UNIVERSITY OF AGDER

A battery and Network Usage Model for Smartphones

by

Ali Raza

Thesis submitted in practical fulfillment of the requirements for the Master Degree in

Information and Communication Technology

Faculty of Engineering and Science
University of Agder

Grimstad
1st June 2012

Keywords: Smartphone, Android, Power Consumption, Measurement, Model

Abstract

Smartphones have emerged into platforms with powerful computational capabilities that generate large amount of data. Smartphones have become an important part of our daily life and we use smartphones more frequently than we used desktop computers to stay connected on internet, reading news, playing games, browsing, watching video and staying connected with friends through social networking websites like Facebook and Tweeter. On the other hand the smartphones have a strict energy budget and limited lifetime on a single charge. Thus it very important, for manufacturers and users, that smart devices and smartphones and the applications running on smartphones, must be very energy efficient.

For this purpose it is very important to understand how and where the energy is used in a smartphone. Also how much of the smartphones' energy is consumed by what application and under what circumstances.

In this thesis we have considered a few popular and commonly used applications on smartphones to analyze their power consumption and then develop a power consumption model for smartphone while running those applications. The platform we studied is Android and the Device under Test (DUT) is Samsung Sidekick 4G and Google Nexus One smartphone. Android is the most famous OS for smartphones that can be defined as, "a software stack for mobile devices that includes an operating system, middleware and key applications".

We relied on three software tools for power consumption measurements of the applications running on smartphones.

We have developed a power model for five commonly used applications on smartphone namely: Skype, Viber, Tango, YouTube and Facebook. We have analyzed the energy usage by these applications so that developers would focus on for further improvements of power management.

Version Control

Version	Status	Date	Change
0.1	Draft	2012 Apr 15	Report structure and chapter one
0.2	Draft	2012 Apr 23	Chapter one
0.3	Draft	2012 May 4	Chapter two
0.4	Draft	2012 May 12	Chapter one and two
0.5	Draft	2012 May 16	Chapter three
0.6	Draft	2012 May 19	Chapter three and four
0.7	Draft	2012 May 24	Chapter three and four
0.8	Draft	2012 May 26	Chapter three, four and five
0.9	Draft	2012 May 28	Chapter six
1.0	Final	2012 May 31	Chapter three, four and chapter six

Preface

This report documents a Master's thesis in Information and Communication Technology, Faculty of Engineering and Science, University of Agder Norway. The master project has been carried out from 1st January to 31st May 2012.

I would like to thank my supervisor Prof. Frank Yong Li at University of Agder, for the insightful guidance of my work, active involvement and helpful support at all times during the project.

The thesis has been initiated by ST-Ericsson. I would especially thank Jonny Ervik and Stian Haugen at ST-Ericsson for their insightful support, many fruitful discussions and successful cooperation.

I would also like to thank Hossein Falaki for allowing us to use SystemSens and access the CENS server.

Special thanks to my parents and wife Fozia for supporting me in every step of my life, having confidence in me and thereby making the master thesis possible.

Grimstad, 1st June, 2012

Ali Raza

Contents

List of Figures

List of Tables

Chapter 1

1. Introduction

The use of smartphones is becoming popular and their market share is increasing rapidly. It is because smartphone devices have emerged into platforms with powerful computational capabilities and equipped with multitudes of sensor and capable of generating vast amount of data. On the other hand smartphones operate on strict energy budget and therefore have a limited lifetime on a single charge [14]. Therefore a deep analysis of usage pattern and power consumption of different components of smartphone is critical for efficient power management of smartphones.

1.1 Background

A smartphone can be defined as a mobile phone with advanced computational capability, often with PC-like functionality and connectivity, with ability to download applications. The smartphones have the capability to take & display photos and videos, watch TV, play games, check and send e-mail, and surf the web. Modern smartphones can run third-party applications, which provides limitless functionality.

Figure 1.1: Smartphone

And thus ultimately making smartphone a simple choice for consumers once mostly used by business users. With advanced features, smartphone has changed the concept of a mobile phone. Since smartphones have a wide range of functionality, they require advanced software, similar to a computer operating system (OS). The smartphone software handles phone calls, runs applications, and provides configuration options for the user. Most smartphones include a USB connection, which allows users to sync data with their computers and update their smartphone software. The most common mobile operating systems used by modern smartphones include Apple's iOS, Google's Android, Microsoft's Windows Phone, Nokia's Symbian, RIM's BlackBerry OS, and embedded Linux distributions such as Maemo and MeeGo.

Today smartphones generate a considerable fraction of mobile networks' and internet's traffic [19]. Smartphone users stay connected with friends and family and share status, photos and videos using many social websites like facebook etc and also are always part of smartphone network. According a recent report on January 17, 2012, mobile application usage trumps web browsing at 94 minutes a day.

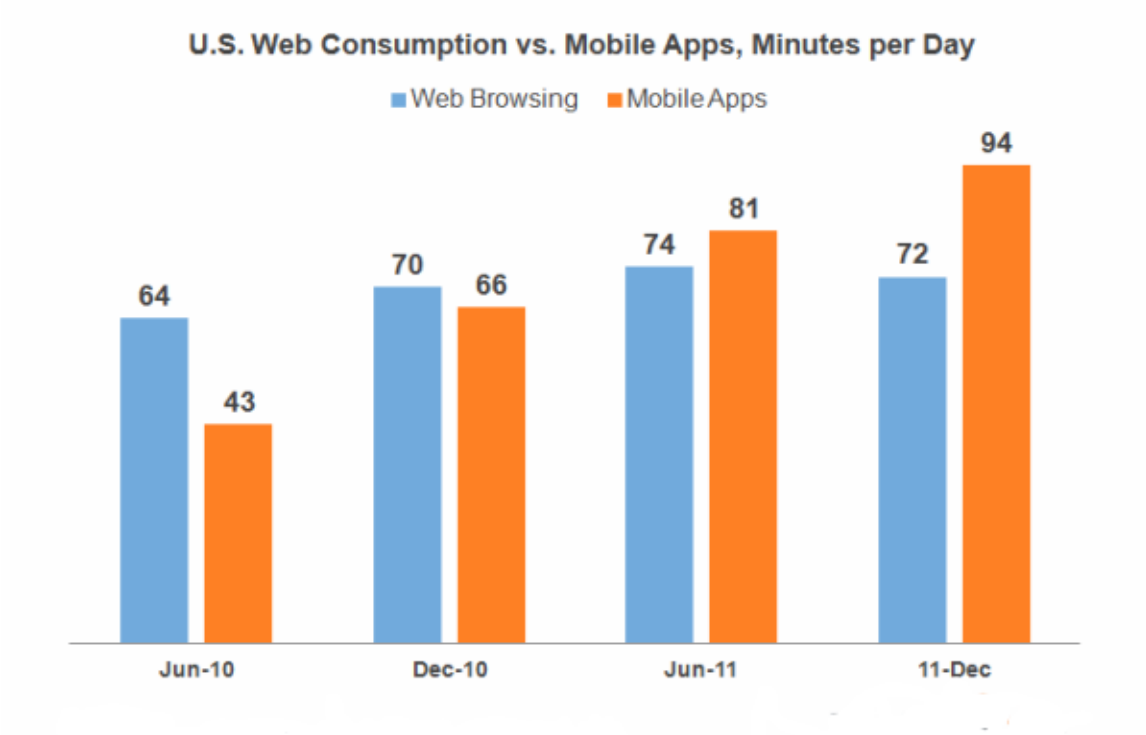

Figure 1.2: Web consumption vs. Mobile Apps, Minutes per Day

Those fingers are flying over mobile keyboards with people now spending, on average, 94 minutes per day using mobile applications, according to analytics firm Flurry [25].

A smartphone network can be defined as "*a set of smartphone devices that communicate in an unobtrusive manner, without explicit user interactions, in order to realize a collaborative or social task*". One example is road traffic delay estimation [12] using WiFi beams collected by Smartphone devices rather than invoking energy-demanding GPS acquisition. On the social site, Google Latitude enables users to track the places they and their social network have visited [1]. In this regards, research and work has been done in performance management of mobile networks. Similarly improvement in power management mechanism of smartphones is also important due its limited battery energy budget.

1.2 Problem Statement

The main task of this thesis is to analyze power consumption of different components and applications in a smartphone. On the basis of the measurements and calculations done we will try to make a usage model for some popular applications used in smartphones.

Firstly, detailed analysis and understanding of traffic types and patterns has to be done and also power consumption of different components and modes are important to understand. Since a main requirement of effective and efficient power management model is good understanding of where and how the energy is used; how much of battery's power is consumed by which part of the system and under what circumstances.

Since the smartphone under study is an Android smartphone therefore a deep understanding of Android Power Management is critical. Android is based on Linux, but does not use a standard Linux kernel. Also the Android power management is built on the top of standard Linux Power Management (PM) but takes a more aggressive policy to manage and save power.

Secondly, some measurements of overall system power consumption and power consumption of device's main components and applications has to be carried out. For this purpose, it was suggested to use Development Kit and a smartphone with a specific application that would be provided by ST-Ericsson to do the measurement task. But later on we decided to use a smartphone and an appropriate application only.

Finally, based on measurements, developing a usage model based on Android smartphone, for some popular and commonly used applications, and then to verify with real-world scenario using an Android smartphone.

1.3 Motivation

Recently, mobile traffic has increased tremendously due to the deployment of smart devices such as smartphones. The smartphones run a wide range of applications and also use various types of access networks such as 3G and WiFi etc. Hence there is growing need to manage these smart devices and also the mobile network.

Most recent research studies were focused on performance of the device and network [1] [8, 9] [11]. Some other research was mainly focused on the energy-delay tradeoffs in smartphone applications [16-18]. Very few research works are focused on battery management and usage pattern [13] [15]. So this thesis is supposed to supplement existing research on power management of smartphones.

The thesis is initiated by ST-Ericsson. This thesis is tailored to investigate the power consumption of different components and applications of an Android smartphone. For this we will be relying on measurements that we do using a suitable application. If needed, we will employ a circuit to determine the power consumption of Android smartphone.

1.4 Research Approach

While working on this thesis, a step-by-step approach has to be adopted. After a thorough study of power management of Android we will choose an appropriate application to that would help us measure the power consumption of main components and applications in an Android smartphone. In case that a single application cannot be used to measure and get the desired data, we will be using another one or two applications that may cover specific aspects of Android Smartphone's power consuming components and applications to be measure.

We verify the measurements we will be using the smartphone repeatedly to ensure the accuracy and correctness of the measurements. We may also employ a circuit to complete the measurement task.

1.5 Limitations and Assumptions

There are some assumptions during the problem solution, which need to be confirmed and mentioned firstly,

- Since we are using some software applications for the power measurement of different popular and commonly used applications therefore it is assumed that the measurement results are accurate or at least with an acceptable accuracy.
- The battery of the device under test (DUT) is normal.
- During many measurement procedures of some applications, we used wireless (WLAN) at different locations within campus and at home. We assume that signal strength of WLAN was the same at all locations.
- Since we using many (three) software applications for power measurement of different components and applications, we assume that the measuring accuracy of all software tools are alike.

- It is assumed that the investigation can be used in real life. So the investigation and measurements can provide applications developers with a reference to improve the existing applications in use or develop new energy efficient applications.
- We are not modeling power consumption of overall system of DUT.
- We are only limited to measure and model power consumption of some popular and commonly used application.

1.6 Thesis Outline

The outline of the thesis is as below

- Chapter 1 gives an introduction to the thesis. Based on the relevant background, the problem statement is presented with the motivation behind it. The assumption and limitation are also discussed.
- Chapter 2 provides technical theory of Android system in general and power management of Android in particular. It discusses how Android is different from Linux. It also discusses the main power consuming components within a smartphone. This chapter also introduces the software we are using for measuring power consumption.
- Chapter 3 deals with the measurement results. Based on power measuring software and applications, the measured results are collected and analyzed.
- Chapter 4 describes the proposed model based on the power consumption measurements carried out.
- Chapter 5 is discussion and an analysis of the thesis.
- Chapter 6 deals the possible future work based on the thesis.

