天珑Android手机充电记录软件V1.0

设计说明书

**深圳天珑无线科技有限公司**

目录

[第1章概述 3](#_Toc13875)

[1.1 软件产生的背景 3](#_Toc17087)

[1.2 软件功能的要求 3](#_Toc29715)

[第2章流程图 5](#_Toc20509)

[2.1 充电记录功能流程 5](#_Toc19741)

[2.2 充电解析功能流程 5](#_Toc16184)

[2.3 充电解析功能截图 6](#_Toc6213)

[第3章 实现过程 1](#_Toc32497)1

[3.1 充电记录实现 1](#_Toc13735)1

[3.2 记录启动方式 1](#_Toc20682)2

[3.3 记录服务保活 1](#_Toc834)3

[3.4 图表展示 1](#_Toc23331)4

[第4章 总结 1](#_Toc21714)6

# 第1章概述

## 1.1 软件产生的背景

由于手机性能的飞速提升，对于手机的续航要求也越来越高，单纯的加大电池容量已经不能完全满足需求目前Android高系统对于功耗的需求。为了准确了解用户对于手机续航的需求，从用户的充电习惯入手，便是一个很好的切入点。总结单个用户的用电、充电规律，才能统计产品整体的续航表现，后后续的产品作指导。因此便有了天珑Android手机充电记录软件的需求。

