Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones

Presented By: Harshang Patel, Daniel Garan, Alex Willis

University of Michigan

September 27, 2022

Outline I

Outline II

Introduction

Motivation

- Tradeoff: Power Hungry Smartphone Application Features vs Low Power Consumption for Longer Battery Life.
- Power Saving Feature: Dynamically adjust power consumption for required performance.
- Bad software design results in power hungry applications which results in short battery lifespan.
- Problem can be solved by understanding power models for portable embedded systems and using it to influence software design decisions.

Paper Contribution(s)

- PowerBooter: Power model generation technique to provide accurate, real-time power consumption estimates for components like CPU, LCD, GPS, audio, Wi-Fi and cellular communication components.
- PowerTutor: On-line power estimation tool to determine system-level power consumption.
- Battery-based automatic power model construction technique using built-in battery voltage sensors and complex automated characterization procedure.

Paper Contribution(s)

- Manually generated power models for HTC Dream and HTC Magic phones.
  - *First time a GPS power model has been described.*
- Measure variation in power consumption properties among phones.
- A novel automated power model construction technique.
  - Uses in-built battery voltage sensors and knowledge of battery discharge behavior to monitor power consumption.
  - *Requires no external measurement equipment!*

Related work

Prior Work: Online Power Modeling and Model Construction Techniques

- Requires deep knowledge between functional units and their power consumption [3, 4]. Activity measured using hardware performance counters.
- Requires no knowledge on hardware component implementation [5, 6]. Assumes linear relationship between processor power consumption and hardware performance counters.
- Workstation power modeling technique that models CPU power consumption only [7]. Incomplete solution for embedded system power estimation.

Prior Work: Online Power Modeling and Model Construction Techniques

- Full-system power model for Palm PDAs [1] and System-level power model for Android platform smartphones [2].
  - Correlates OS visible state variables with power consumption while running normal software applications.
  - Drawback: Model Accuracy relies on component activity and power management states reached using training applications.
- Other battery based behavior models [8, 9] requires knowledge of discharging current and remaining battery capacity, which is not available for most smartphones.
- *Our technique* requires knowledge of battery discharge voltage curve and access to battery voltage sensor, which is available on most smartphones.

Power Model

Smartphone Hardware Components

- Power modeling of an Android Dev Phone 1 (ADP1), a version of HTC Dream phone which permits superuser access.
- Android 1.6 software development kit, which supports Java and C program development.

Hardware component	Detailed description
Processor	MSM7201A chipset, including ARM11 application processor, ARM9 modem, and high-performance DSP
LCD Display	TFT-LCD flat glass touch-sensitive HVGA screen
Wi-Fi interface	Texas Instruments WL 1251B chipset
GPS	A-GPS and standalone GPS
Cellular	Qualcomm RTR6285 chipset, supporting GSM, GPRS/EDGE, Dual band UMTS Bands I and IV, and HSDPA/HSUPA
Bluetooth	Bluetooth 2.0+EDR via Texas Instruments BRF6300
Audio	Built-in microphone and speaker
Camera	3.2-megapixel camera
Battery	Rechargeable lithium-ion battery with capacity: 1,150 mAh
Storage	microSD card slot

Figure: Hardware Components for HTC Dream

Experimental Setup

- Monsoon FTA22D meter for power measurement.
  - It supplies stable voltage to the phone and samples the power consumption at a rate of 5 kHz.
- ADP1 runs two programs during characterization.
  - First program exercises CPU utilization and LCD brightness.
  - Second program logs readings at high frequency to capture most changes in variables indicating the state change.

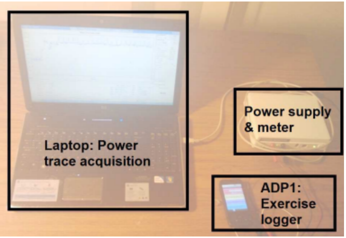

Figure: Experimental setup for power measurement

Selecting Hardware Components

1. Hold the power and activity states of all other components constant.
2. Vary the activity state to extreme values for the components of interest.
   - Set CPU utilization to its lowest and highest values.
   - Configure GPS state to extreme values by controlling activity and visibility of GPS satellites.