Chapter 2

2. Theoretical Background

Android is an open source mobile device operating system developed by Google based on the Linux 2.6 kernel. The Linux kernel was chosen due to its proven driver model, existing drivers, memory and process management, networking support along with other core operating system services [2]. In addition to the Linux kernel various libraries were added to the platform in order to support higher functionality. Many of these libraries originate from open source projects; however the Android team created their own C library, for example, in order to resolve licensing conflicts. They also developed their own Java runtime engine, optimized for the limited resources available on a mobile platform called the "Dalvik Virtual Machine." Lastly, the application framework was created in order to provide the system libraries in a concise manner to the end-user applications [3].

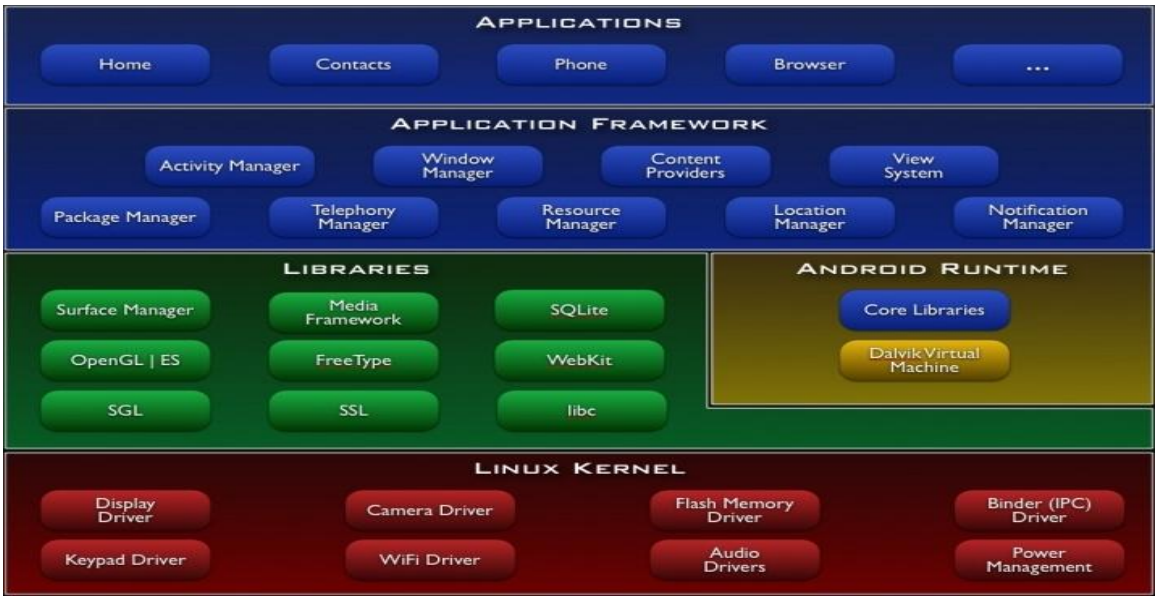

Figure 2.1: Android architecture

The Linux kernel supports many different target architectures. However only two are fully supported by Android at this time: x86 and ARM. The ‘x86 architecture’ for Android is mainly targeted at Mobile Internet Devices (MIDs) whereas the ARM platform is prevalent on mobile phones. Despite no one manufacturer dominating the smartphone segment, they all primarily use one architecture, ARM [3].

One of the main reasons behind the widespread popularity of the ARM platform is due to its focus on power saving features.

Android is based on the Linux, but does not use a standard Linux kernel. The kernel enhancements of Android include *alarm driver*, *ashmem* (*Android shared memory driver*), *binder driver* (Inter-Process Communication Interface), *power management*, *low memory killer*, *kernel debugger* and *logger*.

2.1 Android Power Management

Android Power Management (PM) is built on the top of standard Linux PM and takes more aggressive policy to manage and save power. In order to reduce wasted power, multiple hardware power saving features are employed by Linux such as clock gating, voltage scaling, activating sleep modes and disabling memory cache. Each of these features reduces the system's power consumption at the expense of latency and/or performance. These tradeoffs on a Linux system are managed by either Advanced Power Management (APM) or Advanced Configuration and Power Interface (ACPI).

APM is an older, simpler, BIOS based power management subsystem, which is still used on older systems. Newer systems use ACPI based management instead. ACPI is more operating-system centric [7] than APM and also offers more features such as a tree structure for powering down devices so that subsystem components are not turned off before the subsystem itself.

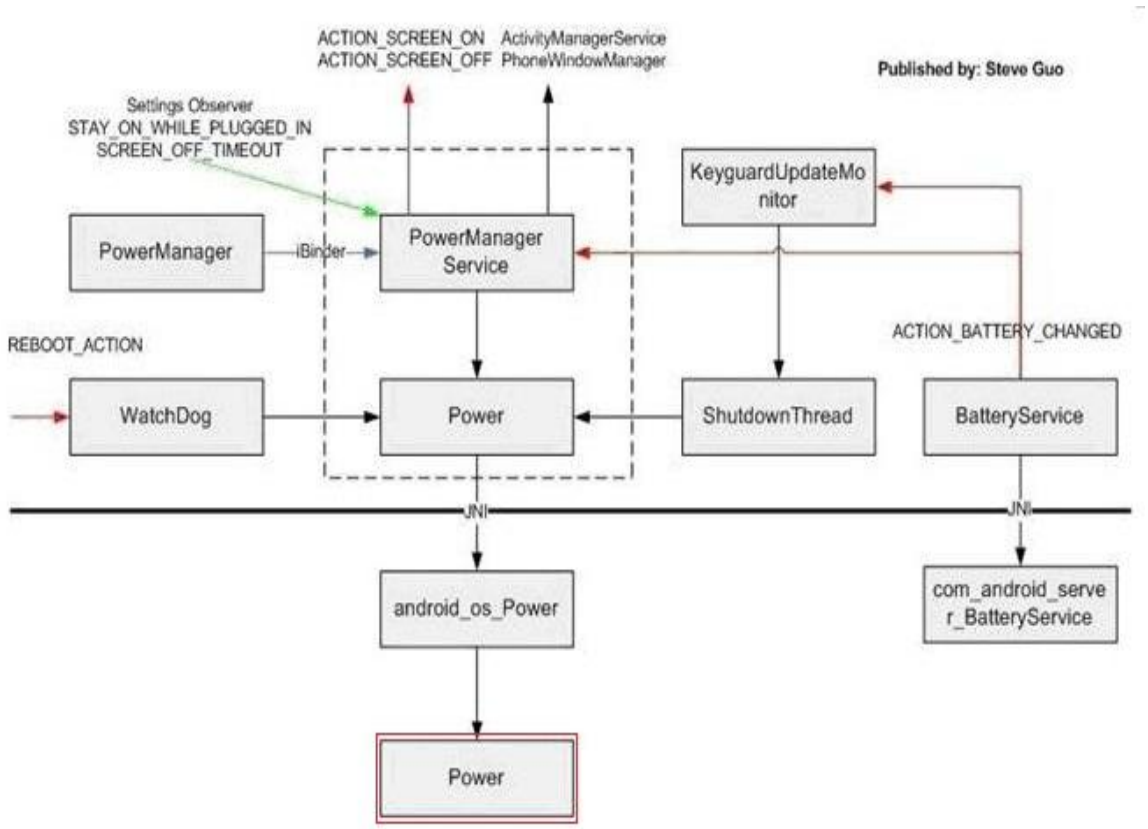

Figure 2.2: Android Power Management

In contrast with a standard Linux system, Android does not use APM, nor uses ACPI for power management. As shown in Figure 2.2, Android instead has its own Linux power extension, PowerManager instead. The core power driver (Shown at the bottom of Figure 3 as "Power") was added to the Linux kernel in order to facilitate this functionality.

This module provides low level drivers in order to control the peripherals supported by the Power Manager. These peripherals currently include: screen display and backlight, keyboard backlight and button backlight. Each peripheral's power is controlled through the use of WakeLocks. These locks are requested through the API whenever an application requires one of the managed peripherals to remain powered on (Each lock setting shown in Table 2.1).

Flag Value	CPU	Screen	Keyboard
Partial Wake Lock	On*	Off	Off
Screen DIM Wake Lock	On	Dim	Off
Screen Bright Wake Lock	On	Bright	Off
Full Wake Lock	On	Bright	Bright

**If you hold a partial wakelock, the CPU will continue to run, irrespective of any timers and even after the user presses the power button. In all other wakelocks, the CPU will run, but the user can still put the device to sleep using the power button.*

Table 2. 1: Wake Lock Settings [6]

If no wake lock exists which "locks" the device, then it is powered off to conserve battery life. In the case of multiple power settings the transition is managed through the use of delays based on system activity. A sample of this behavior is shown in Figure 4 for the screen backlight.

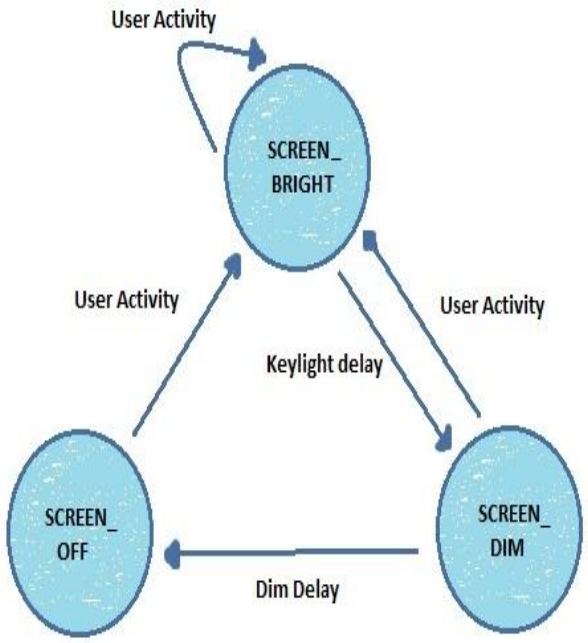

Figure 2.3: PM State Machine (Screen) [10]

In addition to WakeLocks the PowerManager also monitors the battery life and status of the device. This service coordinates with the power circuitry charging in the battery and also powers down the system when the battery reaches a critical threshold [3].

The main components in a smartphone that consume power significantly are CPU, WiFi and LCD. The table below gives energy anatomy of a smartphone (Android based HTC Hero 2.1) [14]

Basic Operation on Smartphone	Power (mW = mJ/s)
CPU Idle (OS running)	175 mW
CPU Busy (Processing)	369 mW
WiFi Idle (Connected)	38 mW
WiFi Busy (Uplink 123Kbps, -58dBm)	600 mW
3G Busy	800 mW
LCD Bright. (low,hig)	300-900 mW
Function ($len(trace) = 100K$ 18B points)	**Time**
$Transmit(trace, server)$ (from Smartphone)	112 seconds
$Compare(query, trace)$ (on Smartphone)	111 seconds

Table 2.2: Energy Anatomy of Smartphone [14]

2.2 Applications/Tools

In order to measure the power consumption of different components and applications in android phone we analyzed many applications. After detailed studies we chose three applications that are SystemSens, Battery Monitor and Power Tutor. It is because none of the applications fulfill our requirements of measuring the power consumption. There was tradeoff among all applications that we used.

Below is a brief introduction of applications that we used to measure the power consumption by different applications in an Android smart phone.

2.2.1 SystemSens

SystemSens is a tool developed at the Center For Embedded Networked Sensing (CENS) to log detailed system events on Android smartphones. SystemSens keeps the logged records in a local database on the phone, and regularly uploads them to a CENS server [5].

The most energy and resource consuming task of the SystemSens client is uploading the records to the server because of high energy consumption associated with network

transmission. The client records and uploads a range of operating system events. Each group of related OS information is recorded by a virtual "sensor."

