**Battery Manager App Project Report**

**Crystle Yi**

**Zhisheng Yan**

**CSC 4320**

**Introduction**

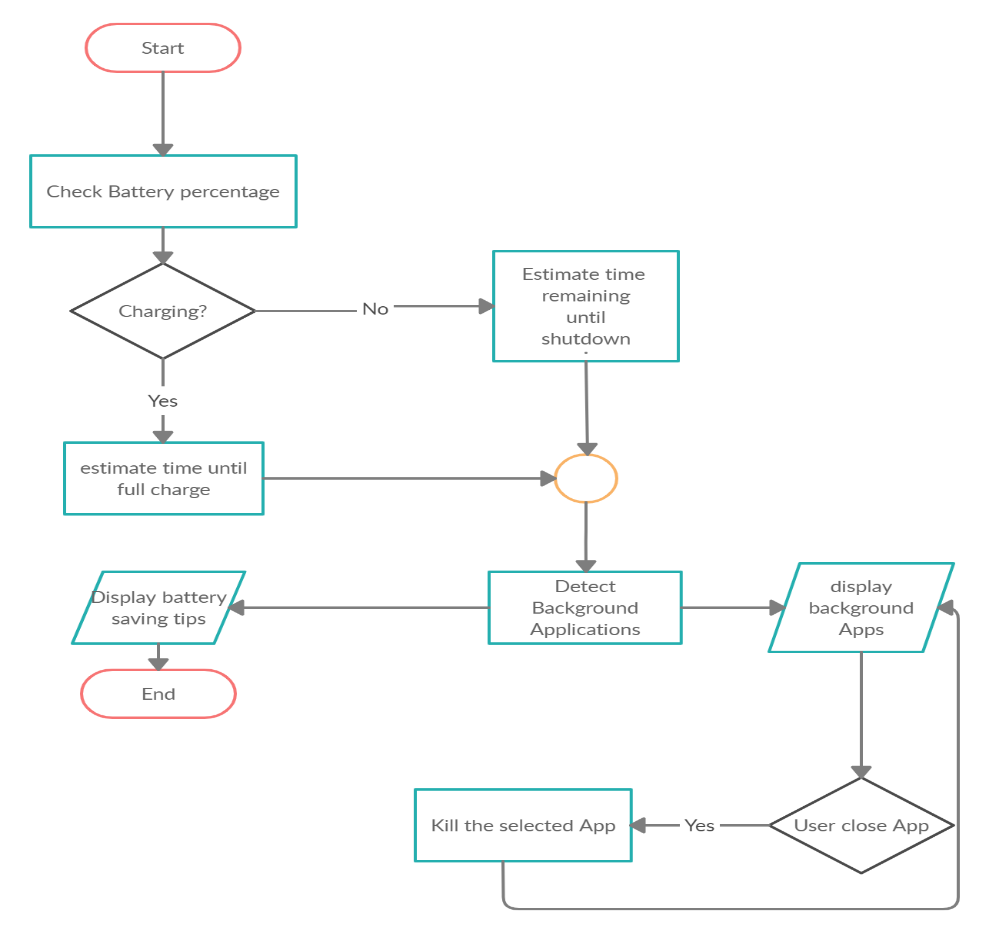
Power management is paramount for all android devices used across the world. The android devices are installed with power-consuming technologies such as WIFI, GPS, and 3G. All these technologies and streaming apps like YouTube consume more power that drains the power of the device. There are also so many other apps that start automatically in android phones, making it even hard for the user to know which apps are draining the battery's power as they are running in the background. Developers choose to allow apps to run in the background to be easy for them to improve the application's features and function in the future. Most end users find it difficult to determine which applications are using more energy.

This paper describes a Battery manager application that aims to determine the amount of power used by all applications in an android smartphone. The Battery manager application will monitor the amount of battery power remaining and enable the user to know the approximate time remaining before the phone shut down. The application will provide tips to the end-user for power saving and integrate functionality that will enable them to close the application draining the battery power while they are not being used.

