Problem Chosen	2026 MCM/ICM Summary Sheet	Team Control Number
A		2611750

River Shield: Mathematical Armor for Water Intake Protection

Summary

summary

Keywords:

keyword1; keyword2; keyword3

Contents

1 Introduction

1.1 Problem Background

Smartphones have been

1.2 Problem Restatement

In this problem, we're asked to build a

1.3 Our Work

2 Model Preparation

2.1 Assumptions and Justifications

2.2 Notations

We listed all the parameters and terminologied here. Our model has x parameters in all, involving n depending on the physical configuration and (x-n) tunable according to different scenarios.

We will restate related parameters before the modeling of each components.

2.3 Data Statement

2.4 Our Modeling Philosophy

Our core view is that software applications are not "true consumers" but "managers" of the hardwares which directly consume power. Therefore, building a precise mapping from software operations to hardware components'consumption is essential, worthing a specific "methodology".

Besides, we begin with comprehensive consideration of all possible influencing factors according to [5] and grdually shrinking it by merging, simplifying and justified ignoring.

We also seek to link our model to real-world insights, bridging theory with practice, with the ultimate goal of developing truly applicable forecasting capabilities.

3 Core Model Development

3.1 Fundamental Equation and Factor Selection

Our modeling framework treats smartphone hardware as parallel current-drawing components, yielding a governing differential equation for SOC(t). Software behavior influences battery drain exclusively by modulating hardware operating parameters—screen brightness, CPU utilization, network states, etc. We therefore develop modular sub-models that first translate software activities into hardware settings, then compute the resulting current consumption for each component.

3.1.1 Continuous-Time Battery Dynamics

Based on the definition of State of Charge and the physical meaning of electric current, we obtain the following system of equations:

$$\begin{cases} \text{SOC}(t) = \dfrac{Q(t)}{Q_{\text{max}}} \\ \dfrac{dQ}{dt} = -I_{\text{total}}(t) \end{cases} \tag{1}$$

where the first equation defines SOC as the ratio of remaining charge to maximum charge, and the second equation expresses charge conservation during discharge.

Therefore, by differentiating the first equation and substituting the second, we derive the governing differential equation:

$$\frac{d(\text{SOC})}{dt} = -\frac{I_{\text{total}}(t)}{C_{\text{eff}}(T, N_{\text{cycles}})} \tag{2}$$

where:

- $\text{SOC}(t) \in [0, 1]$ is the state of charge at time t (dimensionless)
- $I_{\text{total}}(t)$ is the total discharge current (mA) at time t
- $C_{\text{eff}}(T, N_{\text{cycles}})$ is the effective battery capacity (mAh)

To account for environmental and aging effects that reduce usable battery capacity, we introduce the effective capacity C_{eff}, which is related to the nominal capacity by:

$$C_{\text{eff}} = C_{\text{nominal}} \times f_{\text{temp}}(T) \times f_{\text{aging}}(N_{\text{cycles}}) \tag{3}$$

where:

- C_{nominal} is the manufacturer's rated capacity (e.g., 4000 mAh)
- $f_{\text{temp}}(T)$ is the temperature correction factor ($0.7 \leq f_{\text{temp}} \leq 1.0$)
- $f_{\text{aging}}(N_{\text{cycles}})$ is the aging correction factor based on charge-discharge cycles
- T is the battery temperature (°C)
- N_{cycles} is the number of complete charge-discharge cycles

The total discharge current $I_{\text{total}}(t)$ represents the sum of currents drawn by all active smartphone components:

$$I_{\text{total}}(t) = I_{\text{display}}(t) + I_{\text{processor}}(t) + I_{\text{memory}}(t) + I_{\text{network}}(t) + I_{\text{sensors}}(t) \tag{4}$$

Further decomposition and refined modelings are provided in the following subsections.

And each component current incorporates two distinct elements:

- **Baseline overhead**: The minimum power required for the component to remain operational, including essential operating system services and hardware idle states.
- **Software-driven consumption**: Additional power drawn when applications and user activities activate hardware features above their baseline levels.