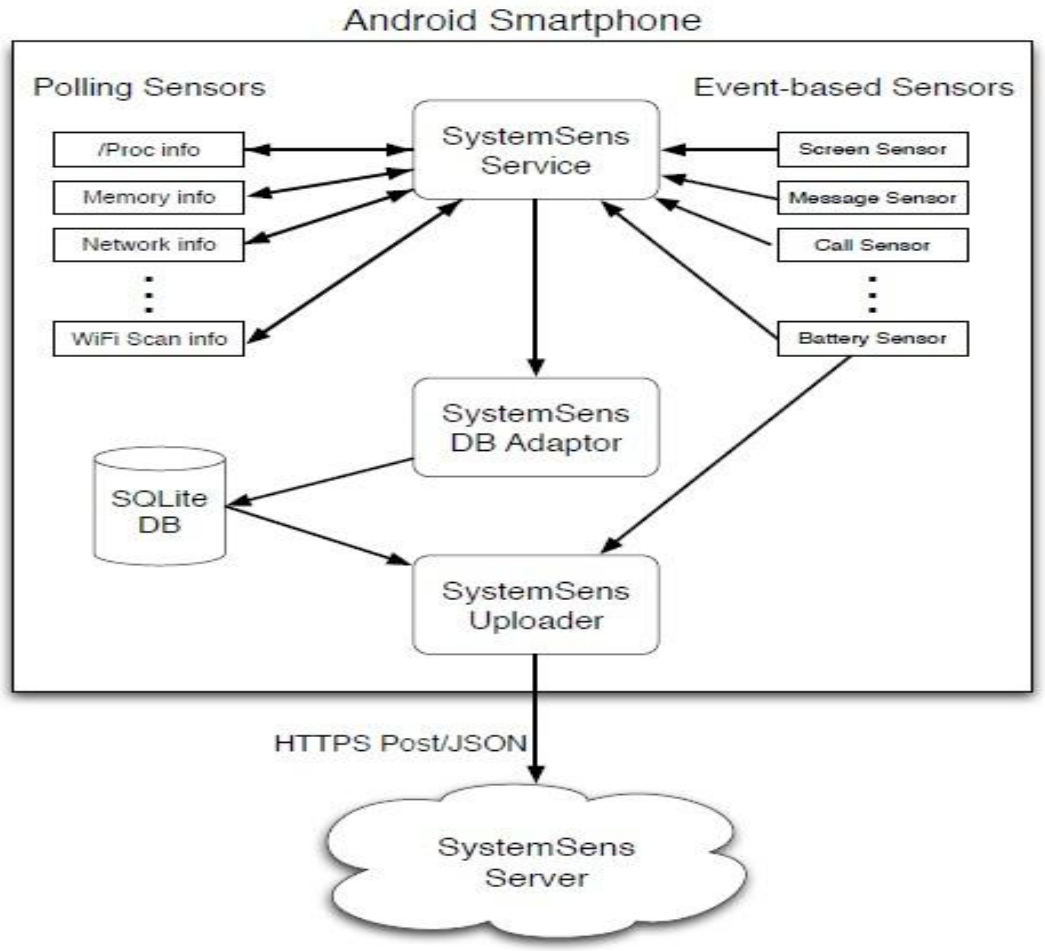

Figure 2.4: Architecture of SystemSens Client Application [20]

Following is the list of information that SystemSens logs:

- Battery information (level, voltage, current, temperature, health, charging status):
- Screen status events (on and off) and brightness
- Application usage times

- Network traffic per application
- CPU and memory usage per application
- WiFi interface status changes and signal strength
- Cellular interface status changes including cell location and signal strength
- Voice call status
- Text message events

2.2.2 Battery Monitor

Battery Monitor is basically a battery monitoring application with notification icon, history, graphics and alarms. It Shows historical data (%, mA, mW, mV and temperature), calculates estimated run-times and battery aging, helps calibrate battery, and improves the battery run-time. It also saves log file to the SD Memory Card in the smartphone. It saves battery voltage, temperature and current at an interval of 60 seconds.

Figure 2.5: Battery Monitor User Interface [26]

2.2.3 Power Tutor

PowerTutor was developed by University of Michigan Ph.D. students Mark Gordon, Lide Zhang and Birjodh Tiwana under the direction of Robert Dick and Zhuoqing Morley Mao at the University of Michigan and Lei Yang at Google.

PowerTutor is an application for Google phones that displays the power consumed by major system components such as CPU, network interface, display, and GPS receiver and different applications. The application allows software developers to see the impact of design changes on power efficiency. Application users can also use it to determine how their actions are impacting battery life. PowerTutor uses a power consumption model built by direct measurements during careful control of device power management states. This model generally provides power consumption estimates within 5% of actual values. A configurable display for power consumption history is provided. It also provides users with a text-file based output containing detailed results.

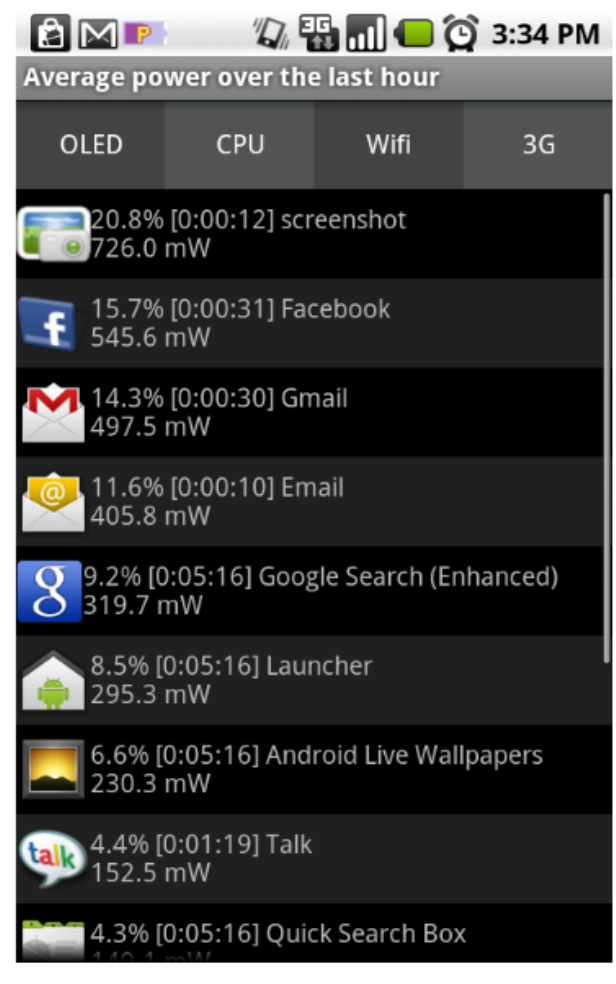

Figure 2.6: Power Tutor User Interface [4]

PowerTutor can also be used to monitor the power consumption of any application.

PowerTutor's power model was built on HTC G1, HTC G2 and Nexus one. It will run on other versions of the Google Phone (GPhone), but when used with phones other than the above phone models, power

consumption estimates will be rough [4].

In the initial stage of master thesis our main focus was on SystemSens since the features of this application were close to our requirements. One of the main problems was that all the data was uploaded on the SystemSens server. We had an ID and password to access our data and get graphs etc. As to be discussed in Chapter 5, we had to rely on some other applications due to some problems in getting our log data.

Chapter 3

3. Power Consumption Measurements

Using the applications mentioned in chapter two we measured the overall power consumption and the power consumption of different applications running on Android phone. The Device Under Test (DUT) were Samsung Sidekick 4G and HTC Google Nexus one.

3.1 Measurements using SystemSens

SystemSens is designed to be unobtrusive --- it has no user interface to minimize impact on usage, and it has a small footprint in terms of memory, CPU and energy consumption. Phone, SystemSens consumes about 19mW just for recording the polling senors [20]. The DUT for these measurements was Samsung Sidekick 4G. Following graphs are the measurements performed using SystemSens.

Figure 3.1: Battery voltage

Figure 3.1 shows the voltage of the battery of DUT. In this measurement DUT is Samsung Sidekick 4G and measurement duration is 48 hours. The graph shows the voltage drop while using different applications.

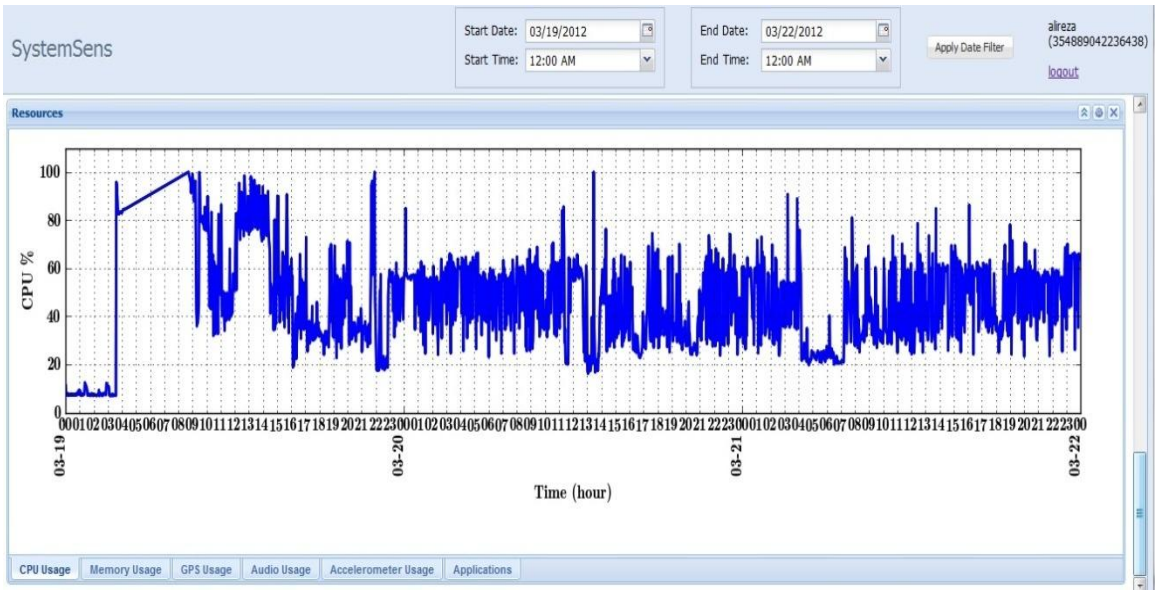

Figure 3.2: CPU usage percentage

Figure 3.2 shows the percentage of the CPU usage for duration of three days. CPU consumes energy while using some applications and this use is not linear.

Figure 3.3: Memory usage percentage

Figure 3.3 shows the memory usage of DUT in MBs. The measurement duration is three days and DUT is Samsung Sidekick 4G. On average, memory usage is around 200 MBs. And different applications occupy memory differently.

Figure 3.4: Network Bytes

Figure 3.4 shows the size of received and transmitted cellular and WiFi data. As it is evident thatwe used WiFi and cellular data is almost nil. It is because we were using WiFi and not using cellular network.

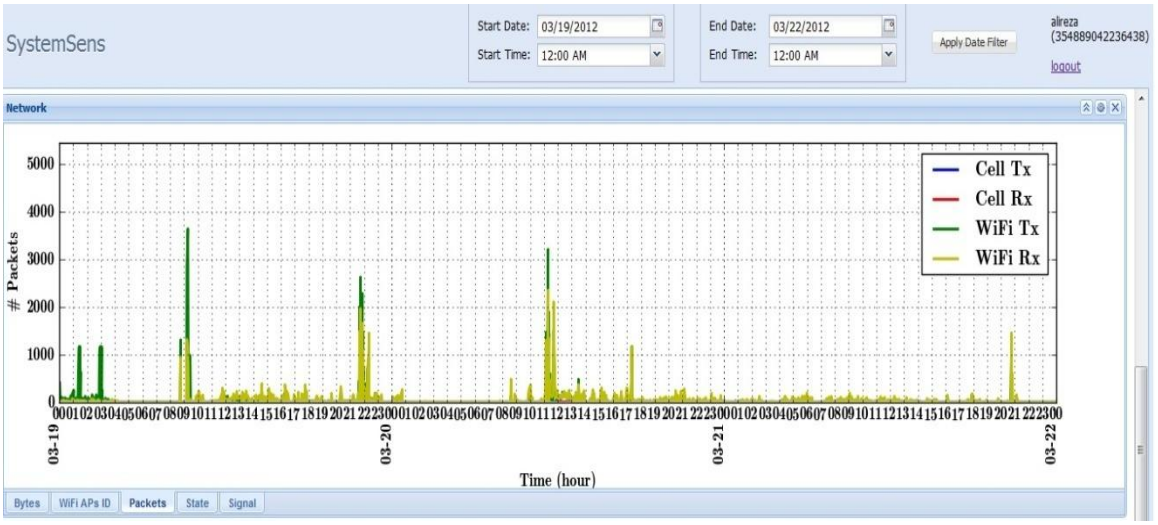

Figure 3.5: Network Packets

Figure 3.5 shows the number of received and transmitted packets of cellular and WiFi data. Here again cellular packets are nil since we didn't use it. It is obvious that most conversation or application use occurs during day time there during night the number of packets are almost nil.