**Application design**

For me to analyze the consumption of power of each application, I designed a power management system. First, the user launches the application on a smartphone. Second, the application will determine the amount of battery power remaining and display it on the screen. The user will be able to see the amount of power remaining in the device. The first screen will also include tips to help the user minimize power usage. The user can go ahead and check the amount of power consumed by each application. At this step, the user can cross the applications that are draining the battery, yet they are not using them. Finally, the user closes the application. The battery viewer system model is used to check the battery information and applications' power usage information.

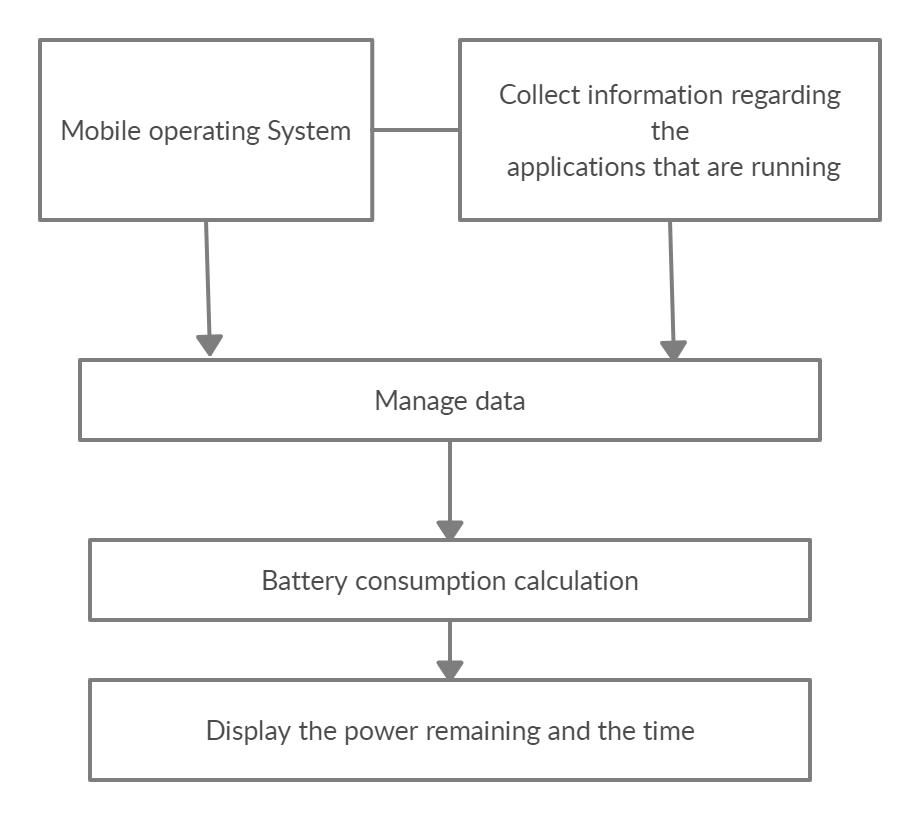
**Flow Chart**

****

The user launches the application, and the application automatically checks the battery percentage. If the phone is charging, it will display the "Device charging" text and show the time remaining until it's fully charged. If it is not charging, the app will display the "Device not charging" text and show the estimated time remaining until the phone shuts down. Then the app will detect the apps draining the battery power. User can kill the apps that are draining battery power, yet he/she is not using it.

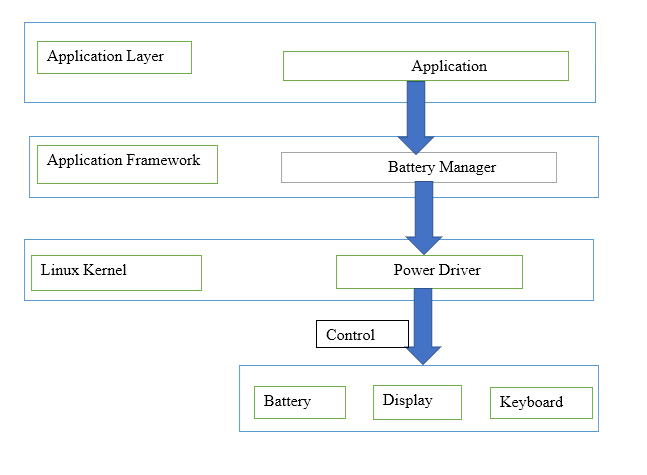
**Block Diagram**

The Battery manager application was developed using the software development kit (SDK) in android studio software. The block diagram was developed using based on the series of events in the flow chart. The batter manager application runs on any android based application. The application records information regarding the device. It checks on the WIFI status, GPS location, and other features that may be draining the battery power. It suggests useful tips to help minimize battery usages, such as switching off WIFI, GPS location, and others. The figure below shows how the application will be operating.