Selecting System Variables

- Experimentally determine the setting that results in extreme power consumption for a component.
- Ignore the components with insignificant impact on system power consumption (e.g. SD card).
- Components modeled: CPU, LCD, GPS, Wi-Fi, Cellular, and Audio interfaces.

Insights: Selecting System Variables

- Assuming that individual components are independent, the maximum error is 6.27%.
- Measured using power consumption of the phone at different cross products of extreme power states.
  - LCD/CPU cross products: [Full brightness, Low CPU] and [Low brightness, High CPU]
- Therefore, sum of independent component-specific power estimates is sufficient to estimate system power consumption.

Main Idea

- For each component, need to determine relationship between each state variable and power consumption.
- Set of training programs to change one activity state variable at a time, while keeping all others constant.
  - Periodically vary each state variable over its full range.
- Fix power states of all other components when exercising one component to reduce measurement noise resulting from state transitions by the other components.

Main Idea: Example

- Determine relationship between CPU utilization and power consumption.
- Fix CPU frequency and disable LCD, cellular, Wi-Fi, and GPS interfaces.
- Use a program to gradually vary the CPU utilization from 0% to 100%.
- **Note:** Some component power state variables cannot be controlled independent (i.e CPU and Wi-Fi).
- To account for this, monitor all component power states while exercising the target component and use that during regression.

CPU

- Power consumption dependent on utilization and frequency-voltage settings.
- HTC Dream platform supports two frequencies: 385 MHz and 246 MHz.
- Only consider application processor (ARM11).
- Training program consists of:
  - CPU use controller: controls the duty cycle of computation intensive task.
  - Frequency controller: writes the system frequency file in the */sys* filesystem.

CPU Power Model

- β_{CPU} shows the power difference between active and idle states of the application processor.
- CPU Model Equation:
  $(\beta_{uh} \times freq_h + \beta_{ul} \times freq_l) \times util + \beta_{CPU} \times CPU_{on}$

Category	System variable	Range	Power coefficient
CPU	util	1-100	β_{uh} : 4.34
			β_{ul} : 3.42
	$freq_l$, $freq_h$	0,1	n.a.
	CPU_{on}	0,1	β_{CPU} : 121.46

Table: HTC Dream CPU Power Model

LCD Power Model

- Power model derived using a training program that turns the LCD on and off and changes its brightness.
- Used 10 uniformly distributed brightness levels to simplify modeling.
- LCD Model Equation: $\beta_{br} \times brightness$

Category	System variable	Range	Power coefficient
LCD	brightness	0-255	β_{br} : 2.40

Table: HTC Dream LCD Power Model

GPS

- Power consumption dependent on mode (active, sleep, off), number of satellites detected, and signal strength of each satellite.
- All variables logged using Android Software Development Kit API.
- GPS state is changed between sleep and active using *requestLocationUpdate* method [12].
- Change physical environment to control the number of satellites available and their signal strength.
    - Use conductive hemisphere to exercise course-grained control over GPS environment.

GPS States

- Three states considered:
  - Active with many satellites available.
  - Active with few satellites available.
  - Sleep.
- Measurement shows power consumption strongly depends on GPS state (active/sleep) but little on number of satellites available and signal strength.

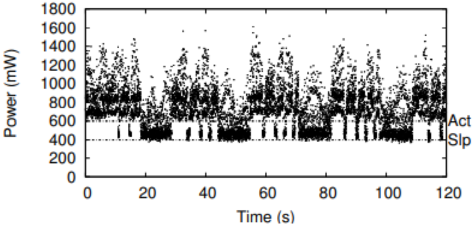

Figure: Power profile for the current GPS policy

GPS Power Model

- GPS Model Equation: $\beta_{Gon} \times GPS_{on} + \beta_{Gsl} \times GPS_{sl}$

Category	System variable	Range	Power coefficient
GPS	GPS_{on}	0,1	β_{Gon} : 429.55
	GPS_{sl}	0,1	β_{Gsl} : 173.55

Table: HTC Dream GPS Power Model

Wi-Fi

- Monitor two network parameters: Data rate and Channel rate.
- 1 KB TCP[1]/UDP[2]packets exchanged between the smartphone and local server.
- Channel data rate by varying delay between transmission from 0s to 2s in steps of 0.1s.
- Uplink channel rate used: 11 Mbps, 36 Mbps, 48 Mbps, and 54 Mbps.

