Research Article

A Tool for Energy Consumption Monitoring and Analysis of the Android Terminal

Deguang Li ¹, Zhanyou Cui ², Chenguang Bai,¹ Qiurui He,¹ and Xiaoting Yan¹

¹School of Information Technology, Luoyang Normal University, Luoyang 471934, China

²College of Mechanical and Electrical Engineering, Zhengzhou Institute of Industrial Technology, Zhengzhou 451150, China

Correspondence should be addressed to Zhanyou Cui; zhanyou_cui@126.com

Received 5 February 2021; Revised 15 June 2021; Accepted 4 July 2021; Published 14 July 2021

Academic Editor: Yang Li

Copyright © 2021 Deguang Li et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the rapid development of communication technology, the intelligent mobile terminal brings about great convenience to people's life with rich applications, while its power consumption has become a great concern to researchers and consumers. Power modeling is the basis to understand and analyze the power consumption characteristics of the terminal. In this paper, we analyze the Bluetooth and hidden power consumption of the android platform and fix the power model of open-source Android platform. Then, a power consumption monitoring tool is implemented based on the model; the tool is divided into three layers, which are original information monitor layer, power consumption calculation layer, and application layer. The original monitor layer gets the power consumption data and running time of the different components under different states, the calculation layer calculates the power consumption of each hardware and each application based on the power model of each component, and the application layer displays the real-time power consumption of the software and hardware. Finally, we test our tool in real environment by using Xiaomi 9 Pro and perform comparison with actual instrument measurement; the error between the monitored value and the measured value is less than 5%.

1. Introduction

With rapid development of communication technology, especially the popularization of 4G and the deployment of 5G, the possibility of mobile intelligent terminal is greatly expanded. The global mobile phone holdings were 6 billion by the end of 2020 according to statistics from IDG [1]; the rapid popularization of intelligent terminals greatly promotes the development of mobile applications. By the end of 2019, the numbers of downloads of App Store and Google Play apps have reached 30.6 billion and 84.3 billion, respectively [2]. Many mobile applications are integrated into various industries, which brings about great convenience to people's work and life. However, most intelligent mobile terminals currently are powered by energy-limited batteries. Although a lot of optimization has been carried out at the battery level, hardware components, and operating system levels to improve energy utilization efficiency [3, 4], these improvements still cannot prevent low energy utilization or

unreasonable consumption of battery. Thus, the energy consumption of mobile terminal is still a focus for many researchers [5–7].

Current mobile terminals are equipped with a variety of I/O components, such as GPU, AI processor, AMOLED flexible screen, gravity sensor, ambient light sensor, and other sensors. Rich hardware brings people a richer and faster experience, while the problem is the increasing power consumption of terminals [8, 9]. However, the battery technology of intelligent terminals has not made a breakthrough. In the past 10 years, the capacity of Li batteries with the same volume has only increased by 10%, and the battery life is about 8 hours for most terminals. As the bottleneck of intelligent terminal battery technology cannot be overcome in a short term, it is impossible to solve the problem of battery manufacturing with higher specific energy (the amount of electric energy released by the electrode material per unit mass involved in the electrode reaction) in a short term. Therefore, it is necessary to reduce the energy

consumption of intelligent terminal and prolong the service life of intelligent terminal.

The first step to analyze the power consumption of intelligent terminal and its software is establishing a power consumption model. At present, the widely used power consumption models are mainly divided into two types: one is based on hardware power measurement instrument and the other is based on software power consumption model. The first method mainly uses the power instrument to measure the power consumption of the intelligent terminal or uses the special hardware sensor embedded in the intelligent terminal to count power consumption. The second method is establishing a power consumption model through software, mainly by recording resource utilization of each component by the software. Although the power consumption measured by the first method is more accurate, the special measuring instrument is very expensive, the test process is more cumbersome, and the measurement cycle is long, which makes it impossible to automate the terminals of different platforms. The second method is establishing a power consumption model in forms of software, which has a wide range and is relatively simple to operate. Therefore, current power consumption modeling method based on software has been recognized by many researchers.

Android is an open-source system and is widely used with an 85.9% market share. In this paper, we select Android system as our research object. Android system has designed a special power management framework for power management, and the power consumption statistics include application power consumption statistics and hardware power consumption statistics. Application power consumption statistics usually displayed the ranking list of the total power consumption of each application, and hardware power consumption statistics display a ranking list that counts the total power consumption of main hardware, which is mainly implemented through the interface of `processAppUsage()` and `processMiscUsage()` in `BatteryStats` service of the system. The energy consumption statistics of each application are usually calculated by accumulating the power consumption of CPU, wake lock, radio, Wi-Fi, Bluetooth, sensors, camera, and flashlight. While the method of calculating Bluetooth power consumption in the source code is an empty method and some power consumption items cannot accurately confirm their sources, these questions are not giving a detailed explanation in the system power model.