**The Application Architecture**

The figure below shows the architecture of the application. When the application is launched, it requires to use of the CPU resources, so it requires the power manager API in the application framework. The application framework turns transfers using java native interface, which requests the power driver to in the Linux Kernel.

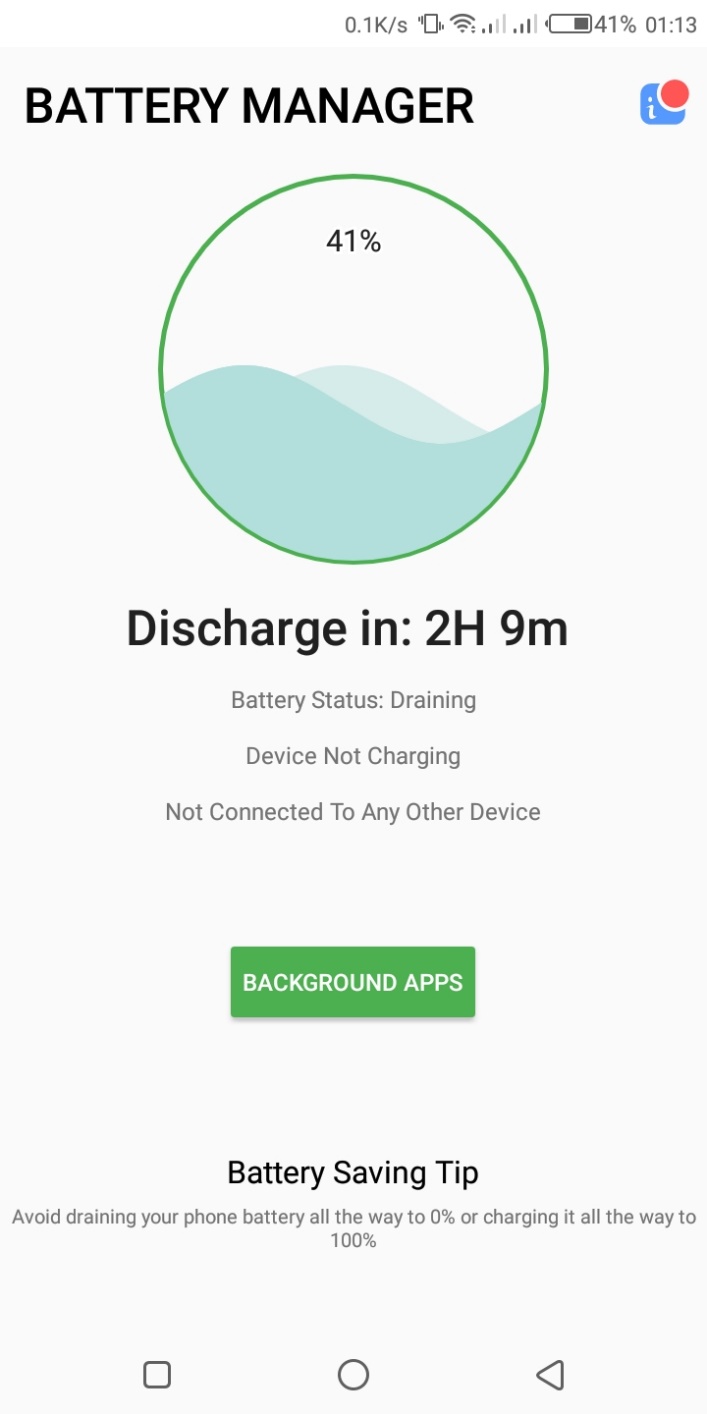
****

**User Interface**

The user interface represents the screenshots of the applications to show how it is working step by step.