Time-to-Empty Prediction Given initial conditions $\text{SOC}(t_0) = \text{SOC}_0$, the time until complete discharge ($\text{SOC} = 0$) is obtained by solving:

$$t_{\text{empty}} = \int_{\text{SOC}_0}^{0} \frac{C_{\text{eff}}}{I_{\text{total}}(\text{SOC}, t)} \, d\text{SOC} \tag{5}$$

In practice, this is computed numerically due to the time-dependent nature of $I_{\text{total}}(t)$.

This foundational model enables quantitative predictions of battery life under various usage scenarios. In the following chapter, we apply it to specific cases, analyze the results, and conduct comprehensive model validation.

3.1.2 Factor Selection and Simplification Strategy

Thanks to [5] , we're able to consider all kinds of

3.1.3 Environmental and Aging Factors

Lithium-ion battery capacity degrades with usage due to irreversible electrochemical changes during charge-discharge cycling. The aging correction factor f_{aging} depends on the accumulated cycle count N_{cycles}, where one cycle represents 100% cumulative depth-of-discharge.

3.2 Hardware Components Modeling

3.2.1 Display Power Model

While Carroll & Heiser's seminal work [1] established a linear relationship between display power and backlight intensity for LCDs, this model is inapplicable to modern AMOLED displays. The fundamental difference lies in AMOLED's pixel-independent emission, which replaces uniform backlighting. We therefore adopt a more sophisticated model tailored to AMOLED's characteristics [6], whose formulation accurately captures the power consumption patterns of contemporary screens.

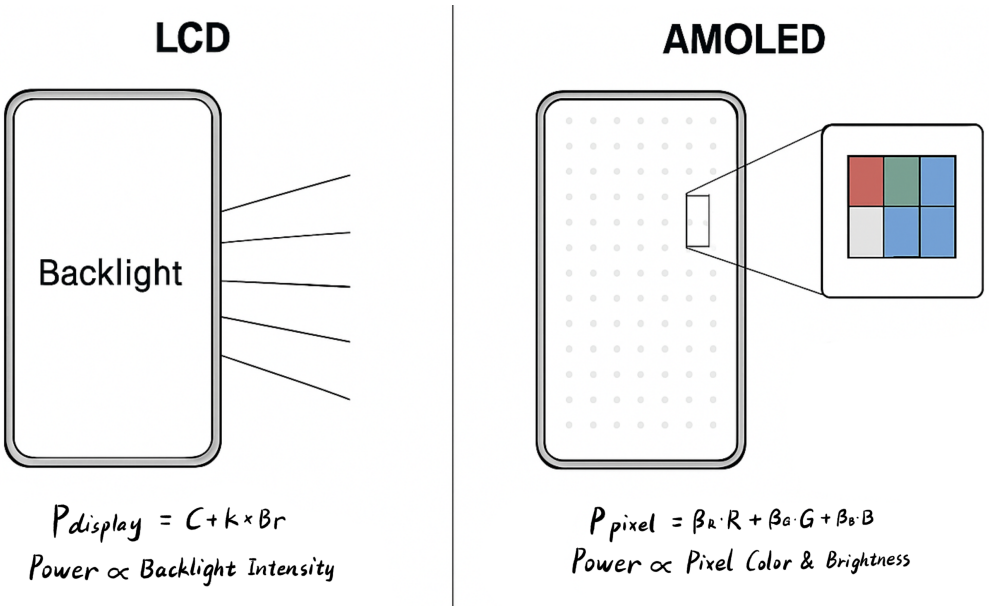

Figure 1: Schematic comparison of LCD and AMOLED power models.

The power consumption of a single AMOLED pixel is modeled as a function of its color intensity and the global brightness setting. The total display current I_{disp} is calculated by summing the contribution of all pixels and converting the total power to current:

$$I_{\text{disp}} = \frac{1}{V}\left[C + Br \cdot N \cdot (\beta_R R + \beta_G G + \beta_B B + a(R+G+B) + b)\right] \tag{6}$$

where $R, G, B \in [0, 1]$ are the normalized intensities of the red, green, and blue sub-pixels derived from $R_{avg}, G_{avg}, B_{avg}$. All other parameters are defined in Table 1.