Aiming at the above problems, we propose a specific calculation model of Bluetooth power consumption and hidden power consumption in the system and implement the power consumption model based on Android source code. According to the power consumption model, we implement the power consumption monitor tool, which is divided into three layers from top to bottom. The names of each layer are the original information monitor layer, the power calculation layer, and the application layer. The original monitor layer gets the power consumption data and running time of the different components under different states. The power calculation layer calculates the power consumption of each hardware and each application based

on the power model of each component, and the power data is abstracted as interfaces to the application layer for different using. The application layer displays the real-time power consumption of the software and hardware in the form of diagrams. The main contributions in this paper are as follows:

- (1) A specific calculation model of Bluetooth power consumption and hidden power consumption of the system is proposed
- (2) A tool for energy consumption monitor and analysis of Android terminal is implemented
- (3) The error between the measured value and the monitor value of the tool is less than 5% in real environment by using Xiaomi 9 Pro and Samsung terminals

2. Related Work

Farooq et al. [10] proposed a probabilistic technique for fine-grained energy estimation to compute the energy consumption of each method in Android applications; they used a probabilistic approach to map application power profile onto the corresponding execution traces to compute method-level energy by extracting the relevant energy consumption and execution traces. Dai et al. [11] designed a static visual detection and classification method based on application energy consumption; they first analyzed the bytecode images generated by the dex and xml files in the Android apk and trained them with deep learning methods; then the energy consumption level of two kinds of different mobile terminal software can be obtained, which can quickly estimate the energy consumption of mobile terminal when the application. However, the accuracy of their model is slightly lower. Amsel and Tomlinson [12] proposed a green tracker to estimate energy consumption of common user behaviors such as browsing Internet pages, creating text files, listening to music, and other behaviors by calculating their CPU utilization rate, so that developers and end users can have an intuitive understanding of energy consumption of commonly used application. In their research, they found that the energy utilization of Firefox is more effective than Safari when the same functions are completed. However, their research only made statistics on the CPU components and did not count the energy consumption of other components, so the results can only show the energy consumption of CPU utilization rate.

Li et al. [13] proposed energy consumption model at application level based on hardware components and designed a software consumption statistical tool named as *bugu* for intelligent terminal; the model calculated not only the CPU utilization of the application but also the power consumption of Wi-Fi, screen, Bluetooth, and radio components, and the accuracy of the model was greatly improved. Zhang et al. [14] used the knowledge of battery built-in sensor and battery discharge curve to train different components for different types of terminals, so as to obtain the battery power consumption of different components in different states. Through multiple linear regression of the

energy consumption data of multiple groups of programs, the energy consumption model of different intelligent terminals can be obtained. This method can automatically generate the energy consumption model based on the actual terminal and display the power consumption information of different software and components in real time. Duan et al. [15] noticed that the energy consumption of each software is proportional to its running time, and the model is relatively simple to implement and consumes less system resources at run time. However, the accuracy of the model is poorer than PowerTutor and has some limitations.

In order to have a clear understanding of the energy consumption of smart terminals, Carroll and Heiser [16] conducted a detailed power consumption analysis on two different smart phones. They not only measured the overall power consumption of the system but also measured the power consumption of the main hardware components in the device. Through a number of test programs, the system energy consumption was decomposed to understand the various components by imitating the use scenarios of real user's energy consumption characteristics. Li et al. [17] proposed a source code level energy consumption calculation method for Android applications. First, they record the energy consumption measurement information based on hardware of the application and the relevant information about the effective path of the application. Then the energy consumption information and path execution information of the collected program are regressed and mapped to each source program to obtain the energy consumption of each source code. Yoon et al. [18] presented an Android-based energy metering system, which monitors application's hardware usage at the kernel level and accurately estimates energy consumption. The system was implemented as a kernel module and uses an event-driven monitoring method that generates low overhead and provides high accuracy. While the method of counting the Bluetooth power consumption in the source code is empty, and some power consumption cannot accurately confirm its source, these questions are not giving a detailed explanation in the system power model; thus we try to find answers to these questions.

3. System Design of Our Model

3.1. Overview of the Design. The power statistics architecture of our model is divided into the original information statistics layer, power calculation layer, and application layer from bottom to top, as shown in Figure 1.

The original information layer mainly monitors the power information of each hardware and the running time information of each hardware in different states. The power consumption calculation layer includes the power calculation model and statistical model. The power calculation layer calculates the power consumption information of each hardware according to the power information and service time of each hardware, and the statistical model accumulates the power consumption information of each hardware. The application layer displays the real-time power consumption of the software and hardware in the form of diagrams, which relies on the power calculator and the battery sipper to

provide statistical power data, while the calculation layer of the middle layer relies on the power profile and battery stats in the original information statistics layer.

