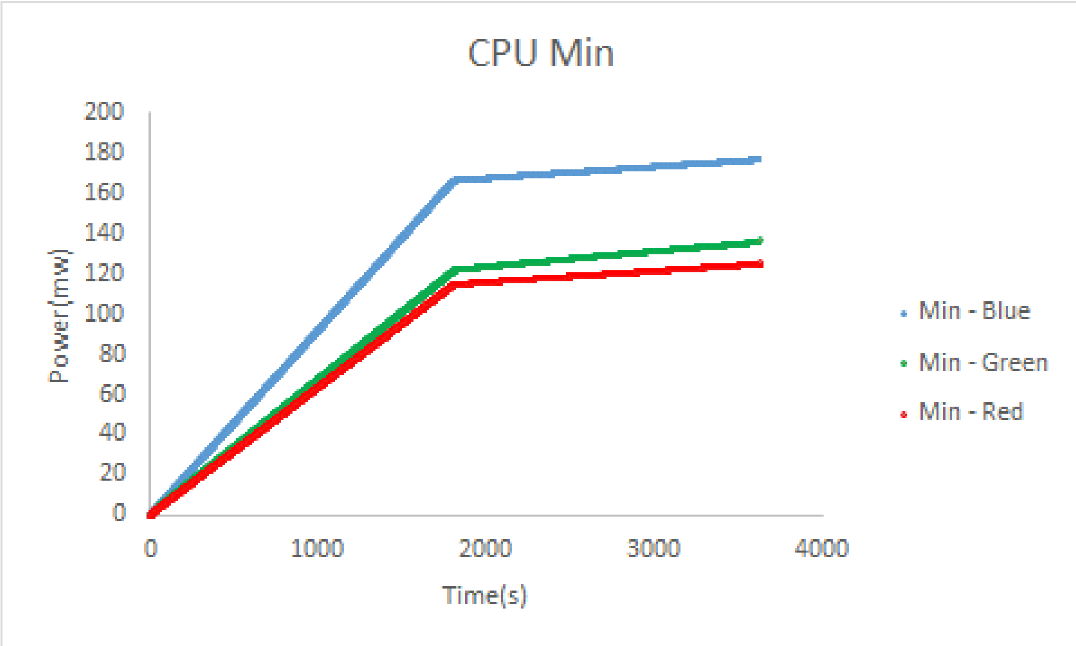


Figure 4.6. Total Power consumption for different color in max CPU Frequency

We can see from figure 4.5, 4.6 that the result of different color displays has been power modeled as a linear function of a standard RGB color scheme, the slope of each function basically shows that the darker the color, the more power is drawn.

4.2 Camera Power Consumption measurement

The discharge curve of camera experiment is shown in figure 4.7.