[1]TCP: Transmission Control Protocol
[2]UDP: User Datagram Protocol

Wi-Fi

- Power model depends on:
  - Number of packets transmitted and received per second (*npackets*).
  - Uplink channel rate ($R_{channel}$).
  - Uplink data rate (R_{data}).
- Wi-Fi interface has 4 power states: *low-power*, *high-power*, *ltransmit*, and *htransmit*.
- Network card briefly enters *ltransmit* and *htransmit* states when transmitting data, then returns to previous state.

Wi-Fi States

Packet rate decides power state not bit rate.

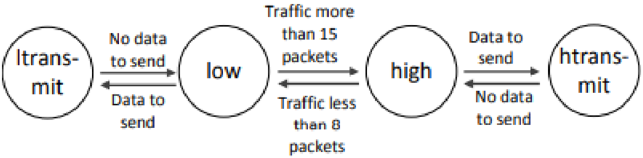

Figure: Wi-Fi Interface Power States

Wi-Fi States

- *ltransmit* state:
  - Time: $<$ 10-15 ms
  - Power consumed: 1000 mW
- *low-power* state:
  - Less than 8 packets are transmitted or received per second.
  - Power consumed: 20 mW
- *high-power* state:
  - More than 15 packets are transmitted or received per second.
  - Power consumed: 710 mW
- *htransmit* state:
  - Time: 10-15 ms
  - Power consumed: 1000 mW

Wi-Fi States

- For a given Channel and Packet rates, power consumption is independent of packet size.
- But for low channel rate, more time is spent in the *htransmit* state to transmit same amount of data.
- Power consumption for *high-power* state is modeled as:

$$\beta_{Wi-Fi\_h} = 710mW + \beta_{cr}(R_{channel}) \times R_{data} \tag{1}$$

$$\beta_{cr}(R_{channel}) = 48 - 0.768 \times R_{channel} \tag{2}$$

Wi-Fi Power Model

- Wi-Fi Model Equation:
  $\beta_{Wi-Fi\_l} \times Wi-Fi\_l + \beta_{Wi-Fi\_h} \times Wi-Fi\_h$

Category	System variable	Range	Power coefficient
Wi-Fi	$npackets, R_{data}$	$0-\infty$	n.a.
	$R_{channel}$	1-54	β_{cr}
	$Wi-Fi\_l$	0,1	β_{Wi-Fi_l} : 20
	$Wi-Fi\_h$	0,1	$\beta_{Wi-Fi\_h} : Equation(1)$

Table: HTC Dream Wi-Fi Power Model

Cellular

- Send TCP/UDP packets between a smartphone and a local server via the T-Mobile UMTS 3G network.
- Vary packet size from 10 B to 1 KB.
- For each packet size, vary transmission delay from 0 s to 12 s in 0.1 s intervals.
- This model does not consider signal strength. Focus of future work.

Cellular States

- The model depends on transmit and receive data rate (*data_rate*) and two queue sizes (*downlink_queue* and *uplink_queue*).
- Three states to communicate between base station and cellular interface.