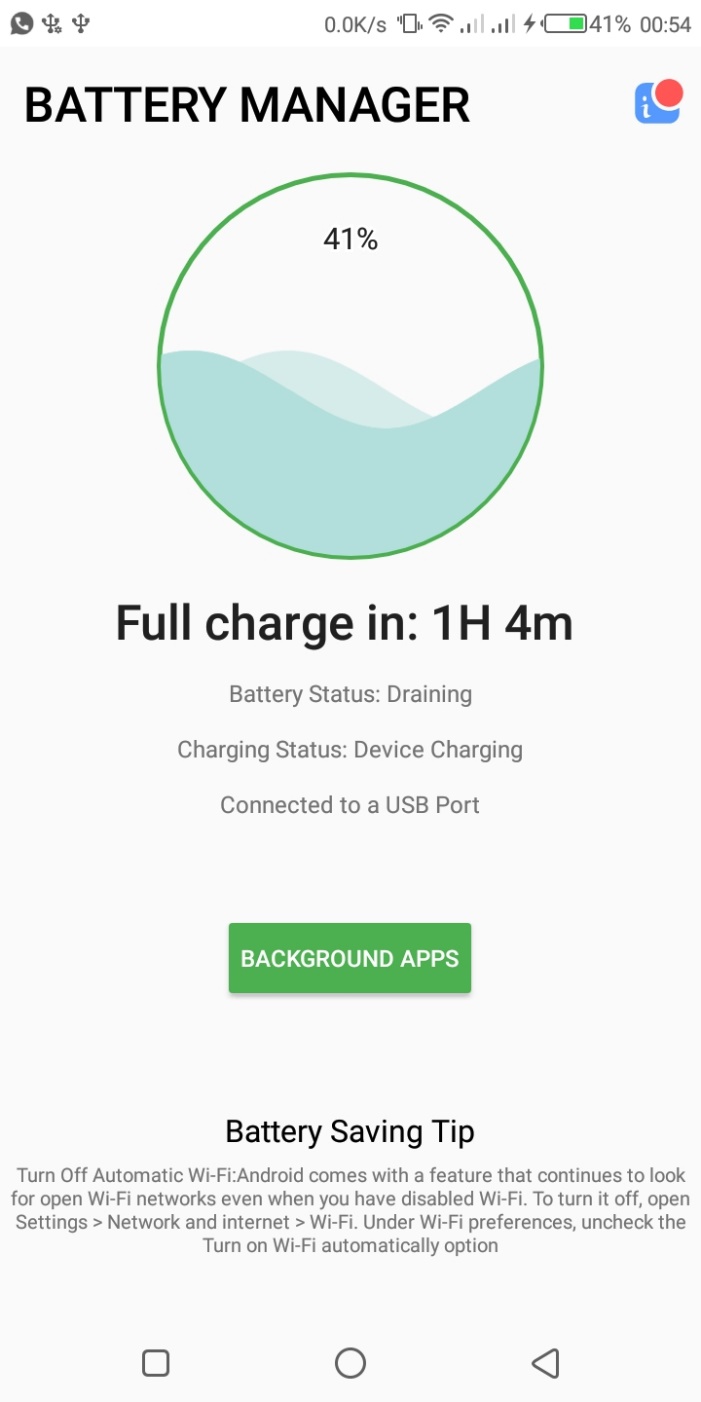
**First Screen**

This is the first screen to display when the user launches the application. If the charger is not connected, it will show the battery percentage remaining in a graph. It shows the time remaining until the phone shuts down. The application will also suggest useful tips to reduce battery consumption. On the battery status, the application will display “Draining” text to show that the app is discharging.