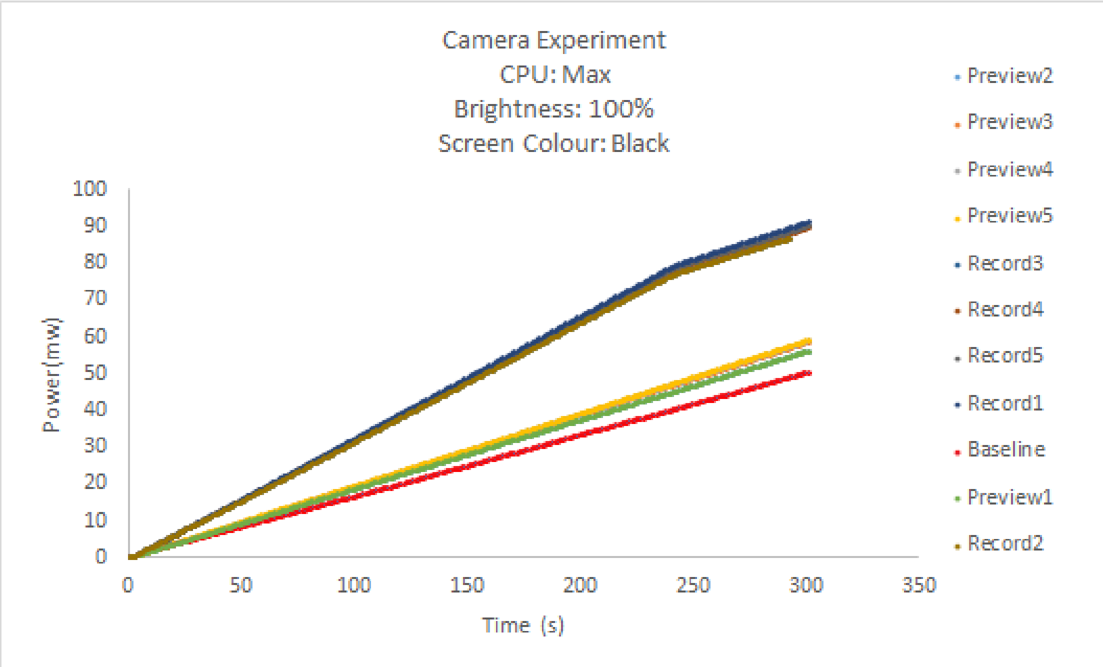


Figure 4.7. Power Consumption for Camera

We can see from the figure that there exists a significant difference between two different modes. The red line shows the benchmark of this experiment, which represent to the power consumption of switching off the camera.

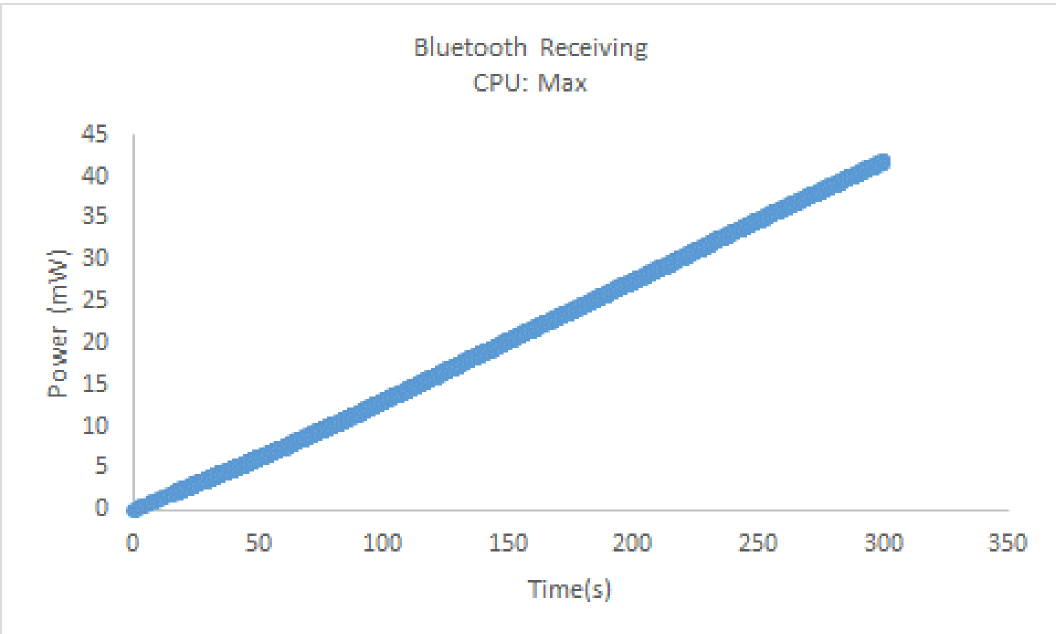


Figure 4.8. Power Consumption for Bluetooth receiving

4.3 Bluetooth Power Consumption measurement

The discharge curve of Bluetooth experiment is shown in figure 4.8 and figure 4.9. We can see from the figure that when Bluetooth receive data, it consumes more power than sending data.