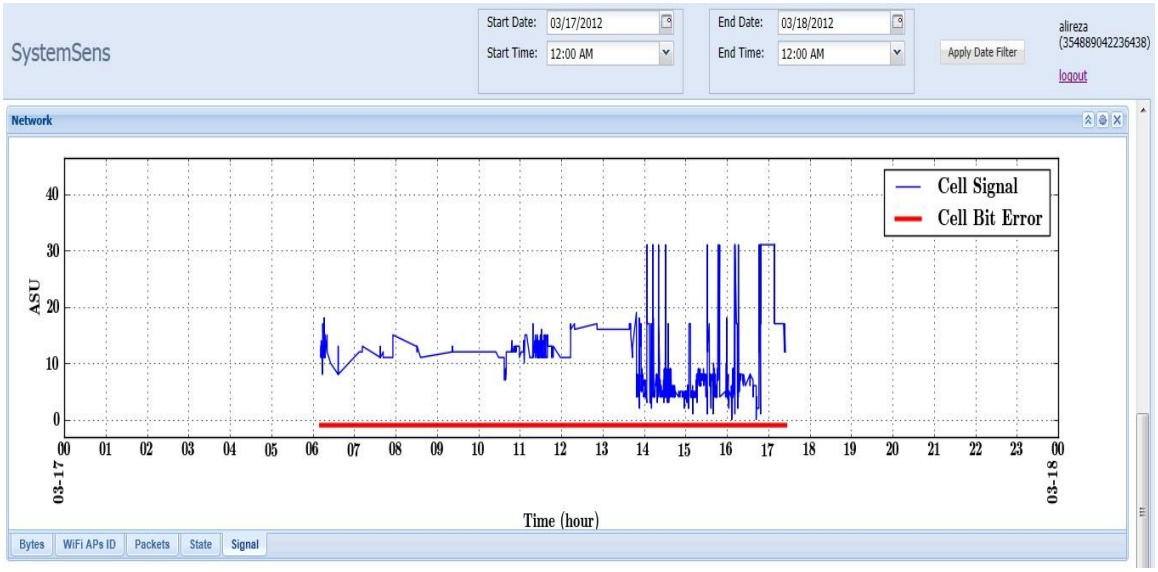

Figure 3.6: Network Signal

Figure 3.6 shows the network signal and Cell Bit Error. The measurement duration is eleven hours. ASU or "Arbitrary Strength Unit" is an integer value proportional to the received signal strength measured by the mobile phone. It differs depending on distance from the BTS or the obstacles in between the BTS and smartphone.

It is widely misbelieved that ASU = "Active Set update". The Active Set Update is a signaling message used in handover procedures of UMTS and CDMA mobile telephony standards. On Android phones the acronym ASU has nothing to do with Active Set Update. There are many GSM and UMTS Signal Monitoring applications available for the Android operating system [31].

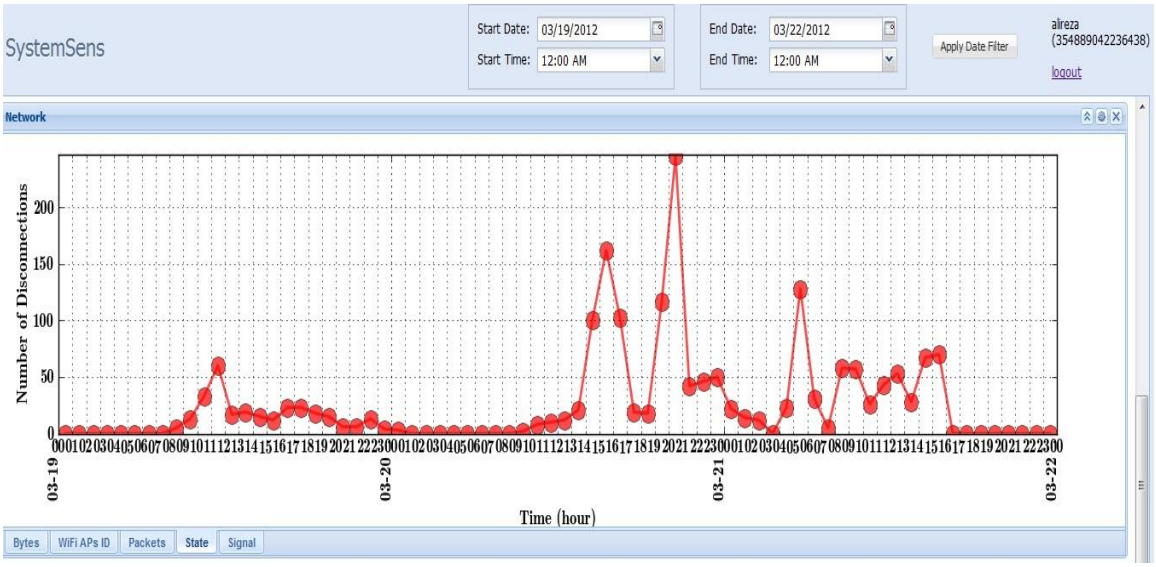

Figure 3.7: Network State

Figure 3.7 shows the network state and the measurement duration is three days. It is evident from the figure that at some hours during the day, the number of disconnections at high. It may possibly be due to weak signal strength or the user is at a location where the signal is not strong enough. Also if user is in a moving car, the number of disconnections may increase due to varying obstacles.

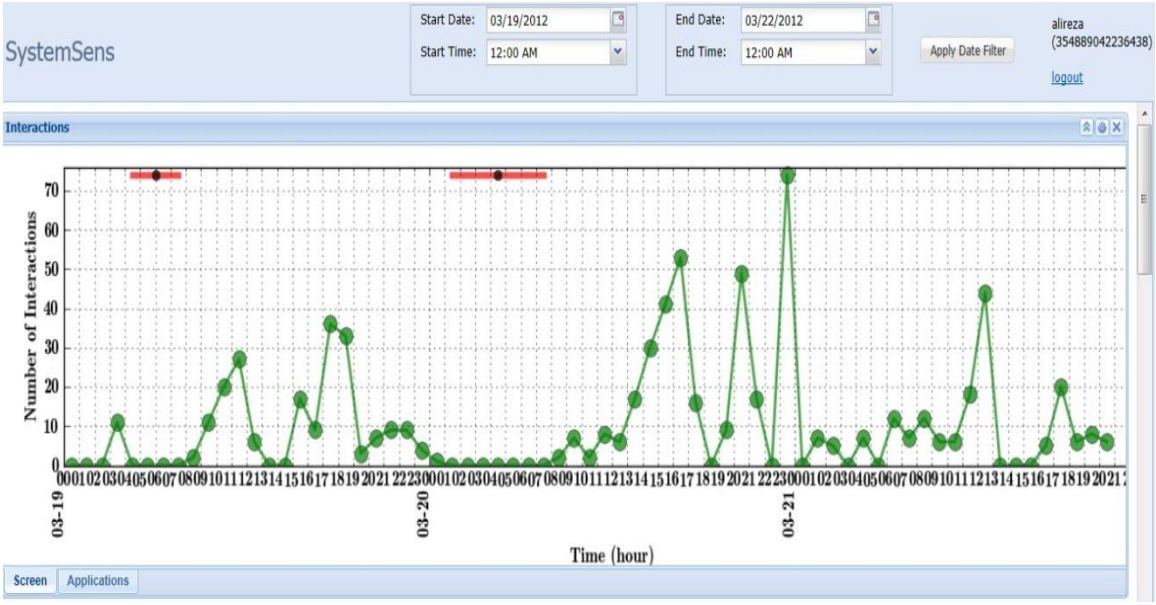

Figure 3.8: Screen Interaction

Figure 3.8 shows the user screen interaction during the measurement period. The graph shows the number of interactions. For some applications, like YouTube, the number of interactions is low because while watching a video on YouTube we do not interact with the screen. The case is different for Facebook or a game application because the user interacts frequently.

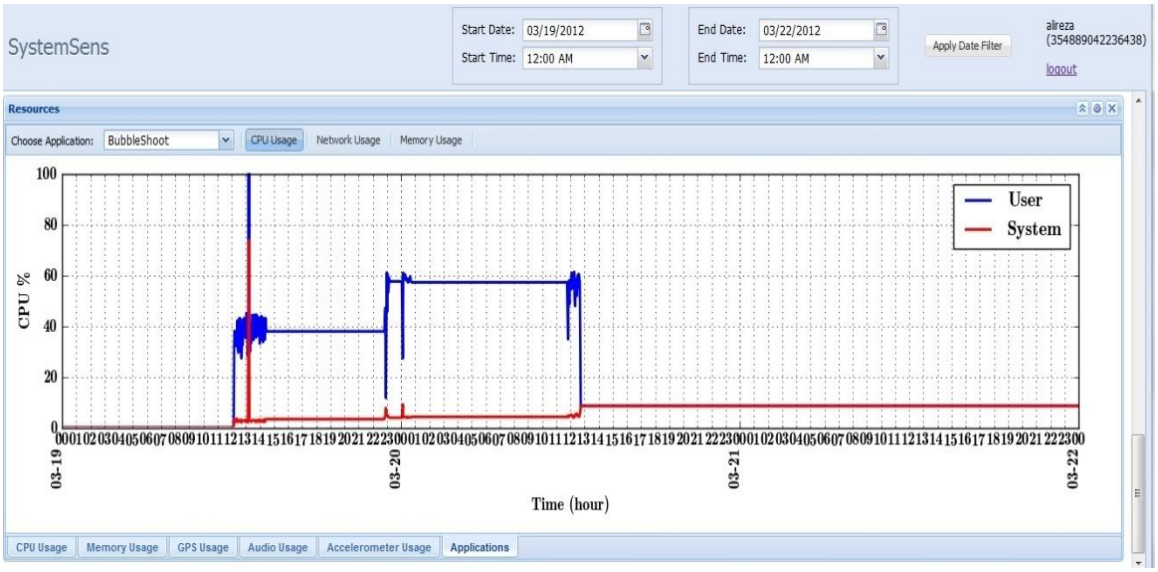

Figure 3.9: 'Bubble Shoot' game's CPU percentage

Figure above shows the user and system CPU percentage usage while using the "Bubble Shoot" game.

All above graphs were drawn based on our log file uploaded to CENS server. Unfortunately I could not get my data (log files) because the CENS server went memory full because it had many clients uploading to the server. The server master was not going to maintain it anymore. Since SystemSens generates a few MB of data each data and in one month total data becomes about 1GB so it is difficult for server master to do DB surgery and get our data out and send it to us. Then on emergency basis I had to look for another appropriate application to proceed with my thesis.

3.2 Measurements using Battery Monitor

We used Battery Monitor for overall system power consumption while running different applications. For some applications like Youtube and Facebook we adopted subtractive measurements. It is because during Youtube usage we cannot go to PowerTutor user interface (UI) and note the data because in that case Youtube stops. The same case is with Facebook.

The graphs below show the power consumption of DUT (in this case Google Nexus one smartphone) in normal situation i.e., not in flight mode.

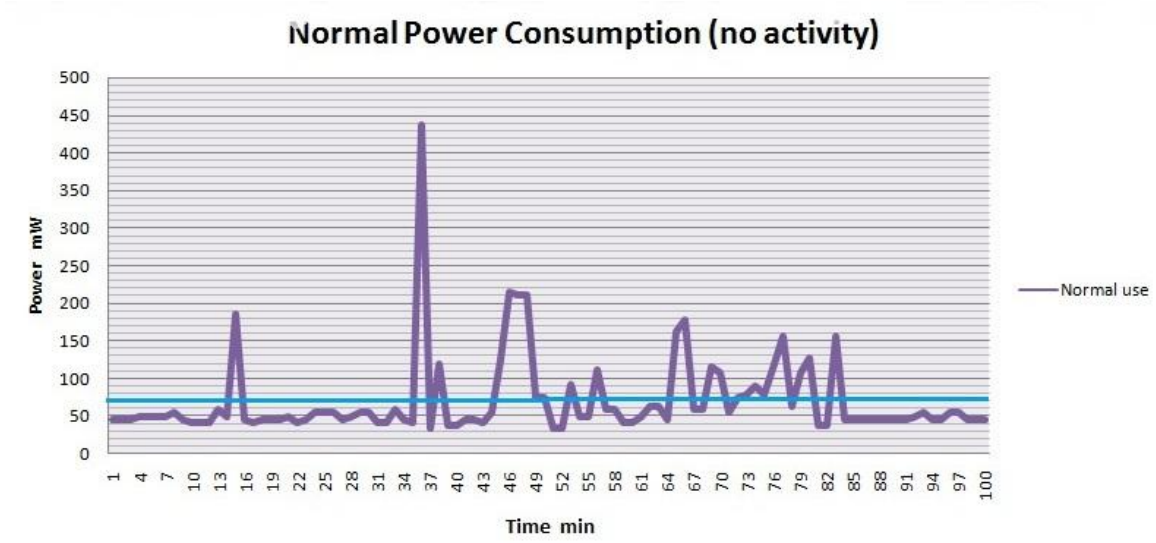

Figure 3.10: Power consumption of DUT with no activity

The DUT for measurement performed by Battery Monitor is Google Nexus One. The duration of the measurement was 100 minutes. During this period the smartphone was ON but with no application active. All radios were on including WiFi.