Table 1: Parameters of the AMOLED Display Power Model

Symbol	Unit	Meaning
C	mW	Base power (black screen).
β_R, β_G, β_B	mW	Power coeff. per sub-pixel intensity.
a	mW	Linear coeff. for RGB sum correction.
b	mW	Constant for RGB sum correction.
R_{avg}, G_{avg}, B_{avg}	– (0-255)	Assumed average screen color.
N	–	Pixel count (10^6, ≈720p).
V	V	System voltage.
Br	– (0.0–1.0)	Global brightness factor.

Equation (6) decomposes the display power into a fixed base cost C and a dynamic component scaled by the global brightness Br. The term inside the parentheses, $(\beta_R R + \beta_G G + \beta_B B + a(R+G+B) + b)$, represents the power consumed by a **single pixel** at full brightness. The coefficients $\beta_R, \beta_G, \beta_B$ capture the efficiency of each sub-pixel, while the linear correction term $a(R+G+B)+b$, identified in [6], is crucial for accurately modeling power at high luminance levels (e.g., white backgrounds common in applications). In contrast to the runtime screen analysis performed in the original study, **we simplify the model by assuming the screen displays a static, average color** $(R_{avg}, G_{avg}, B_{avg})$. This simplification allows for efficient system-level energy estimation without the need for real-time frame-buffer sampling, making it suitable for our integrated power model.

This model provides a physically-grounded method to estimate the AMOLED display's current draw as a function of user-defined brightness. Having established the display power model, we now turn to the processing units—the CPU and GPU—whose power consumption is governed by dynamic voltage and frequency scaling (DVFS).

3.2.2 Processor Power Model (CPU + GPU)

3.2.3 Memory and Storage Model

3.2.4 Sensor Power Model (GPS, Camera, etc.)

3.3 Signaling Modules Modeling

3.3.1 Cellular and Wi-Fi Communication

3.3.2 Bluetooth and Short-Range Communication

Bluetooth Module The power consumption modeling of Bluetooth modules is fundamentally based on the core physical principles of radio frequency (RF) communication: First is the duty

cycle principle. Bluetooth operates in the 2.4 GHz ISM unlicensed band, and the activation time ratio of the RF circuit directly determines power consumption: synchronous packets are sent at high frequency during continuous data transmission (duty cycle ≈ 1, high power consumption), while only intermittent scanning of surrounding devices is performed in idle standby (duty cycle ≈ 0.05, significantly lower power consumption). Second is the power-current linear relationship: different Bluetooth Class levels correspond to fixed maximum transmit powers (e.g., $100\,\mathrm{mW}$ for Class 1, $2.5\,\mathrm{mW}$ for Class 2). With the supply voltage of mobile phone Bluetooth modules fixed at $3.7\,\mathrm{V}$, the basic physical formula $P = U \cdot I$ is satisfied, so the Class level directly determines the baseline current of the Bluetooth module. As the default mode of mainstream mobile phones, Bluetooth Low Energy (BLE) is designed based on the above physical principles: by shortening RF activation time and reducing transmit power (corresponding to Class 4 with a transmit power of only $0.5\,\mathrm{mW}$), its static power consumption is reduced to 1/10 of that of classic Bluetooth.

Based on the above physical background, we construct a continuous-time Bluetooth current model, using an indicator function to dynamically distinguish between BLE and classic Bluetooth modes:

$$I_{\text{bluetooth}}(t) = I_{\text{ble}}(t) \cdot \mathbb{I}(V_{\text{ble}}(t) = 1) + I_{\text{classic}}(t) \cdot \mathbb{I}(V_{\text{ble}}(t) = 0)$$

where the indicator function $\mathbb{I}(\cdot)$ denotes mode switching: $V_{\text{ble}}(t) = 1$ for BLE mode, and $V_{\text{ble}}(t) = 0$ for classic Bluetooth mode.