The original information monitor layer is the summary of the original statistical information needed for power consumption statistics, which mainly records the power profile and battery stats. The power consumption of intelligent terminal processor in each frequency is obtained from powerprofile file, and the power of wireless network, Bluetooth, camera, flashlight, GPS, audio, and other related hardware modules in different states is obtained. The working time of different processes in different states, the number of times the terminal is awakened by the process, and the network usage information of the application are counted in batterystats. The current information of intelligent terminal hardware in different states obtained from powerprofile is recorded in Android power_profile.xml. The current recorded data is the data recorded by the mobile phone manufacturer by testing each part of the mobile phone separately. The power value of each component under different states of the intelligent terminal recorded by the intelligent terminal manufacturer will directly affect the accuracy of the power consumption monitoring of the intelligent terminal. The more accurate the value tested by the manufacturer, the more accurate the power consumption data calculated by using the data. Batterystats records the working time of each process of the application and also includes the statistics of wake-up times of each process to the kernel in sleep state.

The power calculation layer mainly includes the calculation model and statistical model. The calculation model calculates the power consumption of each component according to the information of the underlying power and the use time of the terminal hardware by the process. The statistical model calculates the final power consumption of each application by accumulating the power consumption of each component of the terminal. At the same time, this paper also discusses power consumption of the system which cannot accurately confirm the source and gives corresponding calculation model in the final power consumption calculation. The specific power consumption model of each component is shown in Section 3.2. The application layer displays real-time power consumption of the software and hardware in the form of diagrams, and power usage is used as the calling interface of the application layer, which provides the methods of obtaining statistical time, refreshing statistical data, and classifying statistical data.

3.2. Power Model of Each Component. Power consumption statistical model plays a very important role in terminal power consumption statistics. The power consumption model of each component is directly related to the error between the statistical results and the final results. According to the characteristics and principles of previous works and referring to many other power consumption models, we present power consumption model of the Android application and the power consumption model of each component of the terminal.

FIGURE 1: Overview of the system design.

3.2.1. Power Consumption of the Application. Application power consumption is the total power consumption of the top-level application. Since the parameter *uid* of the application is the only identification of the application at run time, we use *uid* as the basic unit to count the power consumption of all processes of the application. The power consumption calculation model of the application is shown as formula (1), AppSumPower is the total power

consumption of the application, *n* is the total number of processes contained in the application, and the power consumption of each application includes power consumption of the processor, wake-up lock, wireless network, camera, flashlight, mobile network, Bluetooth, and global positioning system. The meanings of all parameters in the model are given in Table 1.

$$\begin{aligned}
 \text{AppSumPower} &= \sum_{i=0}^n \text{ProcessSumPower}, \quad i \in \text{uid}, \\
 \text{ProcessSumPower} &= \text{CpuSumPower} + \text{WakeLockSumPower} + \text{WiFiSumPower} \\
 &\quad + \text{MobileRadioSumPower} + \text{CameraSumPower} + \text{FlashlightSumPower} \\
 &\quad + \text{BluetoothSumPower} + \text{GpsSumPower}.
 \end{aligned} \tag{1}$$

3.2.2. Power Consumption of the Processor. The processor power consumption calculation only needs to count the processor working time at different frequencies. According to the processor working power at different frequencies recorded in the powerprofile file, the working time is multiplied by the corresponding power consumption, and

then the power consumption is accumulated to obtain the final total power consumption of the processor. The processor power calculation model is shown in formula (2), *n* is the number of frequency points supported by processor, and the meanings of all parameters in the formula are given in Table 2.

$$\begin{aligned}
 \text{CpuSumPower} &= \sum_{i=1}^n \text{ratio}_i * \text{cpuTotalTime} * \text{cpuPower}_i, \\
 \text{ratio}_i &= \frac{\text{cpuSpeedTime}_i}{\text{cpuTotalTime}}, \quad (i = 1, 2, \dots, N).
 \end{aligned} \tag{2}$$

3.2.3. Power Consumption of the Wake Lock. Wake lock is a set of mechanisms provided by Android framework layer. Application can control Android device state through this

mechanism. Device state mainly refers to the opening and closing of the screen and the processor keeping running. Generally, as long as an application holds the lock, the

TABLE 1: Definition of each parameter in formula (1).

Name	Definition
<i>AppSumPower</i>	Total power consumption of the application
<i>ProcessSumPower</i>	Total power consumption of each process
<i>CpuSumPower</i>	Total power consumption of the processor
<i>WakeLockSumPower</i>	Total power consumption of the wake lock
<i>WiFiSumPower</i>	Total power consumption of the Wi-Fi network
<i>BluetoothSumPower</i>	Total power consumption of the Bluetooth
<i>RadioSumPower</i>	Total power consumption of the mobile network
<i>CameraSumPower</i>	Total power consumption of the camera
<i>FlashlightSumPower</i>	Total power consumption of the flashlight
<i>GpsSumPower</i>	Total power consumption of the positioning system

TABLE 2: Definition of each parameter in formula (2).