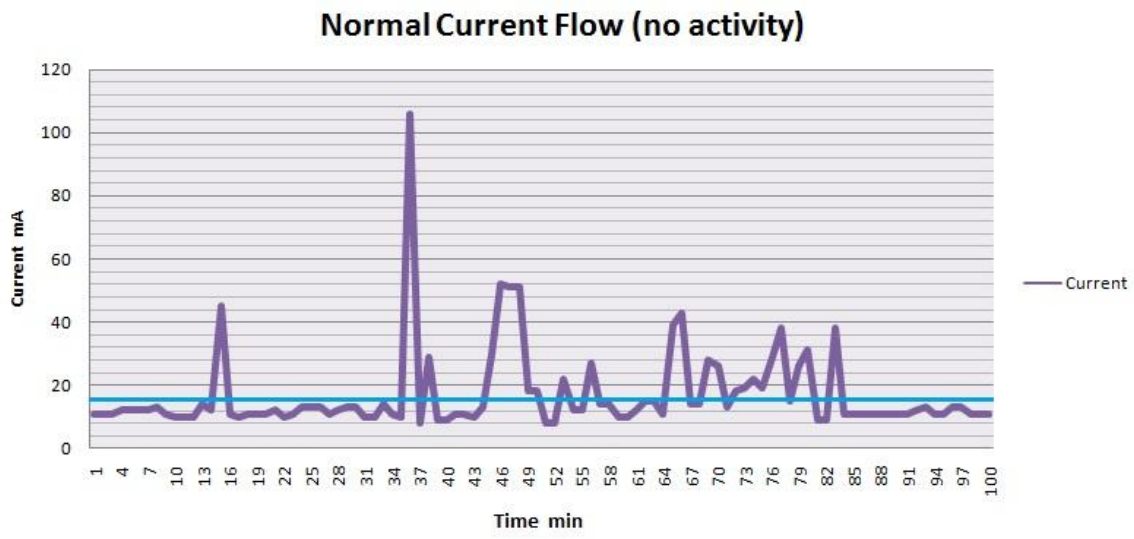

Figure 3.11: Current Flow during no activity

The graphs above show the normal power consumption and the normal current flow, when DUT is performing no activities. The current units are mA. On average the current drawn was 21.01 mA.

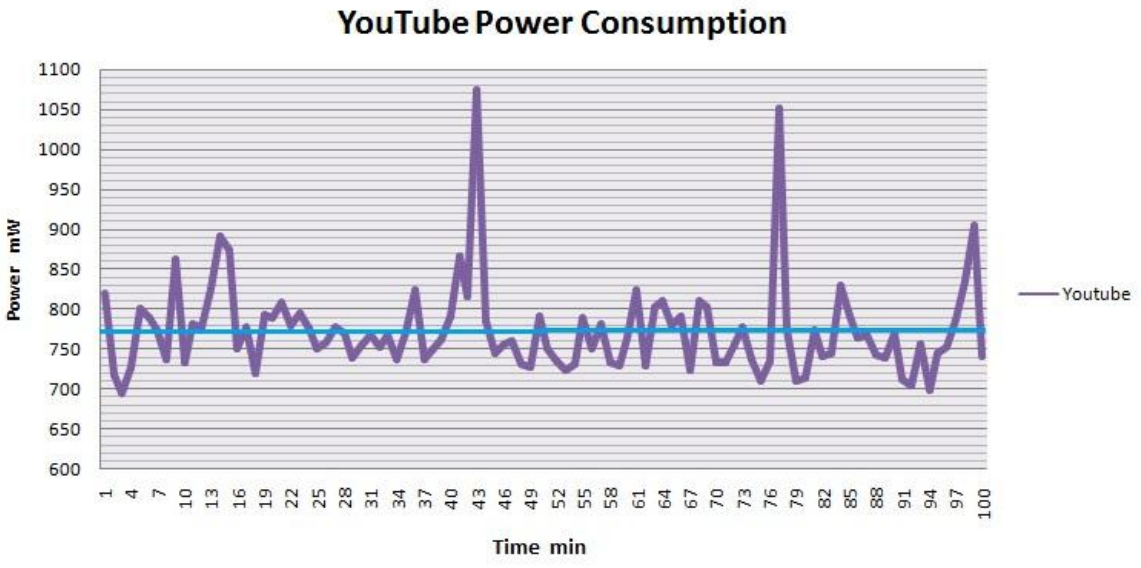

Figure 3.12: DUT power consumption when YouTube active

The graph above shows the power consumed by DUT when the YouTube application is active. We use subtractive measurement method to calculate the power consumed by YouTube. The average current drawn by DUT by YouTube is 686.96 mW.

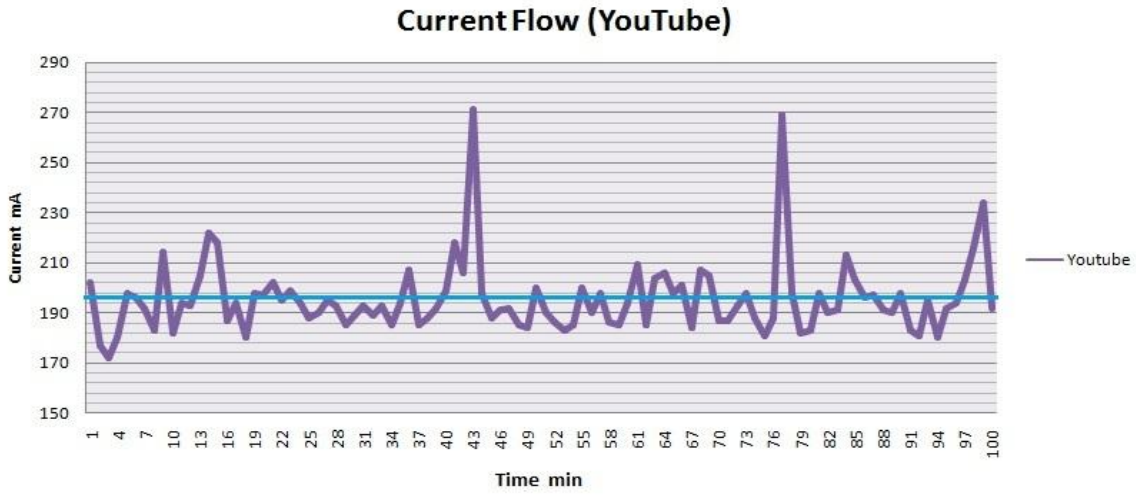

Figure 3.13: Current drawn by DUT when YouTube is active

The average current drawn by DUT when YouTube was ON is 195.59 mA. The duration for this measurement was also 100 minutes. The rise in power or currents flow is mainly due to the screen color changes during a video. Also since WiFi and cellular radios were ON therefore these are combined effects of these factors.

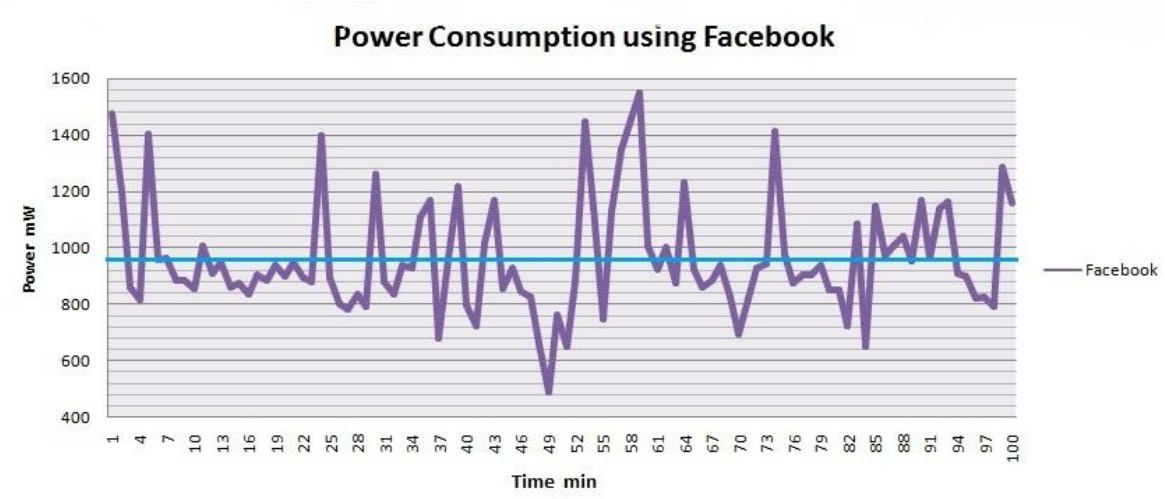

Figure 3.14: DUT power consumption when Facebook active

Figure 3.14 shows power consumption of Facebook. We use subtractive method to calculate the power consumption of Facebook as we did in the case of YouTube. The reference data was the power consumption measurements when DUT was ON but no applications were active.

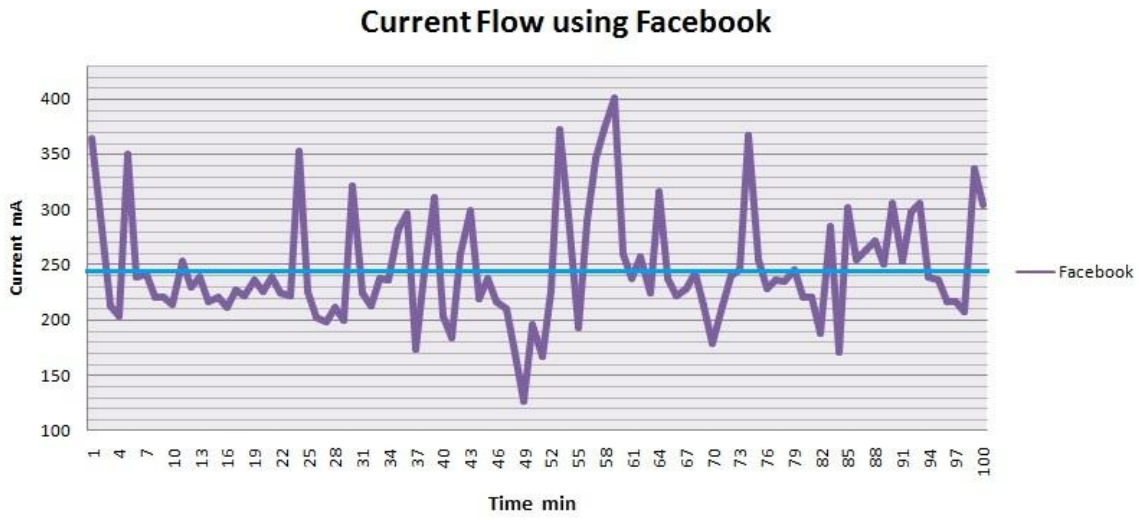

Figure 3.15: Current drawn by DUT when Facebook is active

The figure 3.15 shows the current draw by DUT when Facebook was active. The fluctuation in current flow is mainly due to screen interaction while using Facebook. The average current drawn is 246.57 mA.

3.3 Measurements using Power Tutor

Compare to Battery Monitor we found that Power Tutor gives more details. But a little problem was that the log file did not contain detail information. Also this application is developed for some certain models of Android smartphone. Earlier our DUT was Samsung Sidekick 4G but it was not the model that Power Tutor supported so we had to

purchase the smartphone which was compatible with Power Tutor and due to budget constraint our choice was Android Google Nexus One.

Using Power Tutor we took under consideration the three famous and frequently used applications: Skype, Viber and Tango.