## 1.2 软件功能的要求

**监听模块**：监听所有跟电池充、放电时间有关的活动：

开机、关机时刻。

充电器插拔时刻，充满电时刻。

充电异常（如电量跳变）情况。

**过滤模块**：对于监听到的数据有效性进行过滤：

1.短时间内频繁地插拔充电器判定。

2.对于非常规事件（比如更换电池，时间异常等）导致信息异常的处理。

**记录模块**：对于过滤后数据的保存

1.保存的信息类型：暂定为时间，电量，充电状态

2.数据结构

A.二进制存储

单条信息存储格式和大小为：时间30bit + 电量7bit 状态2bit

一条信息占用内存5B，若用户平均每天充电5次，单个用户一年的数据量大小大致为20kb。 优点：数据量极小，方便存储和传输

缺点：无法直观地根据数据体现结果，数据需要2次处理。

**数据统计：**对过滤后数据的进一步分析与统计

1．计算每次充电和放点的效率：即在一次充放电间隔中，每单位电量，充电需要的时间（或可续航的时间）。

2．周充电频次：以第一条数据记录时间为基准，周为单位，记录每次充电次数

3．充电时时间与电量分布：统计开始充电时的时间与电量，与归类为4个充电时段和电量区间，统计次数分布。

4．周重启和关机次数：统计每周的重启和关机次数

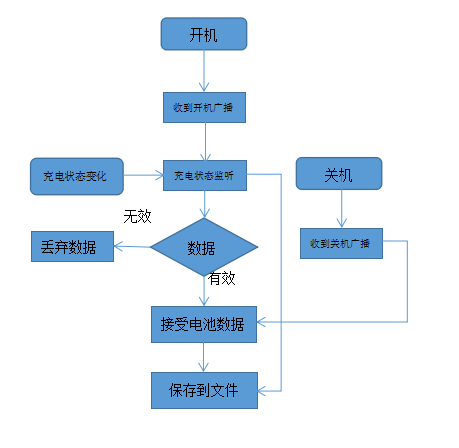
**结果展示：**将统计的结果以图表形式展现在apk

1. 周充电次数柱状图
2. 充电、放电效率波形图
3. 充电时间分布表+饼状图
4. 充电电量分布表+饼状图

**Excel保存：**将统计的结果保存成excel文件

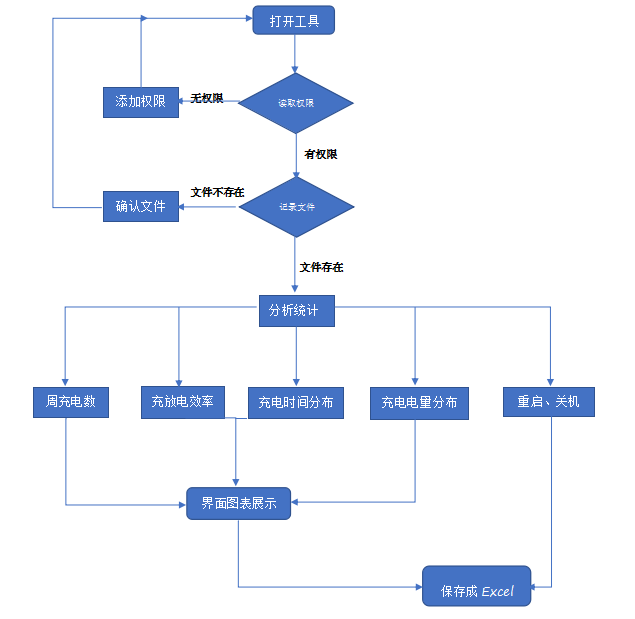
# 第2章流程图

## 2.1 充电记录功能流程



本程序流程图描述了充电记录的基本工作流程。记录功能的启动方式为监听开机广播自启动（第一次打开应用时也会启动），启动后会生成一个常驻后台的服务，用于监听电池状态的变化所发出的广播，如充电，放电，开关机等等。每当相应状态变化时，便会对此时抓取到的数据做一次判断，数据有效则记录，无效（如充电回充，充电时间过短等等）则过滤此条数据，最终生成一个包括手机基本信息，时间，电量，状态的一个记录文档并保存。

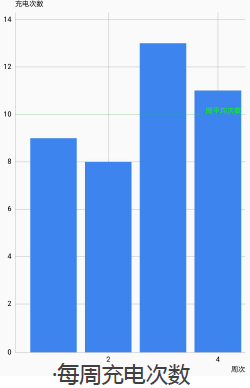
## 2.2 充电解析功能流程



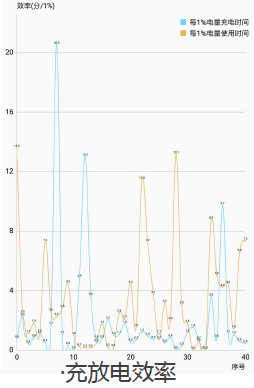
本程序流程图描述了充电解析的基本工作流程。首次打开应用后会获取相应的文件读取等权限，同时确认是否有充电记录的数据存在。当数据存在且有相应的权限后，便会进行下一步的解析操作。会把全部数据用在正则表达式匹配格式，并分割成N条原始数据，并分发给各个所需展示的图表。每个图表会根据各自的需求，再对原始数据进行处理，从而获得期望呈现的数据，并把这些数据绘制成图形。当需要导出数据到excel时，则再次分割成数据并填入对应单元格，同时对图表展现的数据规律做一定的数据描述。

## 2.3 充电解析功能截图

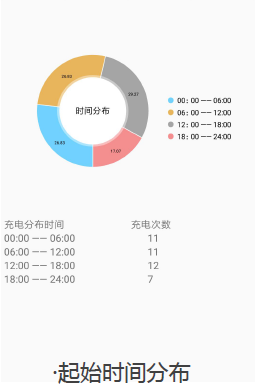
（1）周充电次数：以周为时间单位，统计用户的充电频率，并计算平均次数，展示用户的充电习惯。



（2）充放电效率：显示每次充电的效率，和每两次充电间的续航能力，以每1%电量的充电和使用时间来衡量。



（3）充电时间分布：将每天分为4个时间段，展示了用户的充电时间段分布情况，一定程度上可以体现用户的作息习惯。



1. 充电电量分布：描述用户充电时电量的分布，体现用户对于剩余电量的忍受程度和对于充电的需求，同样分为了4个电量区间。

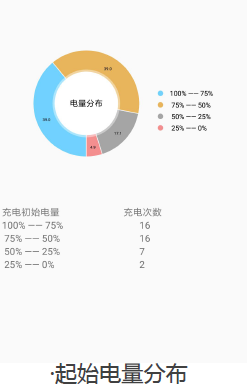
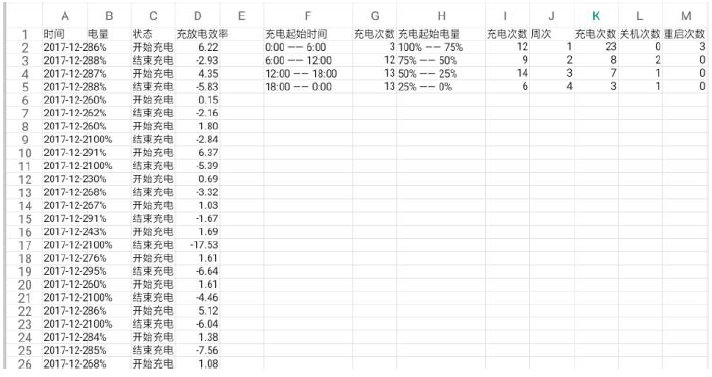


图5

1. Excel综合数据体现：详细地描述了用户的有效充电时间，电量，状态等等。并展现了上述四种图的原始数据和图上体现的数据来源。



## 2.3 轻触返回功能流程

指纹轻触返回与快速启动功能软件开机启动service。在sercice中初始化指纹触摸管理器。指纹触摸管理器开启一个线程循环调用poll方法，当监听驱动得到指纹按下和抬起消息后，读取指纹按下或抬起的消息类型，通过注册的回调函数传递给上层。上层接收到按下或抬起事件后首先判断是否打开了轻触返回功能的开关。如果没有打开，则直接返回不去处理。如果打开了，再去判断手机当前状态是否响应轻触返回。如果是锁屏或者录入指纹界面或者在通话状态则不去响应轻触返回。否则，程序向系统发出返回事件。系统收到事件后执行返回功能。