Name	Definition
<i>CpuSumPower</i>	Total power consumption of the processor
<i>ratio_i</i>	Specific frequencies supported by a processor
<i>cpuTotalTime</i>	Total working time of the processor
<i>cpuPower_i</i>	Operating power at a specific frequency of the processor
<i>cpuSpeedTime_i</i>	Working time at a specific frequency of the processor

processor cannot enter the sleep state and is always in working state. To calculate the power consumption of wake-up lock, we need to count the working time of wake-up lock and multiply it by its working power to get the final total power consumption of wake-up lock; the power calculation

model is shown in formula (3), *WakeLockSumPower* is the total power of wake lock, *wakeLockTime* is the working time of wake lock, and *wakeLockPower* is the working power of wake lock.

$$\text{WakeLockSumPower} = \text{wakeLockTime} * \text{wakeLockPower}. \quad (3)$$

3.2.4. Power Consumption of the Wi-Fi. There are usually two ways to calculate the power consumption of wireless network. The first method is to calculate the power consumption based on the working time in three different working states, namely, standby mode, data transmission mode, and data receiving mode. The second calculation model estimates the data transmission power consumption of wireless network according to the number of sending and receiving packets and the power consumption of single packet. In addition, the power consumption of wireless

network standby and network scanning is used to calculate the total power consumption of wireless network. Since the Wi-Fi activity attribute needs to be queried when the first method is used for calculation, the default value of this attribute is false. In this paper, we use the second calculation model to calculate the power consumption of Wi-Fi, the Wi-Fi power calculation model is shown in formula (4), and the meanings of all parameters in the formula are given in Table 3.

$$\begin{aligned}
\text{WiFiSumPower} &= \text{wifiPacketPower} + \text{wifiLockPower} + \text{wifiScanPower} \\
&\quad + \text{wifiBatchScanPower}, \\
\text{wifiPacketPower} &= (\text{wifiRxPackets} + \text{wifiTxPackets}) * \text{wifiPowerPerPacket}, \\
\text{wifiLockPower} &= \text{wifiRunningTime} * \text{wifiOnPower}, \\
\text{wifiScanPower} &= \text{wifiScanTime} * \text{wifiScanPower}, \\
\text{wifiBatchScanPower} &= \sum_{i=0}^4 \text{batchScanTime}_i * \text{batchScanPower}_i, \quad i \in \{0, 1, 2, 3, 4\}.
\end{aligned} \quad (4)$$

3.2.5. Power Consumption of the Bluetooth. Bluetooth power consumption statistics need to count the power

consumption of sending data, receiving data, and standby time of Bluetooth to get the final total power consumption;

TABLE 3: Definition of each parameter in formula (4).

Name	Definition
<i>WiFiSumPower</i>	Total power consumption of wireless network
<i>wifiPacketPower</i>	Power consumption of data transmission in wireless network
<i>wifiLockPower</i>	Power consumption of wireless network in standby state
<i>wifiScanPower</i>	Power consumption of wireless network in scanning state
<i>wifiBatchScanPower</i>	Power consumption of batch scanning in wireless network
<i>wifiRxPackets</i>	Number of packets received by wireless network
<i>wifiTxPackets</i>	Number of packets sent by wireless network
<i>wifiPowerPerPacket</i>	Power of single packet in wireless network
<i>wifiRunningTime</i>	Standby time of the wireless network
<i>wifiOnPower</i>	The power of a wireless network in standby state
<i>wifiScanTime</i>	Scanning time of wireless network
<i>wifiScanPower</i>	The power of the wireless network in scanning state
<i>batchScanTime</i>	Batch scanning time of wireless network
<i>batchScanPower</i>	The power of a wireless network batch scanning

detailed calculation model of power consumption of the Bluetooth is shown in formula (5), and the meanings of all parameters are given in Table 4.

$$\begin{aligned} \text{BluetoothSumPower} = & \text{BluetoothIdleTime} * \text{BluetoothIdlePower} \\ & + \text{BluetoothRxTime} * \text{BluetoothRxPower} + \text{BluetoothTxTime} \\ & * \text{BluetoothTxPower}. \end{aligned} \quad (5)$$

3.2.6. Power Consumption of the Mobile Radio. The power consumption of mobile radio is divided into two cases. In the first case, when tracking signal is active, the power consumption is obtained by multiplying the activity time and the power of mobile network. In the second case, when there is no tracking signal, the power consumption of mobile

network is obtained by multiplying the number of packets and the power consumption of a single packet; detailed calculation model of power consumption of the mobile radio network is shown in formula (6), and the meanings of all parameters are given in Table 5.