****

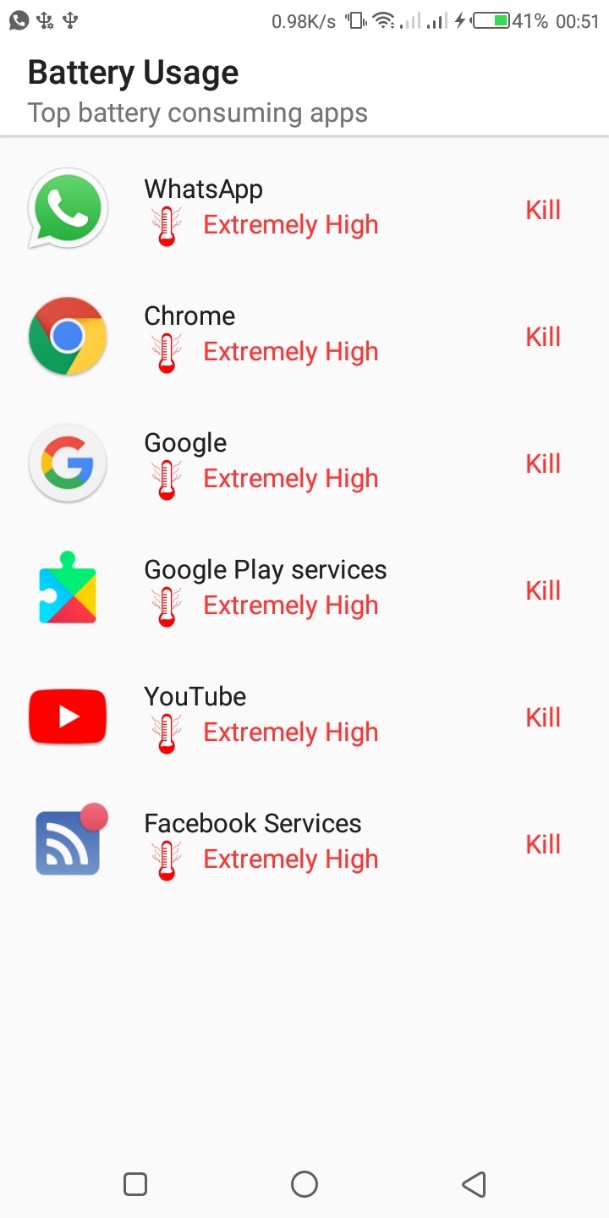
**Second Screen**

This screen appears when the phone is connected to a charger. It displays the "Device Charging" text, which changes to "Device not charging" when the AC power source is disconnected. It also suggests to the user a battery-saving tip.

****

**Third Screen**

This screen appears when the user clicks the "background apps" button on the launching screen. It shows a list of background apps that are consuming the battery power in descending order. The application provides functionality that the user can close the apps consuming high battery power, but it is not in use.

****

**Conclusion**

The power management application is beneficial to the user as they can track battery changes in their smartphones. The application shows whether the device is charging or not, while at the same time, it shows the time remaining until the phone shuts down. When charging, the application shows the amount of time remaining until the battery is fully charged. Further, the application shows the amount the applications that are consuming power in the background. This application is beneficial to the users as they can manage the battery power. It provides functionality that the users can close those apps that are draining the battery power in the backgrounds, yet the user is not using them. Moreover, the application is beneficial to the user as they get real-time tips on minimizing power usage.

**Source Code for one of the java files:**

package com.intelligentSoftwares.Trichain.batterymanager;

import android.content.BroadcastReceiver;

import android.content.Context;

import android.content.Intent;

import android.content.IntentFilter;

import android.graphics.Color;

import android.os.BatteryManager;

import android.os.Bundle;

import android.os.Handler;

import android.support.v7.app.AppCompatActivity;

import android.util.Log;

import android.view.View;

import android.widget.Button;

import android.widget.ImageView;

import android.widget.TextView;

import java.util.Calendar;

import java.util.Date;

import java.util.Random;

import java.util.Timer;

import java.util.TimerTask;

import me.itangqi.waveloadingview.WaveLoadingView;

public class MainActivity extends AppCompatActivity {

    int i;

    boolean isCharging, usbCharge, acCharge;

    WaveLoadingView mWaveLoadingView;

    TextView txtbattery\_status, txtstatus, txtcharge\_status, txtport, tips;

    private int progressStatus = 0;

    private Handler handler = new Handler();

    private Button run\_app;