下图是程序流程图：

开始

启动service

初始化指纹触摸Manager

获取指纹触摸按下和抬起事件

轻触返回开关已经打开

N

Y

非指纹录入和锁屏状态

N

Y

按下抬起事件间隔小于300毫秒

N

Y

向系统发出返回事件

结束

# 实现过程

## 3.1 充电记录实现

每次收到充电变化的广播，会判断此次充电是否有效，并记录：

if (Intent.ACTION\_BATTERY\_CHANGED.equals(action)) {

PackageManager pm = getPackageManager();

boolean permission = (PackageManager.PERMISSION\_GRANTED ==

pm.checkPermission("android.permission.WRITE\_EXTERNAL\_STORAGE", "com.czc.android.chargerecord"));

if (permission == true){

if (onepxActivity == false) {

if (MainActivity.ifHideIcon == true){

Intent onrPXActivity = new Intent(context, OnePxActivity.class);

onrPXActivity.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);

startActivity(onrPXActivity);

Log.e("permision", "true");

}

}

}

BatteryN = intent.getIntExtra("level", 0); //目前电量（0~100）

switch (intent.getIntExtra("status", BatteryManager.BATTERY\_STATUS\_UNKNOWN)) {

case BatteryManager.BATTERY\_STATUS\_CHARGING:

if (firstBoot == true) {

saveToPath(fillTimeFormat.format(System.currentTimeMillis()) + " " + BatteryN + "% " + getString(R.string.restart\_charge) + "\n");

firstBoot = false;

}else {

if (!lastStatus.equals("charging")){

if (ifSaveCharge == true){

ifSaveCharge = false;

recordTime = fillTimeFormat.format(System.currentTimeMillis());

recordBatteryN = BatteryN;

lastStatus = "charging";

new Thread(new Runnable() {

public void run(){

try {

Thread.sleep(5000);

} catch (InterruptedException e) {

e.printStackTrace();

}

if (lastStatus.equals("charging")){

saveToPath(recordTime + " " + recordBatteryN + "% " + getString(R.string.start\_charge) + "\n");

saveCharge = true;

}

ifSaveCharge = true;

}

}).start();

}

}

}

lastStatus = "charging";

break;

case BatteryManager.BATTERY\_STATUS\_DISCHARGING :

notCharge();

break;

case BatteryManager.BATTERY\_STATUS\_NOT\_CHARGING:

notCharge();

break;

case BatteryManager.BATTERY\_STATUS\_FULL:

notCharge();

break;

case BatteryManager.BATTERY\_STATUS\_UNKNOWN:

break;

}

}

}

};

## 3.2 记录启动方式

开机自启动实现：

public class BootCompletedReceiver extends BroadcastReceiver {

public void onReceive(Context context, Intent intent) {

if(intent.getAction().equals(intent.ACTION\_BOOT\_COMPLETED)){

Log.e("ChargeAnalysisStart","from boot broad");

Intent bootIntent = new Intent(context, com.czc.android.chargeanalysis.record.StartService.class);

bootIntent.putExtra("boot", true);

context.startService(bootIntent);

}

}

}

....

## 3.3 记录服务保活

利用jobservice添加充电条件

JobInfo.Builder builder = new JobInfo.Builder(1, new ComponentName(getPackageName(), StartService.class.getName()));

builder.setRequiresCharging(true); // 充电状态

JobScheduler jobScheduler = (JobScheduler) this.getSystemService(Context.JOB\_SCHEDULER\_SERVICE);

jobScheduler.schedule(builder.build());

Log.e("jobservice","startJobSheduler");

在灭屏时启动一个activity，亮屏时销毁

public class OnePxActivity extends Activity {

private static final String TAG = "OnePxActivity" ;

protected BroadcastReceiver receiver = new BroadcastReceiver() {

@Override

public void onReceive(Context context, Intent intent) {

// 收到广播

// OnePxActivity.this.finish();

Log.e(TAG, "moveTaskToBack");

moveTaskToBack(true);

}

};

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

com.czc.android.chargeanalysis.record.StartService.onepxActivity = true;

Window window = getWindow();

// 设置窗口位置在左上角

window.setGravity(Gravity.LEFT | Gravity.TOP);

WindowManager.LayoutParams params = window.getAttributes();

params.x = 0;

params.y = 0;

params.width = 1;

params.height = 1;

window.setAttributes(params);