For BLE mode (default for mainstream mobile phones), the current is the linear superposition of idle scanning current and dynamic transmission current:

$$I_{\text{ble}}(t) = I_{\text{ble\_idle}} + D(t) \cdot (I_{\text{ble\_tx}} - I_{\text{ble\_idle}})$$

For classic Bluetooth mode (compatible with legacy devices), the current is superimposed by Class-level static current and dynamic transmission current:

$$I_{\text{classic}}(t) = I_{\text{class}}(C(t)) + D(t) \cdot (I_{\text{classic\_tx}} - I_{\text{class}}(C(t)))$$

Hotspot Module The power consumption modeling of the hotspot module is based on the underlying physical mechanisms of dual-role communication superposition and radio frequency (RF) transmission: when the hotspot is active, the mobile phone undertakes two communication roles simultaneously — it acts as a cellular network client (maintaining connection with the base station to receive downlink data, with power consumption described by the cellular network model $I_{\text{cellular}}(t)$ mentioned earlier), and as a Wi-Fi Access Point (AP) (transmitting Wi-Fi signals for peripheral device access, generating independent AP RF power consumption). The communication power consumption of these two roles is linearly superimposed, forming the core power consumption source of the hotspot.

The RF power consumption of a Wi-Fi AP follows clear physical laws: it is positively correlated with transmit power and the number of connected devices. Transmit power is determined by the working frequency band — the 5 GHz band has larger path loss and requires higher transmit power ($20\,\mathrm{dBm}$ vs $17\,\mathrm{dBm}$ for 2.4 GHz), resulting in higher RF amplifier current. When multiple devices are connected, communication resources need to be scheduled in time division, leading to an increased activation duty cycle of the RF circuit; experimental data shows that the AP power consumption increases by about 30% for each additional connected device.

Table 2: Parameters of Bluetooth Power Consumption Model

Parameter	Unit	Description
$V_{\text{ble}}(t)$	—	Bluetooth mode at time t (1=BLE, 0=classic)
$C(t)$	—	Classic Bluetooth Class level (dynamically negotiated)
$D(t)$	—	Duty cycle (0 1, reflects user behavior)
$I_{\text{ble\_idle}}$	mA	BLE idle current (measured: 1.2)
$I_{\text{ble\_tx}}$	mA	BLE transmission current (measured: 8.0)
$I_{\text{class}}(C(t))$	mA	Classic Bluetooth static current (30.0/8.0/2.0 for Class 1/2/3)
$I_{\text{classic\_tx}}$	mA	Classic Bluetooth transmission current (measured: 25.0)
U	V	Supply voltage (constant: 3.7)

In addition, the mobile phone needs to run routing protocols (e.g., DHCP, NAT) when acting as an AP, and the processor must continuously forward network traffic. This additional power consumption has been included in the processor current $I_{\text{processor}}(t)$, so this model only focuses on RF power consumption related to communication.

Based on these physical characteristics, we construct a continuous-time hotspot current model, where the total current is the linear superposition of cellular network current, AP RF dynamic current, and AP static current:

$$I_{\text{hotspot}}(t) = I_{\text{cellular}}(t) + I_{ap}(F_{\text{wifi}}(t), N(t)) + I_{ap\_static}$$

where: - $I_{\text{cellular}}(t)$: Cellular network current, directly adopted from the previous cellular network model, reflecting the basic power consumption of the hotspot to maintain cellular connection; - $I_{ap}(F_{\text{wifi}}(t), N(t)) = k_{ap}(F_{\text{wifi}}(t)) \cdot (1 + 0.3 \cdot N(t))$: AP RF dynamic current, determined by the frequency band transmit current coefficient and the number of connected devices; - $I_{ap\_static} = 20\,\text{mA}$: AP static current, the basic power consumption of the Wi-Fi AP baseband circuit, independent of the number of connected devices.

3.3.3 Special Note: GPS as Positioning Sensor

4 Software-to-Hardware Power Mapping and Calibration

4.1 Software Consumption Analysis

4.1.1 Theoretical Framework

Software does not directly consume power; it drives hardware, which then consumes power. We add two new coefficients to the hardware-based model:

$$\frac{dQ(t)}{dt} = -\sum_{i=1}^{N} \underbrace{I_i(\theta_i)}_{\text{hardware}} \times \underbrace{A_i(t)}_{\text{when to use}} \times \underbrace{\eta_i}_{\text{how to use}}$$