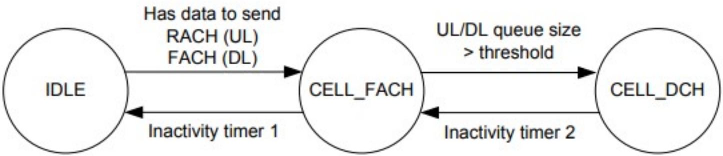

Figure: 3G Interface Power States

Cellular States

- *CELL_DCH* state:
  - Cellular interface has dedicated channel to base station.
  - Uses high-speed downlink/uplink packet access data rates.
  - Power consumed: 570 mW
  - After fixed period of time enters *CELL_FACH* state.
- *CELL_FACH* state:
  - Uses random/forward access (RACH/FACH) common channel.
  - Data rate is few hundred bytes per second.
  - Power consumed: 401 mW
  - Enters *CELL_DCH* state to transmit lot of data.
  - After fixed period of time enters *IDLE* state.
- *IDLE* state:
  - Only receives paging messages and does not transmit data.
  - Power consumed: 10 mW

Cellular State Change Due to Inactivity

- Repeatedly download an 80 KB file using HTTP 30 times with a period that increases from 1-29 seconds in one second interval and record timestamp for each packet.
- Repeat experiment 3 times.
- Calculate two Round Trip Times (RTTs) before beginning each download.
    1. Time between sending SYN packet and receiving SYN-ACK packet during TCP connection set up.
    2. Time between sending HTTP-Get request and receiving the first data packet.

Cellular: First RTTs

- The sum of two inactivity timers is 10 seconds.
- State demotion from *CELL_DCH* to *CELL_FACH* causes large RTTs for download 7 and 8.

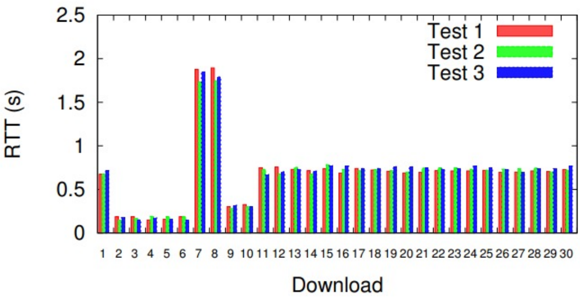

Figure: TCP handshake RTT

Cellular: Second RTTs

- Delay in state promotion from *CELL_FACH* to *CELL_DCH* causes large RTTs for downloads 1 and 7-30.
- Inactivity timer 1 is 6 seconds and Inactivity timer 2 is 4 seconds.

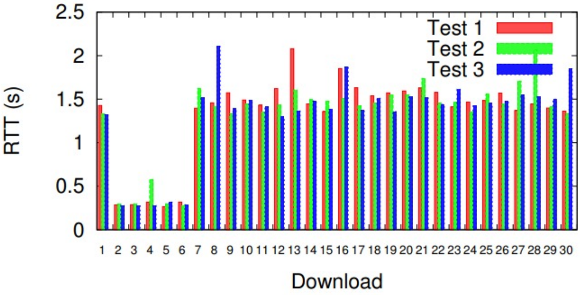

Figure: HTTP GET RTT

Cellular Power Model

- Cellular Model Equation:
  $\beta_{3G\_idle} \times 3G_{idle} + \beta_{3G\_FACH} \times 3G_{FACH} + \beta_{3G\_DCH} \times 3G_{DCH}$

Category	System variable	Range	Power coefficient
Cellular	*data_rate*	$0 - \infty$	n.a.
	downlink_queue	$0 - \infty$	n.a.
	uplink_queue	$0 - \infty$	n.a.
	$3G_{idle}$	0,1	β_{3G_idle} : 10
	$3G_{FACH}$	0,1	β_{3G_FACH} : 401
	$3G_{DCH}$	0,1	β_{3G_DCH} : 570

Table: HTC Dream Cellular Power Model

Audio

- Measure power consumption when audio interface is not in use, and when an audio file is played at different volumes.
- Power consumption dependent on audio interface but not speaker volume.
- Hypothesize that activating a digital signal processor (DSP) and/or speaker amplifier causes increased power consumption during audio output.