Tango is an entertaining, easy to use, free video calling service that connects people around the world with friends and family from wherever they are [21]. Similarly Viber is an application for iPhone®, Android™, Windows Phone and Blackberry phones that lets you make free phone calls to anyone that also has the application installed. When you use Viber, calls and text messages to any other Viber user are free, and the sound quality is much better than a regular call. You can call or text any Viber user, anywhere in the world, for free. All Viber features are 100% FREE and do not require any additional "in application" purchase [22].

Below, the graphs show the power consumption of these applications.

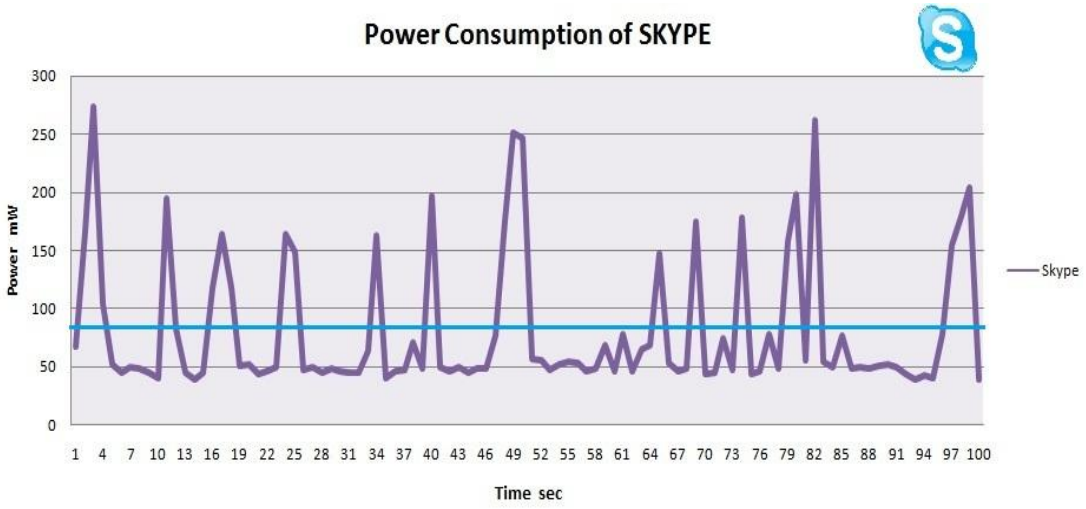

Figure 3.16: Power consumption of Skype

On average the power consumption of Skype is 80.38 mW. It is evident that WiFi consumes more power when transmitting than when receiving.

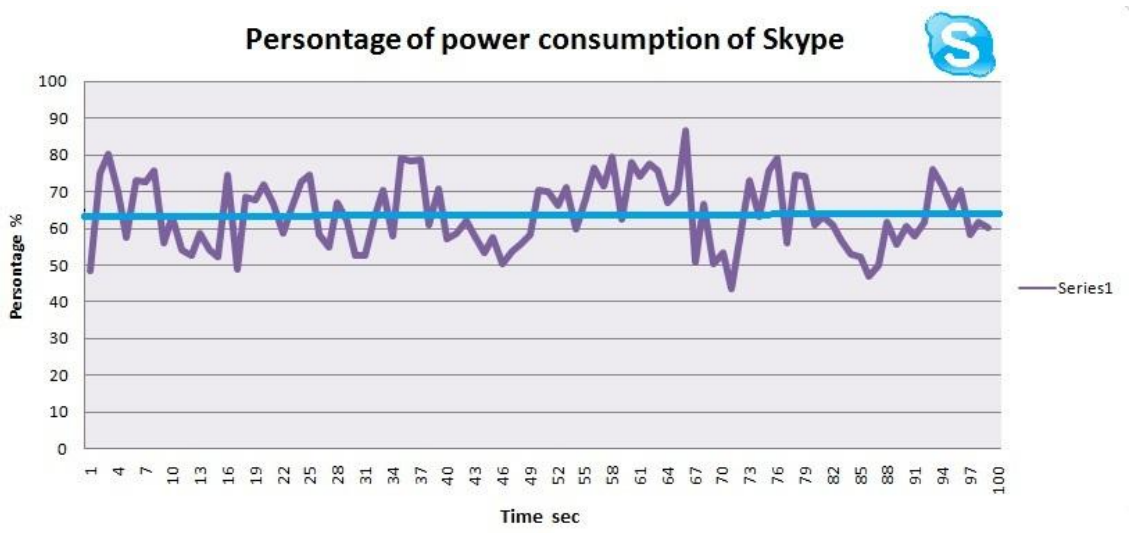

Figure 3.17: Percentage of Skype power consumption (Screen OFF)

Skype consumes about 63.64 percent of overall system power when the screen is turned off. But mostly we use Skype with screen turned on. With screen OFF Skype uses less power than when screen is ON.

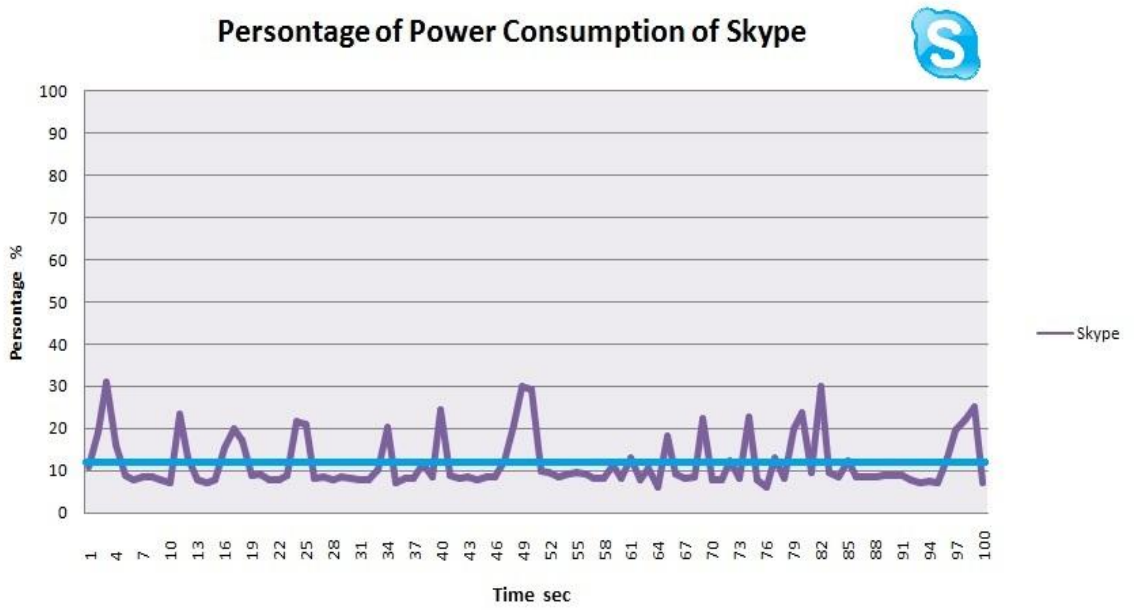

Figure 3.18: Percentage of Skype power consumption (Screen ON)

When the screen is turned on, then Skype consumes about 11.94 percentage of overall system power. With screen ON Skype consumes more power than when screen is OFF.

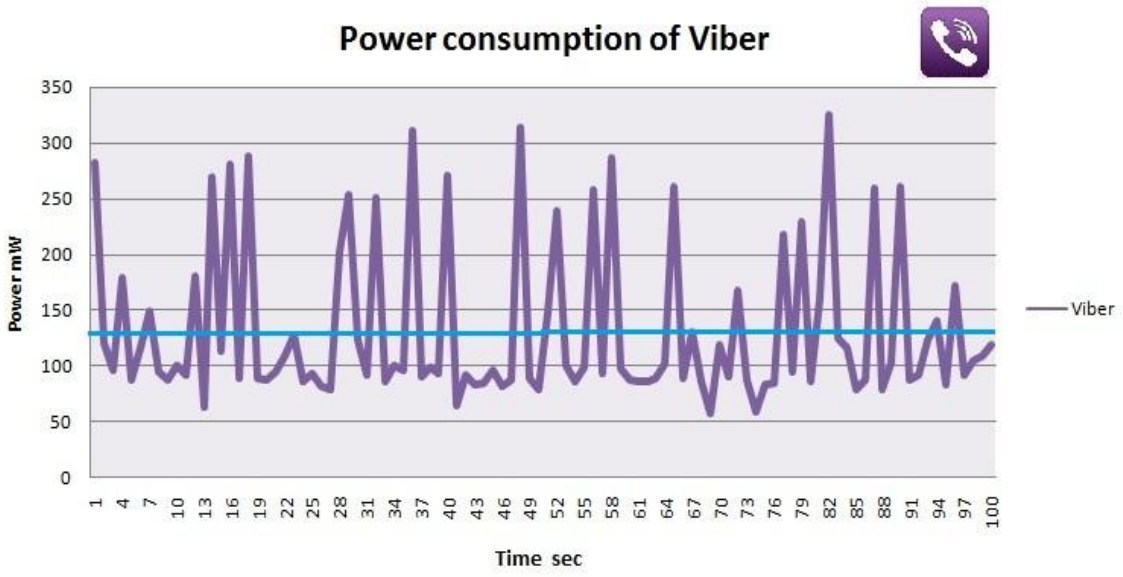

Figure 3.19: Power consumption of Viber

Viber consumes about 131.55 mW power on average. It is evident that WiFi consumes more power when transmitting than when receiving or when listening only.

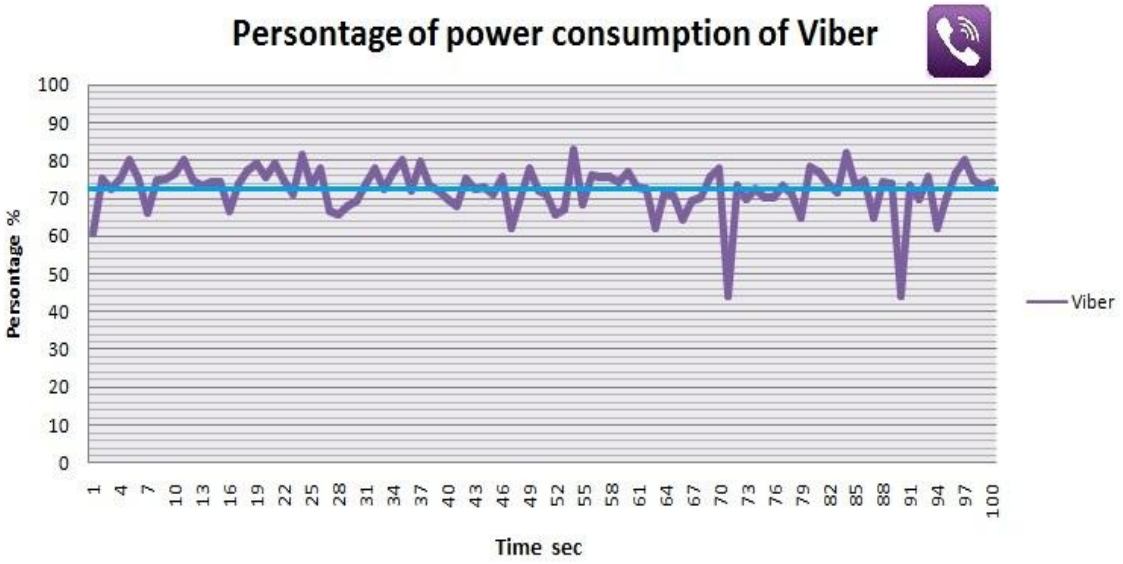

Figure 3.20: Percentage of Power consumption of Viber (Screen OFF)

Viber shares 72.37 percentage of overall system power when the screen is turned off. With screen OFF Viber consumes less power than when screen is ON. The duration of the measurement was 100 seconds.