$$\begin{aligned} \text{MobileRadioSumPower} = & \text{activeTime} * \text{radioOnPower} + (\text{rxPackets} + \text{txPackets}) * \\ & \text{mobilePowerPerPacket}. \end{aligned} \quad (6)$$

3.2.7. Power Consumption of the Camera. The power consumption of camera needs to count the camera's working time and its working power, and the two are multiplied to get the final total power consumption of the camera; detailed calculation model of power consumption of camera is shown

in formula (7): *CameraSumPower* is the total power consumption of camera, *cameraOnTime* is working time of camera, and *cameraOnPower* is the power of camera in working state.

$$\text{CameraSumPower} = \text{cameraOnTime} * \text{cameraOnPower}. \quad (7)$$

3.2.8. Power Consumption of the Flashlight. The power consumption of flashlight needs to count the flashlight's working time and its working power, and the two are multiplied to get the final total power consumption of the flashlight; detailed calculation model of power

consumption of flashlight is shown in formula (8): *FlashlightSumPower* is the total power consumption of flashlight, *flashlightOnTime* is working time of flashlight, and *flashlightOnPower* is the power of flashlight in working state.

TABLE 4: Definition of each parameter in formula (5).

Name	Definition
<i>BluetoothSumPower</i>	Total power consumption of Bluetooth
<i>BluetoothIdleTime</i>	Standby time of the Bluetooth
<i>BluetoothIdlePower</i>	The power of a Bluetooth in standby state
<i>BluetoothRxTime</i>	Working time of Bluetooth receiving data
<i>BluetoothRxPower</i>	The power of Bluetooth receiving data
<i>BluetoothTxTime</i>	Working time of Bluetooth transmitting data
<i>BluetoothTxPower</i>	The power of Bluetooth transmitting data

TABLE 5: Definition of each parameter in formula (6).

Name	Definition
<i>MobileRadioSumPower</i>	Power consumption of mobile radio network
<i>activeTime</i>	Active time of mobile radio
<i>radioOnPower</i>	Power consumption of mobile radio when tracking
<i>rxPackets</i>	Number of packets received by mobile radio
<i>txPackets</i>	Number of packets sent by mobile radio
<i>mobilePowerPerPacket</i>	The power consumption of a single packet of mobile radio

$$\text{FlashlightSumPower} = \text{flashlightOnTime} * \text{flashlightOnPower}. \quad (8)$$

3.2.9. *Power Consumption of the Screen.* The power consumption of screen can be divided into two situations according to the type of terminal screen. The first is power consumption of LCD screen and the second is power consumption of OLED screen. LCD screen is composed by backlight panel and LCD screen; thus, the power consumption of LCD screen is the sum of backlight power consumption and LCD power consumption, so the power of the screen in full-brightness state, dark state, dim state, medium-brightness state, and light state needed to be obtained; then the total power consumption of the screen is

determined by the power consumption in the five states. The power consumption of OLED screen has no backlight, only the power consumption of OLED panel; thus the power consumption of OLED screen needs to count the working time and power under different brightness of OLED screen, and the multiplication of the two is the total power consumption of the screen. Detailed power consumption model of the LCD screen is shown in formula (9) and formula (10) demonstrates power consumption model of OLED screen; the meanings of all parameters are given in Table 6.

$$\text{ScreenSumPower} = \text{screenOnOnTime} * \text{screenOnPower} + \text{backLightPower},$$

$$\text{backLightPower} = \sum_{i=0}^4 \text{backLightTime}_i * \text{backLightPower}_i, \quad i \in \{0, 1, 2, 3, 4\}, \quad (9)$$

$$\text{ScreenSumPower} = \sum_{i=0}^4 \text{screenOnOnTime} * \text{screenOnPower}, \quad i \in \{0, 1, 2, 3, 4\}. \quad (10)$$

3.2.10. *Hidden Power Consumption of the Application.* In the process of power consumption statistics, for the power item whose power source cannot be accurately confirmed, it will not be included in the total power consumption of application, such as screen power in split screen mode of Android terminal; more than one application occupies the screen at the same time and also generates screen power consumption; at this time, the power consumption of each application screen cannot be accurately calculated. Therefore, this kind of power consumption is classified as hidden power consumption. There are usually three items marked as hidden power consumption, which are power consumption of

mobile screen, power consumption of mobile radio, and power consumption of mobile phone in standby state. If we want to divide this power consumption into different applications, we could only divide the total hidden power consumption to each application in proportion according to the ratio of the total power consumption recorded by application and the total remaining power consumption. The total hidden power consumption of the system is the power consumption of all unidentified sources from the time when the system is started until the current time. In order to ensure the universality of the system power consumption model and the subsequent system comparative analysis, the

TABLE 6: Definition of each parameter in formulas (9) and (10).