Audio Power Model

- Audio Model Equation: $\beta_{audio} \times Audio_{on}$

Category	System variable	Range	Power coefficient
Audio	$Audio_{on}$	0,1	β_{audio} : 384.62

Table: HTC Dream Audio Power Model

Regression-Based Approach

- Use multi-variable regression to minimize the sum of squared errors for the power coefficient.

$$\begin{pmatrix} P_0 \\ P_1 \\ \ldots \\ P_n \end{pmatrix} = \beta_1 \cdot \begin{pmatrix} U_{01} \\ U_{11} \\ \ldots \\ U_{n1} \end{pmatrix} \ldots + \beta_m \cdot \begin{pmatrix} U_{0m} \\ U_{1m} \\ \ldots \\ U_{nm} \end{pmatrix} + c \qquad (3)$$

- U_{ij} represents system variable i in the jth state.
- P_j is power consumption when all system variables are in the jth state.
- Regression input is system variables and outputs are power consumption and power coefficients β_i.
- Constant c is the minimum system power consumption.

Intra-And Inter-Phone Power Consumption Variation

Intra- and Inter-Phone Power Consumption Variation

- How does power consumption change for different devices?
  - Different models
  - Different instances of the same model
- Is it necessary to characterize each device, or can results from one device be used for other devices?

Intra-Phone Variation

- Small variation between multiple instances of the same phone
  - Max $\sim$ 10%
- 2x ADP1 (HTC Dream)
- 4x ADP2 (HTC Magic)

Variation (%)		Intra-ADP1	Intra-ADP2
CPU	β_{uh}	1.46	9.6
	β_{CPU}	9.05	9.20
LCD	β_{br}	1.56	2.5
Wi-Fi	$\beta_{Wi\text{-}Fi_h}$	1.31	3.55
	$\beta_{Wi\text{-}Fi_l}$	4.89	4.86
Cell	$3G_{DCH}$	1.03	1.73
	$3G_{FCH}$	2.80	2.94
GPS	GPS_{on}	1.35	3.01
	GPS_{sl}	2.48	3.82
Audio	β_{audio}	3.31	2.57

Figure: Variation of Power Models Among Same Phones

Inter-Phone Variation

- Significant differences between ADP1 and ADP2
  - Up to 62%
- Same processor and LCD specifications, different cellular chipsets

Variation (%)		Inter-type
CPU	β_{uh}	-23.16
	β_{CPU}	33.28
LCD	β_{br}	-28.13
Wi-Fi	$\beta_{Wi\text{-}Fi_h}$	2.86
	$\beta_{Wi\text{-}Fi_l}$	-31
Cell	$3G_{DCH}$	62.01
	$3G_{FCH}$	27.42
GPS	GPS_{on}	-5.12
	GPS_{sl}	-11.50
Audio	β_{audio}	-59.37

Figure: Variation of Power Models Among Different Phones

Battery State Based Automated Power Model Generation

Battery State Based Automated Power Model Generation

- Significant inter-phone variation means each phone model must be separately characterized
- Characterizing with test equipment is time-consuming and requires specialized tools
- Battery state based characterization only requires onboard voltage sensor, already present in smartphones
  - Current sensors available in some phones, but not universal

Simplified Lithium-Ion Battery Model

- Smartphone battery can be modeled as voltage source with series internal resistance