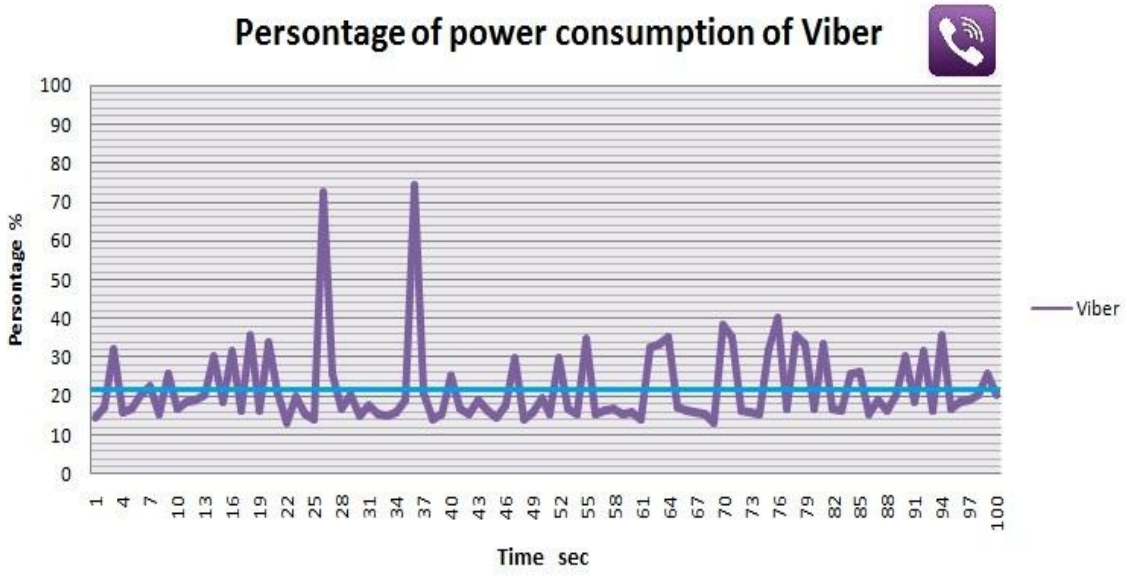

Figure 3.21: Percentage of Power consumption of Viber (Screen ON)

It is evident from figure 3.21 that Viber shares about 21.98 percent of overall system power when the screen is turned on. With screen ON Viber consumes more power than when screen is OFF. But the graph above shows the percentage to overall DUT power consumption.

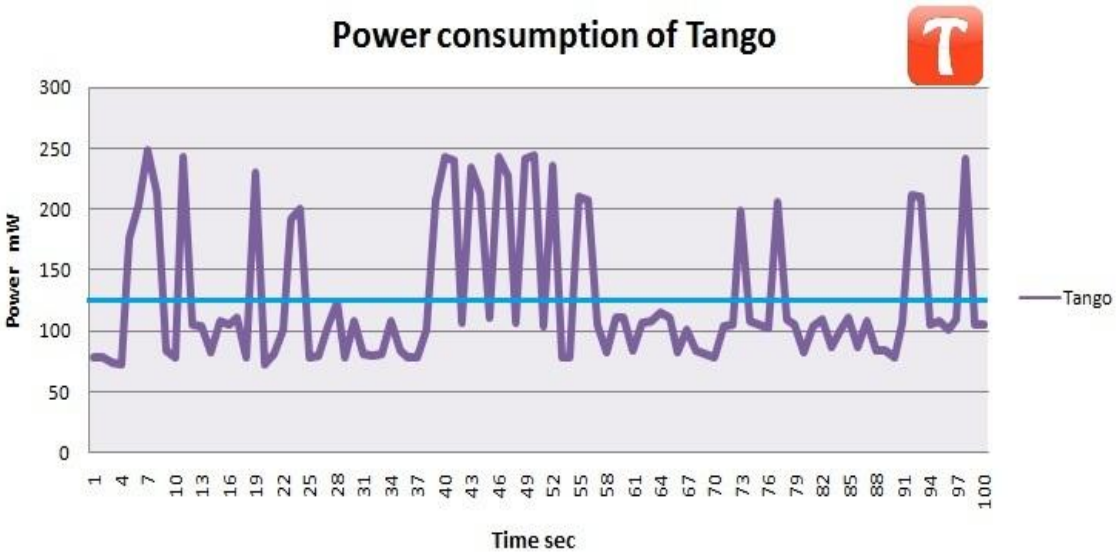

Figure 3.22: Power consumption of Tango

It is evident from figure 3.22 that Tango consumes about 126.98 mW powers on average. It is evident that WiFi consumes more power when transmitting than when receiving or when listening only.

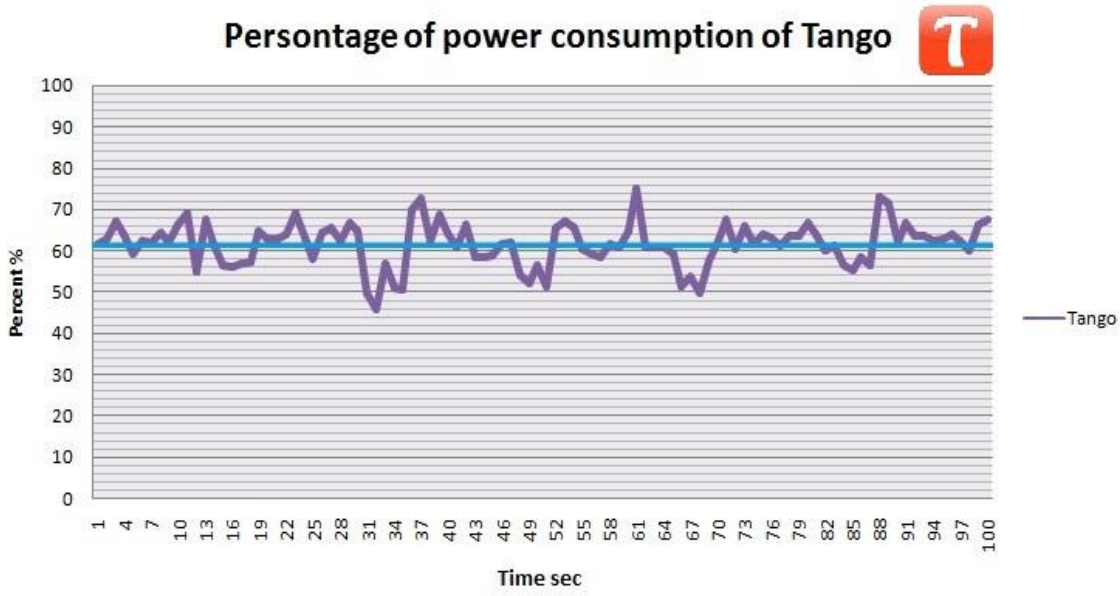

Figure 3.23: Percentage of power consumption of Tango (Screen OFF)

Tango shares 61.61 percentage of overall system power when the screen is turned off. With screen OFF Tango consumes less power than when screen is ON. The duration of the measurement was 100 seconds.

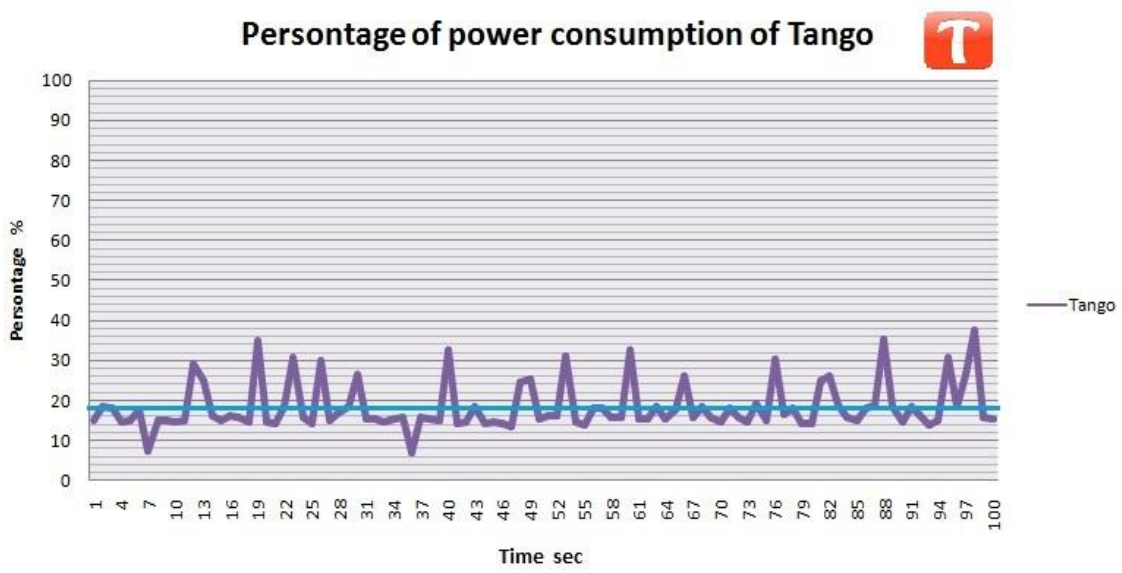

Figure 3.24: Percentage of power consumption of Tango (Screen ON)

It is evident from figure 3.21 that Tango shares about 18.30 percentage of overall system power when the screen in turned on. With screen ON Tango consumes more power than when screen is OFF. But the graph above shows the percentage to overall DUT power consumption.

Chapter 4

4. Measurement Analyses and Proposed Model

On bases of our power consumption measurements perform on DUT using SystemSens, Battery Monitor and Power Tutor; we developed a power model of DUT when using specific applications. The applications studies are Skype, Viber, Tango, YouTube and Facebook. We also have compared Skype, Viber and Tango from energy efficiency point of view.

As to be discussed in Chapter 5, unfortunately we could not get our log file from CENS server therefore we developed our model using the other two applications: Battery Monitor and Power Tutor.

4.1 Skype, Viber and Tango Measurements and Comparison

Table 2 describes the analysis of power consumption of Skype, Viber and Tango. The application used to measure the power consumption of these applications is Power Tutor.

	Skype	**Viber**	**Tango**
Average	81.38 mW	131.55 mW	126.98 mW
Variance	3438.93	4941.40	3215.41
Std. deviation	± 58.64	± 70.29	± 56.70

Table 4.1: Comparison of Skype, Viber and Tango

As it evident from the table that Skype is more energy efficient than Viber and Tango with an average power consumption of 81.38 mW. The standard deviation of Skype is much similar to that of Tango i.e., ± 58.64 and 56.70 respectively while the standard

deviation of Viber is ±70.29 which is much higher than Skype and Tango. The standard deviation is because of power consumption variation during transmitting and receiving data. During transmission power is consumed more than when receiving.

4.2 YouTube and Facebook Measurements

Table 3 describes the analysis of power consumption of YouTube and Facebook. The application used to measure the power consumption of these applications is Battery Monitor.

	YouTube	**Facebook**
Average	686.96 mW	875.97 mW
Variance	3356.51	39497.92
Std. deviation	± 57.93	± 198.74

Table 4.2: Analysis of YouTube and Facebook

The average power consumption of YouTube is 686.96 mW and the average power consumption of Facebook is 875.97 mW.

We cannot directly compare these two applications because these applications do not serve the same purpose. When a user is watching a video on YouTube then the user interaction with the screen is extremely low or sometimes even nil. This differs from Facebook because while using facebook the user continuously interacts with screen thus consuming more power on behave of CPU usage. This affect can evident from the average power consumption difference of these two applications and even more evident from the standard deviation. The standard deviation of Facebook is more than three times to that of YouTube standard deviation.

4.3 Proposed Model

We can express the results of section 4.1 and 4.2 in a scenario based energy model of the DUT using specific applications, which shows the energy for each applications usage scenario as a function of time:

Standard deviation is a statistical measure of spread or variability. The standard deviation is the Root Mean Square (RMS) deviation of the values from their arithmetic mean. The formula for finding standard deviation is as under, [23]

$$\sigma = \sqrt{[\sum (X_i - \mu)^2 / n]}$$

Where,

σ denotes standard deviation

μ denotes sample mean

n is the number of scores in sample

In this case

$$E_{Skype}(t) = (80.38\ mW \pm 58.64) * t$$

$$E_{Viber}(t) = (131.55\ mW \pm 70.29) * t$$

$$E_{Tango}(t) = (126.98\ mW \pm 56.70) * t$$

$$E_{YouTube}(t) = (686.96\ mW \pm 57.93) * t$$

$$E_{Facebook}(t) = (875.97\ mW \pm 198.74) * t$$

Summarizing and formulating the above, we can write

$$E^{i}_{t} = (\mu^{i} \pm \sigma^{i}) * t$$

Where, E is the total energy for the duration of time t.

And *i* denotes application used.

Chapter 5

5. Discussions

In this master thesis, the focus has been on the power consumption measurement and analysis of a few popular and commonly used applications in smartphones.

