Battery Saver App for Windows

Abhishek Kumar[1], Bishwajit Prasad Singh[1], and Pritam Ghosal[1]

[1]NSHM Knowledge Campus, Durgapur

Abstract: More than 70 percent of the world's population utilises the Windows Operating System, primarily due to its portability. Unlike desktop computers and TVs, which require a continuous connection to an electrical socket, Windows OS-powered devices are battery-powered and can be used anywhere. However, these devices still need to be charged at least once a day. The Windows Operating System is known for its resource-intensive nature, which leads to a rapid depletion of battery life. Extensive research has been conducted to improve device battery quality, but due to limited energy supply, these batteries struggle to support applications for an extended duration. Various methods have been explored to enhance power management, one of which involves using a battery saver application. This application assists users in reducing power consumption to some extent.

Keywords: Drowsiness, yawning, classifier, face detection.

1. Introduction

A Windows battery-saving software is designed to prolong your laptop's battery life. It achieves this by automatically deactivating power-hungry processes when you unplug the laptop from the power source. The primary method it employs is disabling certain Windows features. Additionally, it offers the

flexibility to toggle the sidebar, mute the system, and manage Windows power plans. All of these functions are conveniently accessible through a compact utility located in the taskbar tray. In situations where the battery level is critically low and you don't wish to shut down the laptop entirely, you can utilize the quick hibernation feature. However, it's essential to note that during hibernation, the laptop won't be operational. Once you reconnect the power supply, all power-saving features are automatically disabled. The software also retains your customised settings, allowing you to configure it once, and it is accessible via an icon in the taskbar tray. It was initially developed for Windows 7, but during testing, it has proven to work effectively with other Windows versions as well.

The work can be evaluated based on the following features:

- Intelligently enhancing your PC's performance.
- Automatically lowering your laptop's brightness.
- Automatically disabling power-consuming apps like Bluetooth and Wi-Fi when not in use.
- Auto-initiating functionality when the laptop is disconnected from its power source.
- Providing the ability to manually deactivate Aero Glass.
- Offering optional management of Aero Glass through a user-friendly interface.
- Optionally managing your preferred power plans.
- The choice to display a quick hibernate button.
- Adjusting icon color to indicate the power source, whether it's running on battery or connected to AC power.

2. Description

This innovative system is an application that utilizes built-in classes to create a user-friendly list for the user's review. The list includes information about battery usage and monitors the battery level. When the app consumption is high and the battery level is low, the system will activate an alarm, prompting the user to either force stop or close the apps. In essence, the system assists the user in preventing certain apps from excessively depleting the battery's power and provides the user with actionable insights.

3. Methodology

The overall methodology has been broken down into several smaller modules, which are elaborated on in this section. The proposed algorithm operates in seven stages, as illustrated in Figure 1. The video feed used for testing this technique is sourced from the laptop's webcam.

PSUTIL, short for "process and system utilities," is a versatile Python library that functions across different platforms to gather information about running processes and system resource usage, including CPU, memory, disks, network, and sensors. Its primary applications include system monitoring, profiling, managing process resources, and overseeing active processes. PSUTIL incorporates numerous features reminiscent of traditional UNIX command line tools like ps, top, iotop, lsof, netstat, ifconfig, free, and others. PSUTIL currently provides support for a range of platforms, including Linux, Windows, MacOS, FreeBSD, OpenBSD, NetBSD, Sun Solaris, and AIX (compatible with both 32-bit and 64-bit architectures). It is compatible with Python versions 2.6, 2.7, and 3.4 or newer, and it is also known to work with PyPy. Within the main.py script, this module is employed to create a function that generates a list of running processes with CPU usage exceeding 0.0%. Selected processes can be terminated when the app user clicks the "run" button.

When a process is running on a machine, it has a unique number called pid i.e process identification number, this pid can be used to to manage a process, since this is a windows app, using windows command prompt or powershell, if you know a process pid, you can close it by running this command task kill /f /t /pid <replace with the pid you want> this will close the process using the specified pid. The /f means to force kill and /t means to kill all threads, you can decide to remove any of the options. So, this command is integrated in the app using the powerful python OS module.