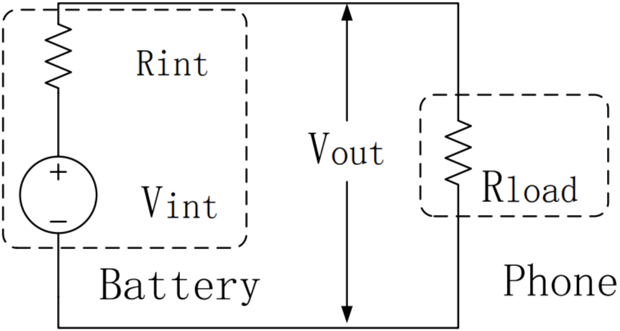

Figure: Equivalent Circuit for Battery

Battery State of Discharge Curve

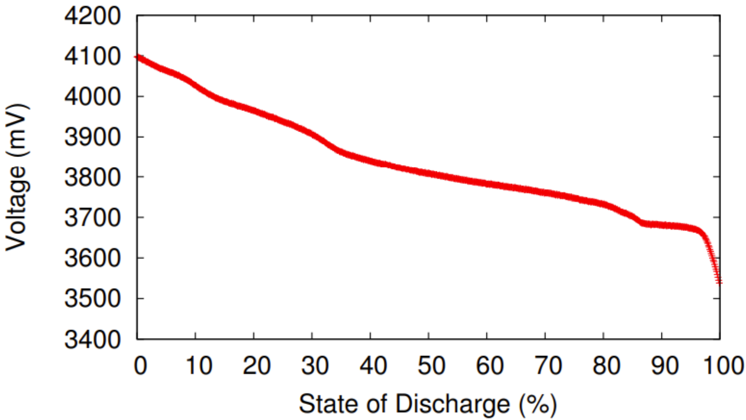

Figure: Discharge curve of ADP2 lithium-ion battery

Battery State of Discharge (SOD) Calculation

$$P * (t_1 - t_2) = E * (SOD(V_1) - SOD(V_2))$$

- t_1: Time at start of test
- t_2: Time at end of test
- V_1: Battery voltage at start of test
- V_2: Battery voltage at end of test
- P: Calculated power draw
- SOD(V): Piecewise linear function to model SOD over voltage, using linear regression to fit measured points
- E: Total battery energy capacity

Finding Battery Energy Capacity

- Can be read directly for new battery
- For old battery, capacity must be estimated using known power consumption mode, such as max CPU usage
- Either known capacity or known draw is needed to calculate absolute power consumption
  - Otherwise only relative result is obtained

Challenges with SOD Calculation Approach

- Discharge curve varies for different batteries, even of the same design
  - Capacity and discharge characteristics vary with design and age
  - Must characterize each device separately
- Discharge curve varies over temperature
  - Recommend characterizing at room temperature (73-78 F)
- Discharge current affects measured voltage due to R_{int}, and R_{int} varies with SOD
  - Reduce current to minimum state for pre-test and post-test voltage measurements

Power Model Validation

Accuracy Analysis for the Meter-Based Power Model

- Power model validated on 6 popular apps
  - Break the Block
  - Google Talk
  - Google Maps
  - The Weather Channel
  - YouTube
  - Browser
- $abs\ avg = \left|\frac{measured - predicted}{measured}\right|$
- $avg = \frac{measured - predicted}{measured}$

Modeled & Measured Power Consumption for Two Apps

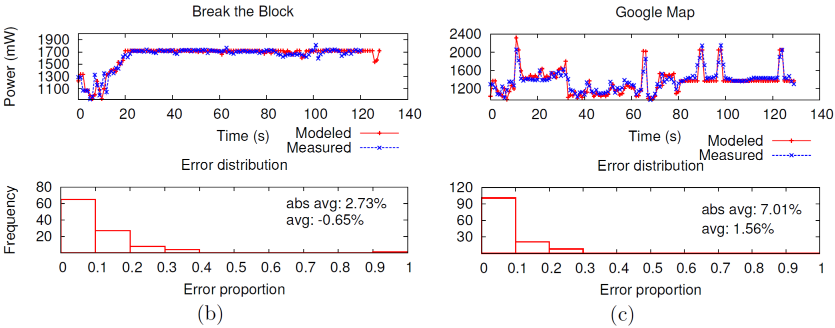

Figure: Power profiles for two selected applications

Results

- Long-term error *avg* $< 2.5\%$
- Average error *abs avg* $< 10\%$
- Power consumption of power estimation technique $= 80$mW

Constructing the Battery Discharge Curve

1. Obtain the discharge curve for each component
2. Determine the power consumption for each component in each state
3. Use regression to create the model