    Timer timer = new Timer();

    private String randomStr;

    private int cnt = 0;

    private float batteryLevel1;

    private float batteryLevel2;

    Date tym1;

    private String TAG = "Intelligent-Apps";

    private ImageView imgUsage;

    private boolean isTrue = true;

    private String charging, chargingMethod;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

        IntentFilter iFilter = new IntentFilter(Intent.ACTION\_BATTERY\_CHANGED);

        registerReceiver(mBroadcastReceiver, iFilter);

        mWaveLoadingView = (WaveLoadingView) findViewById(R.id.waveLoadingView);

        mWaveLoadingView.setShapeType(WaveLoadingView.ShapeType.CIRCLE);

        txtbattery\_status = (TextView) findViewById(R.id.txtbattery\_status);

        run\_app = (Button) findViewById(R.id.run\_app);

        txtstatus = (TextView) findViewById(R.id.txtstatus);

        txtcharge\_status = (TextView) findViewById(R.id.txtcharge\_status);

        txtport = (TextView) findViewById(R.id.txtport);

        tips = findViewById(R.id.tips);

        imgUsage = findViewById(R.id.imgUsage);

        imgUsage.setOnClickListener(new View.OnClickListener() {

            @Override

            public void onClick(View v) {

                imgUsage.setImageResource(R.drawable.ic\_info);

                Intent i = new Intent(MainActivity.this, TopBatteryApps.class);

                i.putExtra("first", isTrue);

                isTrue = !isTrue;

                startActivity(i);

            }

        });

        run\_app.setOnClickListener(new View.OnClickListener() {

            @Override

            public void onClick(View v) {

                startActivity(new Intent(MainActivity.this, RunningAppsActivity.class));

            }

        });

        i = getBatteryLevel(getApplicationContext());

        //battery level conditions

        if (i >= 50 && i <= 75) {

            txtstatus.setText("Battery Status: Moderate");

        } else if (i >= 75 && i <= 99) {

            txtstatus.setText("Battery Status: GOOD");

        } else if (i >= 20 && i <= 49) {

            txtstatus.setText("Battery Status: Draining");

        } else if (i >= 10 && i <= 20) {

            txtstatus.setText("Battery Status: LOW");

        } else if (i < 10) {

            txtstatus.setText("Battery Status: Critically Low, Please Charge the Device!");

        } else if (i == 100) {

            txtstatus.setText("Battery Status: Fully Charged");

        }

        mWaveLoadingView.setTopTitle(i + "%");

        mWaveLoadingView.setCenterTitleColor(Color.WHITE);

        mWaveLoadingView.setCenterTitle("");

        mWaveLoadingView.setBottomTitleSize(18);

        mWaveLoadingView.setProgressValue(0);

        mWaveLoadingView.setBorderWidth(5);

        mWaveLoadingView.setAmplitudeRatio(100);

        mWaveLoadingView.setWaveColor(Color.parseColor("#B2DFDB"));

        mWaveLoadingView.setBorderColor(Color.parseColor("#4CAF50"));

        mWaveLoadingView.setTopTitleStrokeColor(Color.WHITE);

        mWaveLoadingView.setTopTitleStrokeWidth(3);

        mWaveLoadingView.setWaterLevelRatio(0.2f);

        mWaveLoadingView.setAnimDuration(3000);

        mWaveLoadingView.pauseAnimation();

        mWaveLoadingView.resumeAnimation();

        mWaveLoadingView.cancelAnimation();

        mWaveLoadingView.startAnimation();

        Timer timer = new Timer();

        timer.scheduleAtFixedRate(new TimerTask() {

            @Override

            public void run() {

                handler.post(new Runnable() {

                    public void run() {

                        progressStatus = getBatteryLevel(getApplicationContext());

                        mWaveLoadingView.setProgressValue(progressStatus);

                    }

                });

                runOnUiThread(new Runnable() {

                    @Override

                    public void run() {

                        String x = String.valueOf(i);

                        getChargingLevel(getApplicationContext());

                        //Battery charging conditions

                        if (isCharging == true && usbCharge == true) {

                            txtcharge\_status.setText("Charging Status: Device Charging");

