POWER MANAGER TELEMTRY

By: Version Venture

Reported By: Manisha Chillapur

Place: Kalaburgi

Date: 07-07-2024

# POWER MANAGER TELEMETRY

Power manager telemetry typically refers to the process of monitoring and collecting data related to power consumption, battery levels, and operational status of devices within a system. In the context of software development or engineering, especially in fields like IoT (Internet of Things) or embedded systems, power manager telemetry involves:

1. **Monitoring Power Consumption**: Tracking how much power each device or component consumes over time.
2. **Battery Management**: Monitoring the state of batteries, including remaining capacity, charging status, and health.
3. **Operational Status**: Gathering information on device uptime, downtime, and operational efficiency based on power usage.
4. **Alerting and Reporting**: Notifying stakeholders about critical events such as low battery levels or abnormal power consumption patterns.
5. **Data Analysis**: Analysing telemetry data to optimize power usage, predict battery life, and improve device performance.

In software applications, a power manager module or system would typically interface with hardware components to gather real-time telemetry data, process it, and provide actionable insights for efficient power management and device operation. This helps in maximizing device uptime, optimizing energy usage, and enhancing overall system reliability.

This report documents the design and implementation of a Java program for power management telemetry. The program simulates a power management system for multiple devices, allowing users to monitor and manage their power consumption and battery levels.

## Program Overview

The Java program consists of three main classes: Device, PowerManager, and Main. These classes interact to simulate device power usage, monitor telemetry data, and provide alerts for low battery levels.

### Class: Device

* **Attributes**:
  + name: Name of the device.
  + maxBatteryLife: Maximum battery life of the device.
  + powerConsumption: Power consumption rate of the device.
  + remainingBattery: Remaining battery life of the device.
* **Methods**:
  + Device(String name, int maxBatteryLife, int powerConsumption): Constructor to initialize device attributes.
  + usePower(int time): Simulates device usage by decrementing remainingBattery.
  + Getters for name, maxBatteryLife, powerConsumption, and remainingBattery.

### Class: PowerManager

* **Attributes**:
  + devices: List to store instances of Device.
* **Methods**:
  + PowerManager(): Constructor to initialize the list of devices.
  + addDevice(Device device): Adds a device to the list.
  + usePower(String deviceName, int time): Simulates power usage for a specific device.
  + getTelemetry(): Returns a copy of the list of devices.
  + alertLowBattery(int threshold): Returns a list of device names with remaining battery below the threshold.
  + compareDevices(String deviceName1, String deviceName2): Compares two devices based on battery life and power consumption.

### Class: Main

* **Functionality**:
  + Provides a command-line interface to interact with the power management system.
  + Allows users to add devices, simulate power usage, display telemetry data, alert low battery levels, and compare devices.

## Implementation Details

* **Device Initialization**:
  + Devices like WhatsApp, Instagram, and Messenger are initialized with specific attributes (name, max battery life, power consumption).
* **Simulation**:
  + Simulates power usage for each device.
  + Updates and displays telemetry data after usage simulation.
* **Alerts**:
  + Detects devices with battery levels below a specified threshold and alerts the user.
* **Comparison**:
  + Compares selected devices based on battery life and power consumption.

## Conclusion

The Java program effectively demonstrates a power management system for multiple devices, providing real-time telemetry data, low battery alerts, and device comparison capabilities. It serves as a practical tool for monitoring and optimizing power usage in various scenarios.