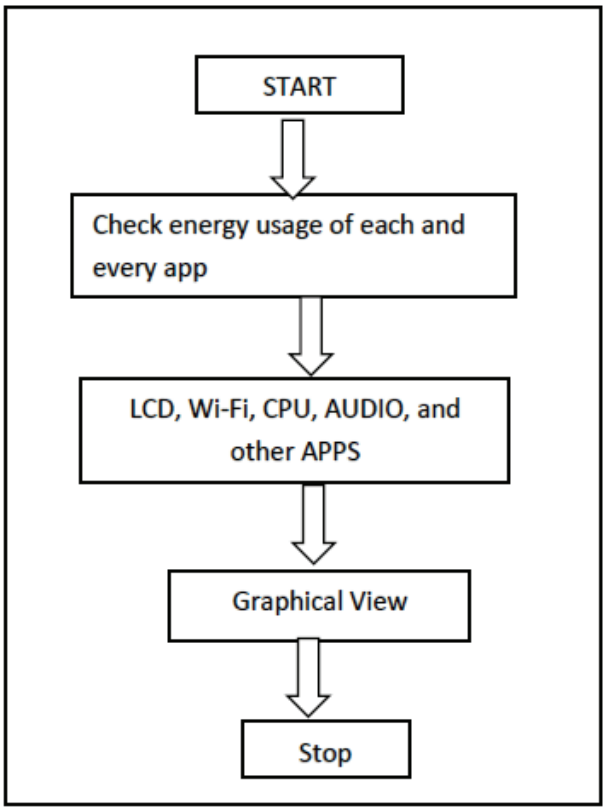

Figure 1: Flow chart of Battery viewer model

3.1 OS Module

OS module is a very big module, because you can use it for a large variety of stuffs and works almost the same on all types of OS, e.g windows, mac, linux etc. its the official module used to communicate directly with the OS of any machine. While working with this OS module, I used os.system() to run the taskkill command, as what the os.system() does is that, it will run any argument you put there like you're running it inside a commandline. This snippet was included in the class MyForm() running the user interface, see the kill_pid() function inside the battery_saver.py to get a hang of how this is used in the os.system() to call a taskkill on all the selected processes. Lastly, the app also reduces the screen brightness to 20% immediately when clicked on the run button.

3.2. Windows Management Instrumentation (WMI)

Windows Management Instrumentation (WMI) represents Microsoft's implementation of Web-Based Enterprise Management (WBEM), which is part of an industry-wide effort to establish a Common Information Model (CIM) covering virtually all aspects of a computer system. When it comes to the Python

WMI module, it serves as a lightweight layer built on the pywin.32 extensions, effectively concealing the intricate technical details required for Python to communicate with the WMI API. It is entirely implemented in Python and is compatible with any Python version from 2.1 onwards (supporting list comprehensions) and most recent versions of pywin32. I leveraged this module to adjust screen brightness within a Windows application, utilizing the WMI API for this purpose. The function responsible for this operation is called within the reduce brightness() function in the MyForm() class, which can be found in the battery_saver.py file.

4. Novelties in the Project:

The paper is entirely Python-based, utilizing select libraries, resulting in fast compilation and rapid outputs. It consolidates application usage data into a single, user-friendly interface, providing a comprehensive list of applications in use. Moreover, the system offers battery alerts to inform the user about low battery status and precisely identifies power-hungry apps. It delivers accurate consumption rate information and pinpoints the specific apps that are power-intensive. In contrast to the pre-existing battery saver tools found on most devices, this system stands out by providing proactive alarms and identifying the culprit apps responsible for power drainage, making it a similar yet distinct solution.

References

1. https://www.resurgentindia.com/paper-feasibility-study-and-its-importance-in-paper-management#
2. Battery Power Saving Profile with Learning Engine in Android Phones - Rimpy Bala & Anu Garg, International Journal of Computer Applications (0975 – 8887) Volume 69– No.13, May 2013
3. Web Performance with Android's Battery-Saver Mode - Utkarsh Goel, Stephen Ludin & Moritz Steiner