                            txtport.setText("Connected to a USB Port");

                        } else if (isCharging == true && acCharge == true) {

                            txtcharge\_status.setText("Charging Status: Device Charging");

                            txtport.setText("Connected to an AC power source");

                        } else {

                            txtcharge\_status.setText("Device Not Charging");

                            txtport.setText("Not Connected To Any Other Device");

                        }

                        if (isCharging == true && i == 100) {

                            txtcharge\_status.setText("Charging Status: Charging Complete");

                        }

                    }

                });

            }

        }, 0, 100);

        startTipsTimer();

    }

    private BroadcastReceiver mBroadcastReceiver = new BroadcastReceiver() {

        @Override

        public void onReceive(Context context, Intent intent) {

            Log.e(TAG, "onReceive: ");

            int scale = intent.getIntExtra(BatteryManager.EXTRA\_SCALE, -1); //get the scale of battery (usually 100)

            int level = intent.getIntExtra(BatteryManager.EXTRA\_LEVEL, -1); //get the level of the battery

            // Are we charging or is the phone fully charged?

            int status = intent.getIntExtra(BatteryManager.EXTRA\_STATUS, -1);

            boolean isCharging = status == BatteryManager.BATTERY\_STATUS\_CHARGING ||

                    status == BatteryManager.BATTERY\_STATUS\_FULL;

            float percentage = level / (float) scale;

            if (cnt < 1) {

                batteryLevel2 = percentage + (float) 0.0001;

                batteryLevel1 = percentage - (float) 0.0001;

                tym1 = Calendar.getInstance().getTime();

                Log.e(TAG, "onReceive: tym1 " + tym1.getTime());

                //get Time 1

                cnt++;

            }

            if (isCharging) {

                Log.e(TAG, "onReceive: charging");

                Log.e(TAG, "onReceive: percent" + percentage);

                Log.e(TAG, "onReceive: batterylevel2" + batteryLevel2);

                Log.e(TAG, "onReceive: charging");

                charging = "Charging";

                if (percentage >= batteryLevel2) {

                    //get Time 2

                    float p\_left = (float) 1 - percentage;

                    Date tym2 = Calendar.getInstance().getTime();

                    Log.e(TAG, "onReceive: tym2 " + tym2.getTime());

                    long tym\_diff = tym2.getTime() - tym1.getTime();

                    long mills = Math.abs(tym\_diff);

                    long rem = (long) (mills \* p\_left \* 100);

                    int hours = (int) (rem / (1000 \* 60 \* 60));

                    int mins = (int) (rem / (1000 \* 60)) % 60;

                    long secs = (int) (rem / 1000) % 60;

                    Log.e(TAG, "onReceive: " + hours);

                    txtbattery\_status.setText("Full charge in: " + hours + "H " + mins + "m ");

                    cnt = 0;

                } else {

                    Log.e(TAG, "onReceive: else");

                    //get Time 2

                    float p\_left = (float) 1 - percentage;

                    Date tym2 = Calendar.getInstance().getTime();//13245233546

                    Log.e(TAG, "onReceive: tym2 " + tym2.getTime());

                    long tym\_diff = tym2.getTime() - (tym1.getTime() - new Random().nextInt(65758));

                    Log.e(TAG, "onReceive: time diff" + tym\_diff);

                    long mills = Math.abs(tym\_diff);

                    Log.e(TAG, "onReceive: abs millis " + mills);

                    long rem = (long) (mills \* p\_left \* 100);

                    Log.e(TAG, "onReceive: remaining millis " + rem);

                    int hours = (int) (rem / (1000 \* 60 \* 60));

                    int mins = (int) (rem / (1000 \* 60)) % 60;

                    long secs = (int) (rem / 1000) % 60;

                    Log.e(TAG, "onReceive: h " + hours);

                    Log.e(TAG, "onReceive: m " + mins);

                    txtbattery\_status.setText("Full charge in: " + hours + "H " + mins + "m ");

                    cnt = 0;