Table 3: Parameters of Hotspot Power Consumption Model

Parameter	Unit	Physical Definition
$F_{\text{wifi}}(t)$	—	Wi-Fi band of hotspot at time t (2.4GHz/5GHz), determined by user settings
$N(t)$	—	Number of connected devices at time t, dynamically changing with scenarios (e.g., 1 for commute, 3 for home)
$k_{ap}(F_{\text{wifi}}(t))$	mA	AP transmit current coefficient by band ($45\,\text{mA}$ for 2.4GHz, $75\,\text{mA}$ for 5GHz)
0.3	—	Device count influence coefficient, reflecting power consumption increment from multi-device scheduling
I_{ap_static}	mA	AP static current ($20\,\text{mA}$), basic power consumption of Wi-Fi AP baseband circuit

- $I_i(\theta_i)$: hardware current model (from Chapter 2)
- $A_i(t)$: activation level (0=off, 1=full load)
- η_i: software efficiency factor ($\eta = 1$ = best, higher = worse)

Example: same network chip can be used efficiently ($\eta = 1.2$, big downloads) or inefficiently ($\eta = 2.5$, many small HTTP requests).

4.1.2 Literature-Based Key Factors

From papers, we know:

Scenario	**Percentage**	**Key Hardware**
Idle state	61%	Screen (45%), Background network (16%)
Active state (no screen)	39%	Network (70%), CPU (15%), GPS (10%)
API calls share	85%*	Network APIs (60%), Sensor APIs (25%)

Table 4: Power distribution in smartphones (summarized from literature)

*Percentage of active non-screen consumption.

Takeaway: Focus on screen and network modules. Their η values need careful calibration.

4.1.3 Application Category Selection

We pick five app types that are:

1. Simple (no complex games)

2. Similar within category (low variance)
3. Cover different hardware usage patterns

Category	Examples	Key Hardware Focus
Video streaming	TikTok, YouTube	Screen, Network, Decoder
Social media	WeChat, Weibo	Network (small HTTP), Screen
Navigation	Gaode Maps	GPS, Network, Screen
Web browsing	Chrome	Network, CPU (rendering)
Music	Spotify	Audio CPU, Background network

Table 5: Five selected app categories and their hardware focus

These five cover most daily usage scenarios while keeping analysis manageable.

4.2 App-Based Parameter Calibration

4.2.1 New Parameter Definitions

Where do $A_i(t)$ and η_i come from?

1. Activation Level $A_i(t)$:

- From app behavior logs (e.g., Android Profiler)
- Binary or continuous: 0/1 for on/off, or 0-1 for partial usage
- Example: For video app, $A_{\text{screen}}(t) = 1$ when playing, 0 when paused

2. Efficiency Factor η_i:

- From protocol analysis: HTTP vs. WebSocket, big files vs. small requests
- From literature measurements
- Network: $\eta_{\text{network}} = 1.2$ (video streaming) to 2.5 (social media)
- Screen: $\eta_{\text{screen}} = 1.0$ (dark mode) to 1.5 (bright UI)

4.2.2 Calibration for Five App Categories

We calibrate parameters for each category:

How we got these numbers:

1. Literature values for similar apps
2. Common-sense estimates (e.g., HTTP is inefficient)
3. Will be refined in future work with real measurements

App Type	η_{network}	η_{screen}	Typical A_i Pattern
Video streaming	1.2	1.3	Screen: always on, Network: bursts
Social media	2.5	1.8	Screen: on/off, Network: frequent small
Navigation	1.5	1.5	GPS: always on, Screen: mostly on
Web browsing	1.8	1.2	Network: mixed, Screen: reading mode
Music	1.3	N/A	Screen: mostly off, Network: occasional

Table 6: Calibrated parameters for five app categories

4.2.3 Implementation in the Model

To use in code:

```
# Example: social media app for 1 minute
for t in range(60):  # seconds
    # Screen: on for 45s, off for 15s
    A_screen = 1.0 if t < 45 else 0.0

    # Network: every 10s a small HTTP request
    if t % 10 == 0:
        I_network = network_model(...) * 2.5  # eta=2.5
    else:
        I_network = 0

    total_current = I_screen*A_screen*1.8 + I_network + ...
```

This completes the software-aware power model. Next chapter will combine these apps into full usage scenarios.

4.3

4.4

4.5

5 Scenario-Based Analysis and Prediction

6 Model Analysis and Evaluation

[4] [3]

[7] [2] [8]

References

[1] Aaron Carroll and Gernot Heiser. “An analysis of power consumption in a smartphone”. In: *2010 USENIX Annual Technical Conference (USENIX ATC 10)*. 2010.