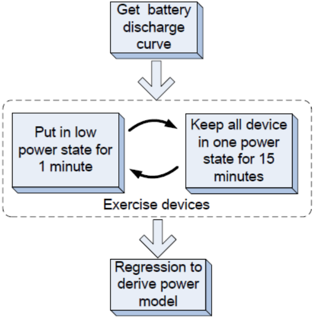

Figure: Battery SOD based power model construction

Selecting Discharge Interval

- Mean of the errors for all intervals deviates by no more than 0.4%
  - *Battery-based model is as accurate as meter-based model*
- Variance decreases with longer intervals
  - Over 92% of trials have $<$ 10% error for 45 minute intervals

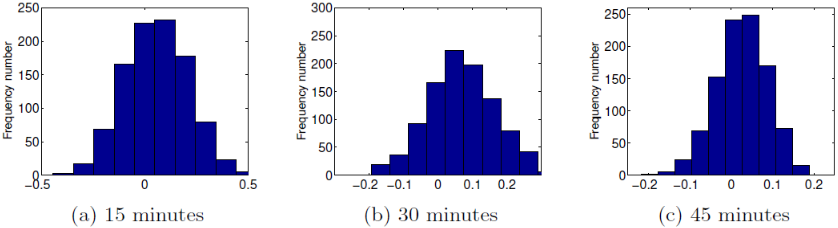

Figure: Error distribution for LCD

Accuracy Analysis for the Battery-Based Power Model

- Error distribution based on 15 minute discharge interval.
- The line in the middle of the box is the mean of all errors.
- The box boundaries indicate the 25th and 75th percentiles.
- The line span indicates maximum positive and negative errors.

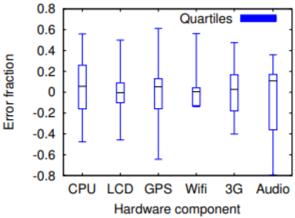

Figure: Error distributions for components

Power Estimation Tool

PowerTutor

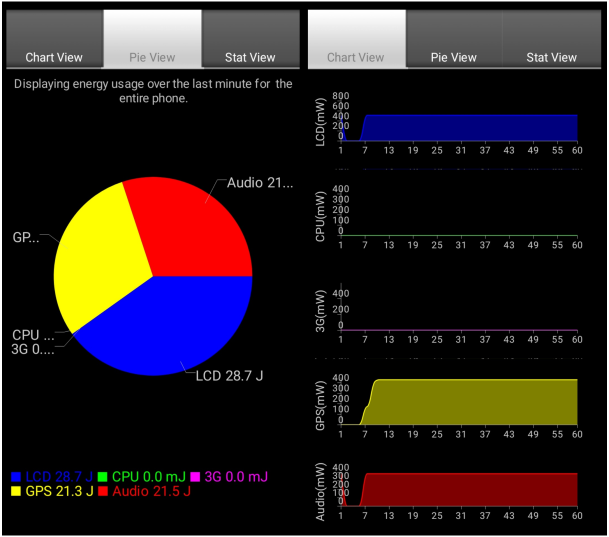
Chart View
Pie View
Stat View
Displaying energy usage over the last minute for the entire phone.
Audio 21...
GP...
CPU ...
3G 0....
LCD 28.7 J
LCD 28.7 J
CPU 0.0 mJ
3G 0.0 mJ
GPS 21.3 J
Audio 21.5 J
Chart View
Pie View
Stat View
LCD(mW)
CPU(mW)
3G(mW)
GPS(mW)
Audio(mW)

Purpose & Impact

- Developers can see the impact of software design decisions
- Users can compare apps and make better informed decisions

Conclusion

Conclusion

- PowerTutor: Power estimation and model generation
  - Includes six components: CPU, LCD, GPS, Wi-Fi, Audio, and Cellular
  - Accurate to within 0.8% on average for 10-second intervals
  - Released publicly
- PowerBooter: Automatic battery-state-of-discharge power model generation
  - No need for a power meter
  - Accurate to within 4.1% for 10-second intervals

Questions?

Questions?

- Are the components mentioned above enough to model power consumption in today's smartphones?
  - Should we add Bluetooth to the mix as we have devices connected via Bluetooth all the time now?