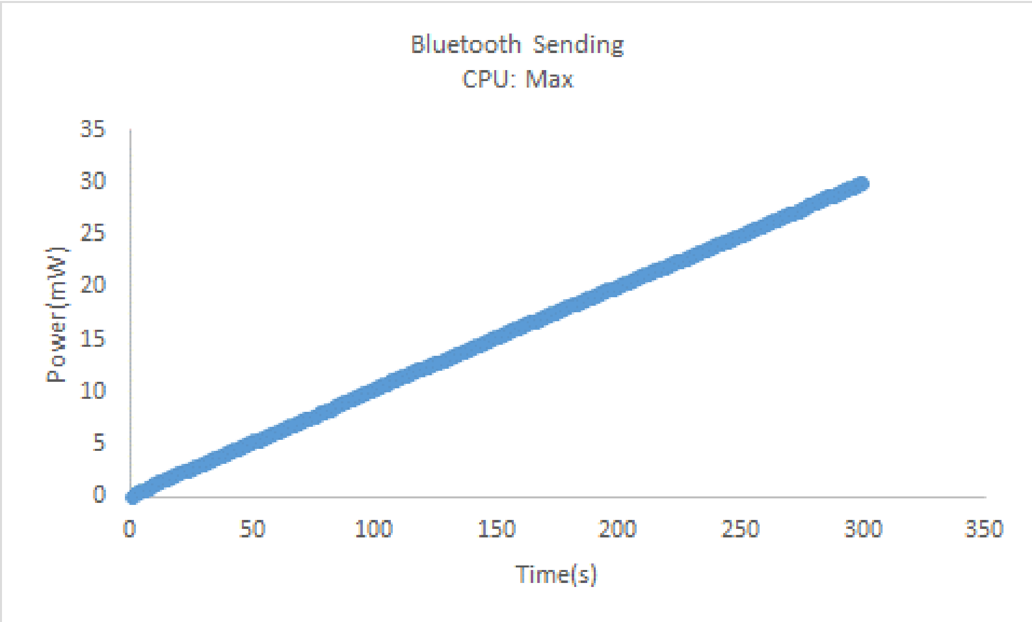


Figure 4.9. Power Consumption for Bluetooth sending

4.4 The Comparison with Power Profile Values

In most Android devices, manufacturers have provided a component based power profile in an XML file that defines the power consumption value for the component and the approximate battery drain caused by the component over time. We can find this profile in the system directory as follows:

platform/frameworks/base/core/res/res/xml/power_profile.xml.

Part of the content of power profile value of Nexus 6 is shown in figure 4.5.

```xml
<?xml version="1.0" encoding="utf-8"?>
<device name="Android">
    <item name="none">0</item>
    <item name="screen.on">170</item>
    <item name="screen.full">440</item>
    <item name="bluetooth.active">30</item>
    <item name="bluetooth.on">3</item>
    <item name="wifi.on">10</item>
    <item name="wifi.active">50</item>
    <item name="wifi.scan">100</item>
    <item name="dsp.audio">25</item>
    <item name="dsp.video">180</item>
    <item name="camera.avg">272.57</item>
    <item name="camera.flashlight">149.28</item>
    <item name="gps.on">70</item>
    <item name="radio.active">350</item>
    <item name="radio.scanning">10</item>
    <array name="radio.on">
        <value>30</value>
        <value>30</value>
        <value>25</value>
        <value>10</value>
        <value>5</value>
        <value>5</value>
    </array>
```


Figure 4.5 Power profile value in Nexus 6

According to [20], the display power profile specifies the mA of current required to keep the display on at minimum brightness and maximum brightness. To determine the power cost (i.e., the battery drained by the display component) of keeping the display on, the framework tracks the time spent at each brightness level, then multiplies those time intervals by an interpolated display brightness cost. However, this file did not provide information about how other subcomponent works when metering one component.

Compared the results generated from the experiment with power profile value. We can see that the ‘screen.on.' value (150) of power profile value is between CPU Max 0 level (171.607) and CPU Min 0 level (158.729). However, the ‘screen. full’ value of power profile value is outside the range of CPU max 100 level (403.051) and CPU Min 100 level (390.594). This could partly because other factors affected the result such as the display color. The result of display color experiment also show that different color will result in different power consumption.

5. Conclusion and Future Work

The main goal of this thesis was to provide a new approach for attributing power consumption to smartphone functionality. The component based power consumption measurement was presented to contribute to power optimization of mobile devices. We measure the key components and our observations included the following states.

(1) Display, Camera Bluetooth exhibits a very significant power draw. Thus, it is important to optimize the power consumption when application designers build an app.

(2) We have shown the efficacy of our proposed ideas by means of extensive experiments throughout the thesis. Application developers will benefit from the results of this thesis to optimize power usage. Also, the results could help users improve their habits of using smartphones.