[2] Chien Aun Chan et al. “Assessing network energy consumption of mobile applications”. In: *IEEE Communications Magazine* 53.11 (2015), pp. 182–191.

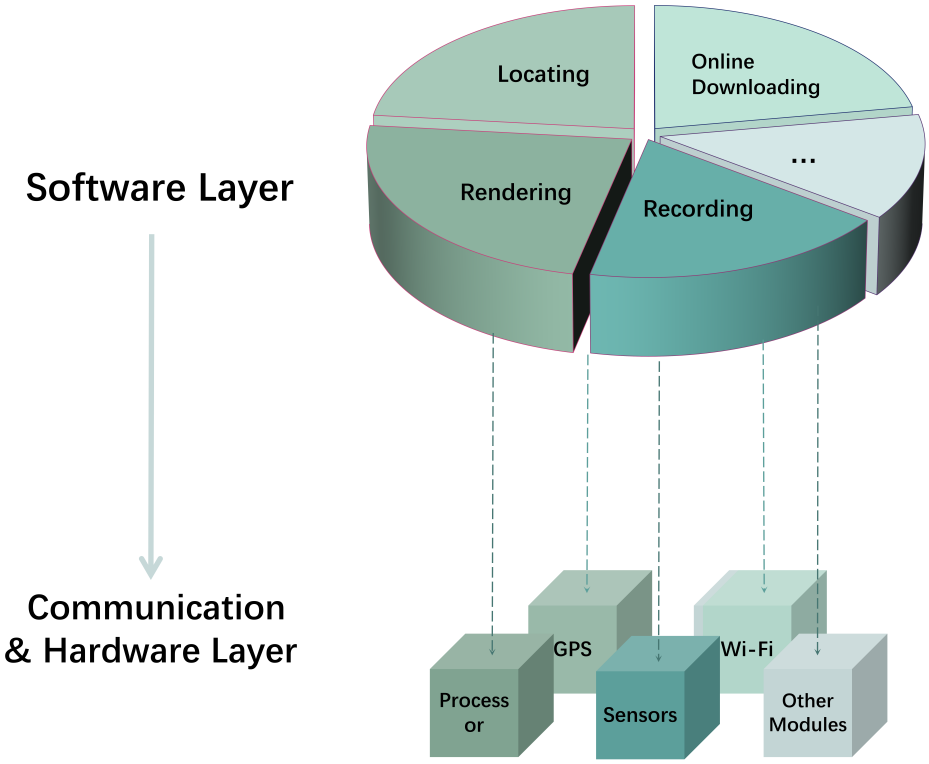

Figure 2: Software behavior to hardware component mapping diagram.

[3] Luis Cruz and Rui Abreu. "Catalog of energy patterns for mobile applications". In: *Empirical software engineering* 24.4 (2019), pp. 2209–2235.

[4] Dario Di Nucci et al. "Software-based energy profiling of android apps: Simple, efficient and reliable?" In: *2017 IEEE 24th international conference on software analysis, evolution and reengineering (SANER)*. IEEE. 2017, pp. 103–114.

[5] Kerry Hinton et al. "Power consumption and energy efficiency in the internet". In: *IEEE Network* 25.2 (2011), pp. 6–12.

[6] Russ Joseph and Margaret Martonosi. "Run-time power estimation in high performance microprocessors". In: *Proceedings of the 2001 international symposium on Low power electronics and design*. 2001, pp. 135–140.

[7] Ding Li et al. "An empirical study of the energy consumption of android applications". In: *2014 IEEE International Conference on Software Maintenance and Evolution*. IEEE. 2014, pp. 121–130.

[8] Abhinav Pathak, Y Charlie Hu, and Ming Zhang. "Where is the energy spent inside my app? Fine Grained Energy Accounting on Smartphones with Eprof". In: *Proceedings of the 7th ACM european conference on Computer Systems*. 2012, pp. 29–42.

7 AI Use Report

Project Title: The River Intake Shield: Optimized Sensor Deployment and Emergency Response

Team Control Number: [2611750]