// 动态注册广播，这个广播是在屏幕亮的时候，发送广播，来最小化当前的Activity

registerReceiver(receiver, new IntentFilter("FinishActivity"));

Log.e(TAG, "start");

moveTaskToBack(true);

Log.e(TAG, "toback");

}

protected void onResume() {

super.onResume();

Log.e(TAG, "toback");

moveTaskToBack(true);

}

@Override

protected void onDestroy() {

unregisterReceiver(receiver);

Log.e(TAG, "end");

super.onDestroy();

}

}

## 3.4 充电记录图表展示

将充电数据按正则表达式读取并解析后，按照图表需要的形式在进行转换，最后调用mpchart和jx进行图表输出l。代码如下：

读取数据：

private String[] splitRecord(String rawRecord) {

String totalStandardData = "";

String regularExpression = "[1-9][0-9][0-9][0-9]-[0-1][0-9]-[0-3][0-9] [0-2][0-9]:[0-5][0-9]:[0-5][0-9] "

+ "\\d{1,3}%" + " [^x00-xff]{4,4}\n";

Pattern p = Pattern.compile(regularExpression);

Matcher m = p.matcher(rawRecord);

while (m.find()){

totalStandardData = totalStandardData + m.group() + " ";

}

String [] processedRecord = null;

processedRecord = totalStandardData.split(" ");

for (int i = 0; i < processedRecord.length; i++){

Log.e(TAG, processedRecord[i]);

}

return processedRecord;

}

public boolean start(int s) {

if(s == 0) {

start();

}

else {

Fp\_native fn = new Fp\_native();

fn.init(1, fpEventManager.this);

}

return true;

}

分发数据

public Bundle deliverData() {

//transfer format for bundle

double[] efficiencyBundle = new double[efficiency.size()];

for (int i = 0; i < efficiency.size(); i++){

efficiencyBundle[i] = efficiency.get(i);

}

Bundle bundle = new Bundle();

Log.e("weektime", String.valueOf(weekTimesData.size()));

bundle.putIntegerArrayList("weekTimesData", weekTimesData);

bundle.putDoubleArray("timeDistribute", timeDistributeArray);

bundle.putDoubleArray("powerDistribute", powerDistributeArray);

bundle.putDoubleArray("efficiency", efficiencyBundle);

return bundle;

}

图表展现

private LineDataSet drawDataset(ArrayList<Double> data, int color, String labelle) {

int colorNumber = color;

yVals = new ArrayList<>();

if (data.size() > 0){

for (int i = 0; i < data.size(); i++){

yVals.add(new Entry(i, data.get(i).floatValue() / 100));

}

}

dataSet = new LineDataSet(yVals, labelle);

//dataSet.setColors();

dataSet.setColors(new int[]{COLORS[colorNumber]});

dataSet.setDrawFilled(false);

dataSet.setCircleSize(1.0F);

dataSet.setCircleColor(COLORS[colorNumber]);

dataSet.setDrawCircleHole(false);

dataSet.setValueTextSize(4.0F);

dataSet.setMode(LineDataSet.Mode.CUBIC\_BEZIER);

return dataSet;

}

Excel输出

private void addAverageBatteryLife() throws WriteException {

double expendPower = 0;

double expendTime = 0;

//构造格式：ARIAL字体、10号、粗体、非斜体、无下划线、黑色

WritableFont wordsFont = new WritableFont(WritableFont.ARIAL,10,WritableFont.NO\_BOLD,

false, UnderlineStyle.NO\_UNDERLINE, Colour.BLACK);

WritableCellFormat format = new WritableCellFormat(wordsFont);

//文字垂直居中对齐

format.setAlignment(Alignment.RIGHT);

for (int i = 0; i < stateRecord.size() - 1; i++){

expendTime = (timeToLong(timeRecord.get(i + 1)) - timeToLong(timeRecord.get(i))) / 1000;

expendPower = powerToInt(powerRecord.get(i + 1)) - powerToInt(powerRecord.get(i));

if (expendPower != 0){

Log.e("expendPower", String.valueOf(expendPower));

try {

DecimalFormat df = new DecimalFormat("0.00");

sheet.addCell(new Label(3, i + 1, df.format(expendTime / expendPower / 60), format));

} catch (WriteException e) {

e.printStackTrace();

}

}else {

try {

sheet.addCell(new Label(3, i + 1, "0", format));

} catch (WriteException e) {

e.printStackTrace();

}

}

}

}

# 总结

天珑Android手机充电记录软件界面友好，操作简单。针对充放电时间和效率，进一步精准记录用户插拔充电和开关机的具体行为，为后续功耗工作提供依据，同时也可为个人用户总结手机用电规律。是一款既能用于功耗分析，又可提供给用户个人使用的好软件。