                }

            } else {

                Log.e(TAG, "onReceive: not charging");

                charging = "Not Charging";

                if (percentage <= batteryLevel1) {

                    //get Time 2

                    Date tym2 = Calendar.getInstance().getTime();

                    long tym\_diff = tym2.getTime() - tym1.getTime();

                    long mills = Math.abs(tym\_diff);

                    long rem = (long) (mills \* percentage \* 100);

                    int hours = (int) (rem / (1000 \* 60 \* 60));

                    int mins = (int) (rem / (1000 \* 60)) % 60;

                    long secs = (int) (rem / 1000) % 60;

                    txtbattery\_status.setText(hours + "H" + mins + "m" + secs + "s");

                    cnt = 0;

                } else {

                    Log.e(TAG, "onReceive: else");

                    //get Time 2

                    float p\_left = (float) 1 - percentage;

                    Date tym2 = Calendar.getInstance().getTime();//13245233546

                    Log.e(TAG, "onReceive: tym2 " + tym2.getTime());

                    long tym\_diff = tym2.getTime() - (tym1.getTime() - 65758 \* 2);

                    Log.e(TAG, "onReceive: time diff" + tym\_diff);

                    long mills = Math.abs(tym\_diff);

                    Log.e(TAG, "onReceive: abs millis " + mills);

                    long rem = (long) (mills \* p\_left \* 100);

                    Log.e(TAG, "onReceive: remaining millis " + rem);

                    int hours = (int) (rem / (1000 \* 60 \* 60));

                    int mins = (int) (rem / (1000 \* 60)) % 60;

                    long secs = (int) (rem / 1000) % 60;

                    Log.e(TAG, "onReceive: h " + hours);

                    Log.e(TAG, "onReceive: m " + mins);

                    txtbattery\_status.setText("Discharge in: " + hours + "H " + mins + "m ");

                }

            }

        }

    };

    private void startTipsTimer() {

        timer.scheduleAtFixedRate(

                new TimerTask() {

                    @Override

                    public void run() {

                        runOnUiThread(new Runnable() {

                            @Override

                            public void run() {

                                // Stuff that updates the UI

                                String[] array = getResources().getStringArray(R.array.tips);

                                randomStr = array[new Random().nextInt(array.length)];

                                Log.i("tag", "Tip" + randomStr);

                                tips.setText(randomStr);

                            }

                        });

                    }

                },

                0,

                60000);

    }

    public int getBatteryLevel(Context context) {

        IntentFilter ifilter = new IntentFilter(Intent.ACTION\_BATTERY\_CHANGED);

        Intent batteryStatus = context.registerReceiver(null, ifilter);

        return i = batteryStatus.getIntExtra(BatteryManager.EXTRA\_LEVEL, -1);

    }

    //Checking the charging status.

    public void getChargingLevel(Context bl) {

        IntentFilter ifilter = new IntentFilter(Intent.ACTION\_BATTERY\_CHANGED);

        Intent batteryStatus = bl.registerReceiver(null, ifilter);

        int status = batteryStatus.getIntExtra(BatteryManager.EXTRA\_STATUS, -1);

        isCharging = status == BatteryManager.BATTERY\_STATUS\_CHARGING ||

                status == BatteryManager.BATTERY\_STATUS\_FULL;

        int chargePlug = batteryStatus.getIntExtra(BatteryManager.EXTRA\_PLUGGED, -1);

        usbCharge = chargePlug == BatteryManager.BATTERY\_PLUGGED\_USB;

        acCharge = chargePlug == BatteryManager.BATTERY\_PLUGGED\_AC;

    }

}

Link to the project: <https://github.com/NinjaBuddyYin/OperatingSystemProject>

**References**

Awal, A., Mishra, A., Joseph, E., & AbdulG, A. (2018). A POWER MANAGEMENT MODEL FOR ANDROID OPERATED SMART PHONES. *A POWER MANAGEMENT MODEL FOR ANDROID OPERATED SMART PHONES*, *3*(1), 15-15.

Zaman, N., & Almusalli, F. A. (2017, April). Smartphones power consumption & energy saving techniques. In *2017 International Conference on Innovations in Electrical Engineering and Computational Technologies (ICIEECT)* (pp. 1-7). IEEE.