Name	Definition
<i>ScreenSumPower</i>	Total power consumption of the screen
<i>screenOnTime</i>	Working time of the screen
<i>screenOnPower</i>	The power of the screen in working state
<i>backLightPower</i>	The power of the screen backlight
<i>backLightTime</i>	Working time array of screen backlight (dark, dim, medium, light, and bright)
<i>backLightPower_i</i>	Power array of screen backlight (dark, dim, medium, light, and bright)
<i>screenOnTime</i>	Time array of screen in working state (dark, dim, medium, light, and bright)
<i>screenOnPower</i>	Power array of screen in working state (dark, dim, medium, light, and bright)

hidden power consumption is separated from the total application power consumption by referring to the power consumption statistical models of other mobile phone manufacturers, and the hidden power consumption is not included in the total application power consumption temporarily. Detailed calculation model of hidden power

consumption of application is shown in formula (11): hiddenPowerMah is total hidden power consumption of the system, totalPowerMah is total power consumption of the application, and totalRemainingPower is total remaining power consumption of the system.

$$\text{proportionalSmearMah} = \text{hiddenPowerMah} * \frac{\text{totalPowerMah}}{\text{totalRemainingPower}} \quad (11)$$

4. System Implementation

As the realization of power consumption tool in this paper requires system root authority, most Android terminal manufacturers have blocked the root permission of the terminal. It is not recommended that Android users root their own devices to make the devices run in an unsafe state. Therefore, the ultimate design goal of this tool is to integrate into ROM as a system application. Our power consumption tool requires the battery of Android intelligent terminal_ stats permission and dump permission, allowing applications to obtain the underlying information of terminal.

In the specific implementation, based on the Android system source code, the Bluetooth and hidden power consumption of the system which are not calculated in original source code are counted in our tool. It can be seen from system design in Section 3.1 that the power consumption model calculation layer is the core part of the power model in this paper, and the power consumption calculation model of each component in the power calculation model layer completes the calculation of the power consumption information of processes of all the applications. All the power consumption calculation functions are shown in Table 7 and different calculation functions correspond to different component power consumptions. The batterystats class records the running time of Android intelligent terminal in different states, including the statistics of wake-up lock and system wake-up times under locked screen, call duration, standby state, no data connection state, different signal states, different screen states, and other information. According to the information of the original information layer, the power computing layer can calculate the power of hardware based on the information and the time information.

The power calculator calculates the power consumption according to the information obtained from the bottom layer and stores the statistical information in batterysipper. Batterysipper stores the corresponding power consumption according to the *uid* of the application; batterystats also provides access to battery usage statistics, including information on wake locks, processes, packages, and services. The class of BatteryEntry stores the package name and the icon corresponding to the application, which wraps the power data. The system source code implementation folder mainly contains the data layer of alarm, app usage, network usage, and other statistical information. The internal package contains the batterysipper class, which is the underlying model layer of application power consumption, kernelutil is a class that identifies the state of the system, the location package contains the class for obtaining the system location, the nameutils package contains the class for obtaining the name and icon of the application, the networkutils package contains the class of network statistics, the privateapiproxies package contains the original data model of various statistical data, the shellutils package contains classes to execute shell commands, the utils package mainly contains the classes for common use, and the wifi package contains the class for obtaining screen lock information.

5. Results and Discussion

In order to verify our power consumption monitor tool, we use different types of Xiaomi and Samsung to test the tool and perform comparison with the power consumption data of Xiaomi original model and actual measurement. We evaluate our tool by using KA3005P DC power supply to provide stable and controllable voltage for the computer, and then we leverage HOIKI 3334 power measuring

TABLE 7: Power consumption calculation function of different hardware.

Class name	Definition
<i>CpuPowerCalculator</i>	CPU power calculation function
<i>BluetoothPowerCalculator</i>	Bluetooth power consumption calculation function
<i>WakelockPowerCalculator</i>	Wake-up lock power calculation function
<i>SensorPowerCalculator</i>	Sensor power consumption calculation function
<i>MobileRadioPowerCalculator</i>	RF signal power consumption calculation function
<i>CameraPowerCalculator</i>	Calculation function of camera power consumption
<i>WifiPowerCalculator</i>	Wi-Fi power consumption calculation function
<i>FlashlightPowerCalculator</i>	Flash power consumption calculation function
<i>HiddenpowerCalculator</i>	System hidden power calculation function

TABLE 8: Configuration of the mobile terminal and power meter.

Mobile terminal	Power meter
Mi 9 Pro 5G (root) ROM 256G RAM 8G	HIOKI 3334
Android 10 kernel version 4.14.117-perf+	Sampling frequency 74.4 kHz
CPU Qualcomm Snapdragon 855 plus	Measurement accuracy $\pm 0.5\%$ rdg
Android security patch 2020-03-01	Measurement range 1.5000 W–9.000 kW
Baseband version 1.5.2.c8-00039-0320_0238_07b83a7	

instrument to measure instantaneous power consumption and cumulative power consumption of the terminal.