5.1 Future work

Our research of power consumption measurement indicate that the power consumption of a smartphone relies on which subcomponents are used simultaneously and the operating system that the smartphone is used. However, power consumption measurement only answer how much energy is being consumed by smartphone or a hardware component such as the display. Meanwhile, more information is needed to analyze the factors that affect the power consumption. The main future work is generating a power model with the exist datasets. A power model can be generalized for a certain hardware component, a certain smartphone, or a certain piece of software and can be used for estimating the energy consumption of the hardware or software component.

6. References

[1] http://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

[2] Mittal R, Kansal A, Chandra R. Empowering developers to estimate app energy consumption[C]//Proceedings of the 18th annual international conference on Mobile computing and networking. ACM, 2012: 317-328.

[3] Tarkoma S, Siekkinen M, Lagerspetz E, et al. Smartphone energy consumption: modeling and optimization[M]. Cambridge University Press, 2014.

[4] Hoque M A, Siekkinen M, Khan K N, et al. Modeling, Profiling, and Debugging the Energy Consumption of Mobile Devices[J]. ACM Computing Surveys (CSUR), 2015, 48(3): 39.

[5] Zhang L. Power, Performance Modeling and Optimization for Mobile System and Applications[D]. Northwestern University, 2013.

[6] Pathak A, Hu Y C, Zhang M, et al. Fine-grained power modeling for smartphones using system call tracing[C]//Proceedings of the sixth conference on Computer systems. ACM, 2011: 153-168.

[7] Yang Z. Powertutor-a power monitor for android-based mobile platforms[J]. EECS, University of Michigan, retrieved September, 2012, 2: 19.

[8] Jung W, Kang C, Yoon C, et al. DevScope: a nonintrusive and online power analysis tool for smartphone hardware components[C]//Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software

codesign and system synthesis. ACM, 2012: 353-362.

[9] Pathak A, Hu Y C, Zhang M. Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof[C]//Proceedings of the 7th ACM european conference on Computer Systems. ACM, 2012: 29-42.

[10] Kjærgaard M B, Blunck H. Unsupervised power profiling for mobile devices[M]//Mobile and Ubiquitous Systems: Computing, Networking, and Services. Springer Berlin Heidelberg, 2011: 138-149.

[11] Carroll, Aaron, and Gernot Heiser. An Analysis of Power Consumption in a Smartphone. USENIX annual technical conference. Vol. 14. 2010.

[12] M. Dong and L. Zhong, “Chameleon: A color-adaptive web browser for mobile OLED displays,” in Proc. 9th Int. Conf. on Mobile Systems, Applications, and Services . New York, NY, USA: ACM, 2011, pp. 85–98.

[13] R. Mittal, A. Kansal, and R. Chandra, “Empowering developers to estimate app energy consumption,” in Proc. 18th Annu. Int. Conf. on Mobile Computing and Networking . New York, NY, USA: ACM, 2012, pp. 317–328.

[14] X. Chen, Y. Chen, Z. Ma, and F. C. A. Fernandes, “How is energy consumed in smart-phone display applications?” in Proc. 14th Workshop on Mobile Computing Systems and Applications, ser. HotMobile ’13. New York, NY, USA: ACM, 2013, pp. 3:1–3:6.

[15] R. LiKamWa, B. Priyantha, M. Philipose, L. Zhong, and P. Bahl, “Energy characterization and optimization of image sensing toward continuous mobile vision,” in Proc. 11th Annu. Int. Conf. on Mobile Systems, Applications, and Services, ser. MobiSys ’13. ACM, 2013, pp. 69–82.

[16] Perrucci, Gian Paolo, Frank HP Fitzek, and Jörg Widmer. "Survey on energy consumption entities on the smartphone platform." Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd. IEEE, 2011.

[17] M. O. Khan, V. Dave, Y.-C. Chen, O. Jensen, L. Qiu, A. Bhartia, and S. Rallapalli, “Model-driven energy-aware rate adaptation,” in Proc. 14th ACM Int. Symp. on Mobile ad hoc Networking and Computing. New York, NY, USA: ACM, 2013, pp. 217–226.

[18] C. Drula, C. Amza, F. Rousseau, and A. Duda, “Adaptive energy conserving algorithms for neighbor discovery in opportunistic Bluetooth networks,” IEEE J. Sel. Areas Commun., vol. 25, no. 1, pp. 96–107, Jan. 2007.

[19] https://source.android.com/devices/tech/power/#profile-values

[20] https://www.kingoapp.com/root-tutorials/how-to-root-android-6.0-marshmallow.htm