We divided the thesis in multiple activities and phases. In the first phase we concentrated in a thorough understanding of Android system and that how it differs from Linux despite that Android is based on Linux. We studied that what are main power consuming components in an Android smartphone.

In the second phase we performed measurements regarding power consumption of different application in Android smartphone. The DUT was Samsung Sidekick 4G. Initially we studies power consumption and performed specific measurement using SystemSens. As discussed in Chapter 3 the log file was uploaded to SystemSens server on regular basis. After using the application for almost two months, unfortunately we were unable to get our log data saved on the CENS server. It was due to CENS server memory full problem and the server master was unable to dig the database and sent us the log.

It was much unexpected situation. We proceeded in measuring power consumption of different application in DUT in laboratory.

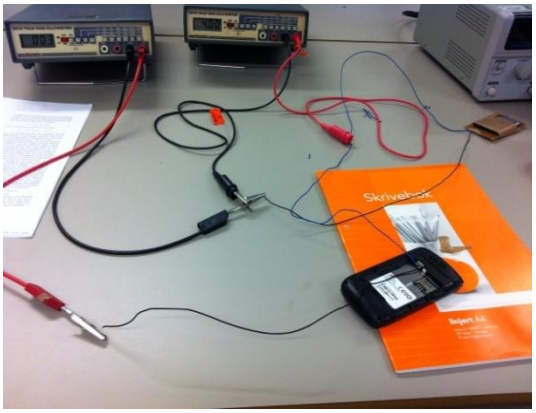

Figure 5.1: Power measurement performed in Lab1

Figure 5.2: power measurement performed in Lab 2

We were not satisfied with the measurements accuracy therefore we quit manual power measuring in lab.

Then we used Power Tutor and Battery Monitor for power consumption measurements. These two applications do not cover many aspects of power measurements as SystemSens did. But with the use to both of them we could perform measurements to some satisfactory extend.

Initially our DUT was Samsung Sidekick 4G but when we started using Power Tutor, we encountered a problem. Power Tutor was not compatible with the DUT. Then on eleventh hour we purchased a smartphone that was compatible with Power Tutor. We purchased Google Nexus One smartphone.

In the last phase, based on power consumption measurements, we analyzed and developed a power consumption model for some very popular and commonly used applications like Skype, Tango, Viber, YouTube and Facebook.

The measurement results are obtained after repeated tests and observations to minimize the error and increase accuracy.

The model basically does not cover the overall system but one some specific applications. The main reason is that there are no software tools and applications available that performs and covers all aspects of measurement or at least we were unable to find one despite thorough search. But it can help understand what applications among others are

energy-efficient who have the similar use. And also based on this model new application developers may have a reference to develop more energy-efficient applications and gain market success.

Chapter 6

6. Conclusions and Future Work

Since much of the research has been performed in the areas of network management and less studies done on how a smartphone consume energy. Therefore in this thesis we tried to model the energy consumption of some popular applications and to analyze and highlight how these applications use energy differently.

6.1 Contributions

Using three software tools we perform energy measurements and came up with a model for some commonly used applications. A similar research was carried out in [24] but in their model they have not taken the standard deviation into account.

The model may serve as a reference for smartphone application developers so that they may improve or develop new energy-efficient applications for smartphones. Also this may serve as a first step for new researchers and students and they may take steps ahead and make a model for smartphone and may possibly improve the model.

6.2 Future Work

Based on this study there are multiple interesting researches possible.

Firstly, as we found SystemSens to be very useful software tool as it measures energy consumed by main components and applications, numbers of packets transmitted and received, numbers of bytes transmitted and received, signal strength, number of

disconnections etc. there are two ways to save log file. Either to save log on SD card on smartphone or upload log to CENS server on regular bases. We saved data on CENS server and unfortunately could not get our log data due to memory full problem of CENS server. There is possibility to set up one's own server. We could not do that we were short of time. So one possible future step could be setting up one's own server and then uploading all data on server. It has multiple advantages:

1. One can obtain all log data and analyze the aspects of interest in a smartphone.
2. It is possible to study "usage pattern" but allowing many users uploading data to the server, as part of research.

The procedure and code, of how to set up the server, is given in Appendix A.

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.

References

[1] D. Z. Yazti, “Energy Efficient Date Management in Smartphone Networks” US National Science Foundation Workshop on Sustainable Energy Efficient Data Management, April 2-3, 2011, Arlington, Virginia, USA

[2] Androidology: Architecture overview http://developer.android.com/videos/index.html#v=QBGfUs9mQYY

[3] F. Maker, Y. Chan, “A Survey on Android vs. Linux”

[4] www.powertutor.org

[5] http://systemsens.cens.ucla.edu/service/viz/login/?next=/service/viz/#features

[6] [PM4] Android SDK: PowerManager http://developer.android.com/reference/android/os/PowerManager.html

[7] [PM3] Advanced Configuration and Power Interface http://en.wikipedia.org/wiki/Advanced_Configuration_and_Power_Interface

[8] Banerjee, N. Rahmati, A. Corner, M. D., Rollins, S., Zhong, “Users and Batteries: Interactions and Adaptive Energy Management in Mobile Systems”, Ubicomp 2007

[9] Rahmati, A., Qian, A., Zhong, “ Understanding Human-Battery Interactions on Mobile Phones” Pervasive and Mobile Computing 5, 5 (2009)

[10] [PM2] Android Power Management http://letsgoustc.spaces.live.com/Blog/cns!89AD27DFB5E249BA!526.entry

[11] Shye, A., Sholbrock, B., G, M. “Into the Wild: Studying Real User Activity Patterns to Guide Power Optimization for Mobile Architectures” In Micro (2009)

[12] Thiagarajan A., et. al., “VTrack: Accurate, Energy-aware Road Traffic Delay Estimation using Mobile Phones”, In *SenSys*, 2009

[13] H. Falaki, R.Mahajan, S. Kandula, D. Lymberopoulos, R. Govinda, D. Estrin, “ Diversity in Smarphone Usage” MobiSys’ 10, June 15-18, 2010, San Francisco, USA

[14] Z. Zhuang, K. Kim, J. Singh, “Improving Energy Efficiency of Location Sensing on Smartphones,” in proceedings of ACM MobiSys’ 10, pp. 315-330, June 2010

[15] J. Kang, S. Seo, J. Hong, “Usage Pattern Analysis of Smartphones”

[16] R. Friedman, A. Kogan, Y. Krivolapov, “On Power and Throughput Trafeoffs of WiFi and Bluetooth in Smartphones”

[17] M. Ra, J. Paek, A. Sharma, R. Govinda, M. Krieger, M. Neely, “Energy-Delay Tradeoffs in Smartphone Applications” MobiSys’10, June 15-18, 2010, San Francisco, USA

[18] H. Choi, J. Lee, “Analysis of Tradeoff Between Energy Consumption and Activation Delay in Power Management Mechanism” IEEE 2011, 1089-7798

[19] “Cisco Virtual Networking Index: Global Mobile Data Traffic Forecast Update, 2010-1015,” http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html

[20] H. Falaki, R. Mahajan, D. Estrin, “SystemSens: A Tool for Monitoring Usage in Smartphone Research Deployments” MobiArch’ 11, June 28, 2011, Bethesda, Maryland, USA

[21] http://www.tango.me/how-to-tango/

[22] http://viber.com/

[23] http://www.mathsisfun.com/data/standard-deviation-formulas.html

[24] A. Carroll, G. Heiser, "An Alanysis of Power Consumption in a Smartphone" USENIXATC'10 Proceedings of the 2010 USENIX conference on USENIX annual technical conference

[25] http://blog.hubspot.com/blog/tabid/6307/bid/30862/Mobile-App-Usage-Trumps-Web- Browsing-at-94-Minutes-a-Day-Data.aspx#ixzz1wIMJ5yw8

[26] https://play.google.com/store/apps/details?id=ccc71.bmw&hl=no

[27] S. Hu, Y. Yu, L. Xie, "Comparing Power Management Strategies of Android and TinyOS*", 978-1-4577-0856-5/11, 2011 IEEE

[28] A.Shye, B. Scholbrock, G. Memik, P. Dinda, "Characterizing and Modeling User Activity on Smartphones: Summary", SIGMETRICS' 10, June 14-18, 2010, New York, USA

[29] H. Falaki, D. Lymeropoulos, R. Mahajan, "A First Look at Traffic on Smartphones", IMC' 2010, November 1-3, 2010, Melbourne, Australia

[30] G. Majer, F. Schneider, and A. Feldmann, "A First Look at Mobile Hand-Held Device Traffic", in PAM, 2010.

[31] http://en.wikipedia.org/wiki/Mobile_phone_signal

[32]]M. A. Viredaz, L. S. Brakmo, and W. R. Hamburgen, "Energy Management on Hand-Held Devices," Queue, Vol 1, no. 7, pp. 44-52, October 2003.

[33] J. Sorber, N. Banerjee, M. D. Corner, and S. Rollins, "Turducken: Hierarchical Power Management for Mobile Devices," in proceedings of ACM MobiSys '10. Pp. 261-274, 2005

[34] Y. Won, B. Park, S. Hong, K. Jung, H. Ju, and J. Hong, "Measurements Analysis of Mobile Data Networks", PAM, 2007.

[35] E. Shih, P. Bahl, and M. J. Sinclair, “Wake on Wireless: An Event Driven Energy Saving Strategy for Battery Operated Devices,” in Proc, ACM MOBICOM, 2002, pp. 160-171

Appendix A

SystemSen's source code can be downloaded from: <https://github.com/falaki/SystemSens>

The default version uploads its data to SystemSens server. To have it dump the data on the SD card one needs to modify src/edu/ucla/cens/systemsens/util/Uploader.java. There is a method called tryUpload(). This method gets called every time the phone is plugged to the charger. All one needs to do is to modify this method to instead write the records in a text file on the SD card.

The method to dump the data on SD card is attached and normally if one replaces tryUpload() in the main source with the tryDump() method then it should dump data on SD card.

And following is the method to setup a server of one's own.

0. Install Apache, Python and Django on your Linux machine (Tested on Ubuntu). Just in case, these are the packages that CENS server master installed on server

python 2.6.5-0ubuntu1

apache2 2.2.14-5ubuntu8.6

python-django 1.1.1-2ubuntu1.3

libapache2-mod-python 3.3.1-8ubuntu2

libpython2.6 2.6.5-1ubuntu6

mysql-common 5.1.41-3ubuntu12.7

mysql-server 5.1.41-3ubuntu12.7

mysql-server-5.1 5.1.41-3ubuntu12.7

mysql-server-core-5.1 5.1.41-3ubuntu12.7

python-mysqldb 1.2.2-10build1

python-matplotlib 0.99.1.2-3ubuntu1

1. Download the SystemSens server source from:

<http://systemsens.cens.ucla.edu/~falaki/systemsens-server.tar.gz>

Expand it under /opt (so that you will have /opt/systemsens/service).

2. Install apache mod_python and add the following to your httpd.conf

< Location "/service/">
SetHandler python-program
PythonHandler django.core.handlers.modpython
SetEnv DJANGO_SETTINGS_MODULE service.settings
PythonInterpreter service
PythonOption django.root /service
PythonDebug On
PythonPath "['/opt/systemsens', '/opt/systemsens/service',
'/opt/systemsens/service/visualization'] + sys.path"
</Location>

Alias /service/media "/opt/systemsens/service/media"
<Location "/service/media">
SetHandler None
</Location>

3. Add a mysql database user named 'serviceuser' with the password 'servicepass' and create a database named 'service' and grant all access to 'serviceuser' (If you prefer other db config you can change

these in /opt/systemsens/service/settings)

4. Go to '/opt/systemsens/service and run:

$python manage.py syncdb

This will tell django to create all the tables that it needs.

5. Restart your apache server and go to the following address to see if it is working.

http://<yourdomain>/service/viz/login

6. The clients will upload their data without needing to authenticate. If you want to give each individual user access to view his/her data, then you need to make an account for each user and link it to the IMEI (the IMEI goes in the email field). But for you to see/fetch all users' data you do not need an account (you can access the MySQL DB directly)

To add a user account, do the following:

$cd /opt/systemsens/service

$python manage.py shell

> from django.contrib.auth.models import User

> u = User.objects.create_user('username', 'imei', 'password')

> u.save()

> exit()

7. Use this user/pass to login and test everything.

Remember to modify the client source code to upload to your new server (the default will upload its data to CENS server).