As shown in Table 8, the configuration of Xiaomi terminal (Mi 9 Pro 5G) used in the test is listed. Due to space limitation, configuration information of other terminals is not provided. In order to verify the accuracy and precision of our power consumption model, HIOKI 3334 measurement instrument is used to measure the power consumption of intelligent devices. The actual measured power consumption value is compared with the power consumption model proposed in this paper and the Xiaomi original power consumption model. During measurement, HIOKI 3334 is connected to the interface between the battery and the terminal of the mobile device, so as to realize the real-time measurement of the voltage and current of the intelligent terminal when running the tool; HIOKI 3334 can not only record the voltage and current of the intelligent terminal in real time but also record the instantaneous power of the device to be tested and the accumulated power consumption in a period of time.

The interface of our tool is presented in Figure 2; Figure 1(a) is the main interface of our tool. Power consumption information of each application in the system is displayed as a list. The pie chart on the right side of the list shows the percentage of current application power consumption occupying the total system power consumption, and the user can click any application to display the detailed power consumption information of the application, as shown in Figure 2(c). Figure 2(b) shows the power consumption statistical ranking list of Xiaomi native power consumption model, which only shows the application name and power consumption information in the system. Figure 2(c) shows the detailed power consumption information of the application, including the running time of the application, the detailed power consumption information of each sub-power-consumption model, and the total power

consumption information; through this figure, users can intuitively understand the detailed composition of power consumption of the application. Figure 2(d) shows the power consumption curve, wake-up lock, bright screen, and charging change of the system. The electric quantity curve shows the electric quantity loss in the form of broken line graph, with the horizontal axis as the time and the vertical axis as charge of the system. There are only two states in the vertical axis of wake-up lock, bright screen, and charging, and the horizontal axis is time: for the vertical axis of wake-up lock is wake-up and sleep, for the screen is the bright screen and the rest screen, and for the charging is charging and not charging. The main domain of the software includes black and white themes, and the default is white theme. If the user wants to switch, they can switch through the software settings; the black theme can reduce the irritation to eyes when users use it at night.

Figure 3 shows the power data comparison of our tool, model of Xiaomi native system, and the measured data of HIOKI 3334. In order to ensure the effectiveness and accuracy of the measurement, the two statistical applications are run repeatedly in a period of time, the running data of multiple runs are recorded and the average value is calculated, and the system is in the same state during each measurement. Figure 3 shows the power consumption comparison of the three measurement methods in a period of time interval when the terminal is tested for 20 times. For the application of WebChat, the error between the monitored power consumption value and the actual measurement value is 4.21%, and the measurement error with the original power model is 1.63%; for the second application, the error between the monitored value and the actual measurement value is 5.95%, and the measurement error is 0% with the original power consumption model; and, for the third application QQ, the error of the monitored value and the actual measurement value is

FIGURE 2: Different interface of our tool and Xiaomi native power tool. (a) Main interface. (b) Interface of Xiaomi. (c) Detailed information. (d) Battery graph.

FIGURE 3: Power consumption comparisons of the three measurement methods.

5.62%, and the error is 1.9% compared with the Xiaomi original power consumption model. The fourth application Alipay shows that the error between the monitored value and the actual measurement value is 4.08%, and the error between the monitored value and the Xiaomi original power consumption model is 0%. For the fifth application program MIUI, the error between the monitored value and the actual measurement value is 4.71%, and the error with the Xiaomi original power model is 4.71%. For the sixth application, the error between the monitored value and the actual measurement value is 2.18%, and the error with the Xiaomi original power model is 0.49%; for the seventh application, the error between the monitored value and the actual measured value is 2.16%, and the error with Xiaomi original power consumption model is 0%.

Through the measurement and analysis of seven applications, the error between the monitored value of our power consumption model and the actual measured value is 4.22%, and the error between the monitored value and the Xiaomi original power consumption model is 0.81%. Through the test of multiple terminals and multiple groups of applications, the error between the power consumption monitoring tool and the actual measurement is less than 5%. At the same time, the power consumption of the tool in this paper is low when collecting power consumption data, which has no impact on the operation of the system and other applications. Ordinary users can have a timely and comprehensive understanding of the power information of the application in the system and help Android application developers to optimize the power consumption of the application.

6. Conclusion

This paper presents a tool monitoring the power consumption of the intelligent terminal on the Android platform, and the power consumption model provided by Android is improved, especially the Bluetooth power consumption and hidden power consumption of the system and the detailed power consumption model is given. The power consumption statistics tool is designed and implemented in three layers: original information statistics layer, power calculation model layer, and power application layer. Finally, Xiaomi 9 Pro is used as the real test environment, and, compared with the actual measured value, the statistical error of our power consumption tool realized in this paper is less than 5%, which shows that the power consumption model and power consumption statistics tool in this paper meet actual needs, and it is helpful to understand the application power consumption information of intelligent terminal and optimize the terminal power consumption.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the State Key Program of the National Natural Science Foundation of China under Grant nos. 61802162 and 41901297, the Science and Technology Key Project of Henan Province under Grant nos. 202102210370 and 202102310017, and the Key Research Projects for the Universities of Henan Province under no. 20A170013.

References

- [1] Worldwide mobile phone forecast update," 2020, <https://www.idc.com/>.
- [2] A. Almasri and L. Gouveia, "S3: a study on the efficiency of current power-saving approaches used among android-application development and usage stages," *International Journal of Computing and Digital Systems*, vol. 9, no. 4, pp. 725–733, 2020.
- [3] B. Pradeep, P. Kulkarni, and M. Khan, "Comfort monitoring: a prototype and its impact," in *Proceedings of the 2020 IEEE International Conference on Semiconductor Electronics (ICSE)*, pp. 185–188, Kuala Lumpur, Malaysia, July 2020.
- [4] A. Almasri and A. Sameh, "Rating google-play apps' energy consumption on android smartphones," in *Proceedings of the 2019 2nd IEEE Middle East and North Africa Communications Conference (MENACOMM)*, pp. 1–6, Manama, Bahrain, November 2019.
- [5] A. Schuler and G. Anderst-Kotsis, "Examining the energy impact of sorting algorithms on android: an empirical study," in *Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pp. 404–413, Houston, TX, USA, November 2019.
- [6] Y. Zhang, Y. Li, T. Tan, and J. Xue, "Ripple: reflection analysis for android apps in incomplete information environments," *Software: Practice and Experience*, vol. 48, no. 8, pp. 1419–1437, 2018.
- [7] L. He, G. Meng, Y. Gu et al., "Battery-aware mobile data service," *IEEE Transactions on Mobile Computing*, vol. 16, no. 6, pp. 1544–1558, 2016.
- [8] A. Cañete, J. M. Horcas, I. Ayala, and L. Fuentes, "Energy efficient adaptation engines for android applications," *Information and Software Technology*, vol. 118, Article ID 106220, 2020.
- [9] S. K. Datta, C. Bonnet, and N. Nikaein, "Android power management: current and future trends," in *Proceedings of the 2012 the First IEEE Workshop on Enabling Technologies for Smartphone and Internet of Things (ETSIoT)*, pp. 48–53, Seoul, Korea (South), June 2012.
- [10] M. U. Farooq, S. U. R. Khan, and M. O. Beg, "MELTA: a method level energy estimation technique for android development," in *Proceedings of the 2019 International Conference on Innovative Computing (ICIC)*, pp. 1–10, Lahore, Pakistan, November 2019.
- [11] Z. Dai, W. Wang, and Y. Wu, "Static energy consumption analysis for android applications," in *Proceedings of the IOP Conference Series: Earth and Environmental Science*, vol. 512, no. 1, Article ID 012011, Changchun, China, June 2020.
- [12] N. Amsel and B. Tomlinson, "Green tracker: a tool for estimating the energy consumption of software," in *CHI'10 Extended Abstracts on Human Factors in Computing Systems*, pp. 3337–3342, Association for Computing Machinery, New York, NY, USA, 2010.
- [13] Y. Li, H. Chen, and W. Shi, "Power behavior analysis of mobile applications using bugu," *Sustainable Computing: Informatics and Systems*, vol. 4, no. 3, pp. 183–195, 2014.
- [14] L. Zhang, B. Tiwana, R. P. Dick et al., "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proceedings of the 2010 IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS)*, pp. 105–114, Scottsdale, AZ, USA, October 2010.
- [15] L. T. Duan, B. Guo, Y. Shen, Y. Wang, and W. L. Zhang, "Energy analysis and prediction for applications on smartphones," *Journal of Systems Architecture*, vol. 59, no. 10, pp. 1375–1382, 2013.
- [16] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proceedings of the USENIX Annual Technical Conference*, vol. 14, p. 21, Boston, MA, USA, June 2010.
- [17] D. Li, S. Hao, W. G. J. Halfond, and R. Govindan, "Calculating source line level energy information for android applications," in *Proceedings of the 2013 International Symposium on Software Testing and Analysis*, pp. 78–89, Lugano, Switzerland, July 2013.
- [18] C. Yoon, D. Kim, W. Jung et al., "Appscope: application energy metering framework for android smartphone using kernel activity monitoring," in *Presented as Part of the 2012 {USENIX} Annual Technical Conference ({USENIX}{ATC} 12)*, pp. 387–400, Boston, MA, USA, June 2012.