Precise Power Characterization of Modern Android Devices

Wei Lin
Carnegie Mellon University
Pittsburgh, PA
<weilin@andrew.cmu.edu>

Joshua A. Wise
Carnegie Mellon University
Pittsburgh, PA
<jwise@andrew.cmu.ed>

October 29, 2010

http://moroso.emarhavil.com/~joshua/743wiki/

Abstract

The authors propose to perform a novel, *precise*, characterization of the power performance of modern Android cell phones. Finding the battery life on phones such as HTC's EVO 4G inadequate, the authors intend to answer the question: where is the power going? Existing tools (Intel's PowerTOP and the Android built in power meter) are demonstrated to be inadequate for this end, and an alternative mechanism of accurately measuring the power of each independent device on the system is proposed. Additionally, the authors propose easily-buildable hardware to perform the precise power measurement, as well as a series of analyses that can be performed given the recorded data to develop a model of the power consumption of such a device.

1 Introduction

With society's increasing dependency upon mobile cellular devices to perform functions beyond simply making a telephone call, the cell phone itself has evolved to into an entirely new device–the smartphone. The desire for speed and more integrated functionality has led these devices to sport bigger and more vibrant screens, higher resolution cameras equipped with flash, and support for the hundreds of thousands of little applications (apps) that a user can dream of. To keep up with communication, the modern device also comes equipped with a wide assortment of radios including 802.11n ("Wi-Fi"), Bluetooth, WiMAX, and LTE – all in addition to the basic CDMA/GSM stack. However, all this comes at a price; what used to last a week detached from a wall outlet now barely survives the better half of a day. This phenomenon is especially apparent with the Android line of cell phones; on many of these devices, thirty hours of standby time is almost unheard of. It's true that battery density has failed to scale linearly with performance, but that doesn't explain the standby performance – where is all this power going?

In its current state, power consumption modeling for the Android platform is in its infancy. There exists the official application that ships with the operating system, and there also exist a few third party power instrumentation tools. The current state of some of these tools is detailed below; in short, though, they simply paint an incomplete picture of the power consumption of an Android phone.

Given the inadequate nature of the existing tools, the goal of this project is to more accurately model the power draw of the Android platform. In doing so, the project should distinguish between the power required by the software stack (by waking the processor from idle via interrupts) and the devices integrated within the hardware stack. This will allow the attribution of the poor battery life of the platform to a source that is disproportionally drawing power. With the results from these experiments, we hope to provide the community and industry with a direction for exploration which will hopefully lead to an improvement in battery life on the Android platform.

Figure 1: An HTC EVO 4G in its standard configuration.

2 Prior work

2.1 PowerTOP

In the early days of energy-aware Linux optimizations, Intel developed a wakeup-analysis tool called PowerTOP [2]. In Linux kernel version 2.6.21, support was introduced for a "tickless kernel", in which the CPU is not to be regularly woken up; prior to that version, the kernel would cause the CPU to wake from a sleep state on every tick of the system timer. With the "tickless kernel", the system timer is *disabled* while the system is blocked on I/O (i.e., there are no processes ready to run). In early iterations of the tickless kernel, drivers were poorly optimized for a system in which wakeups cost power, and would perform inefficient tasks such as busy-waiting for hardware.

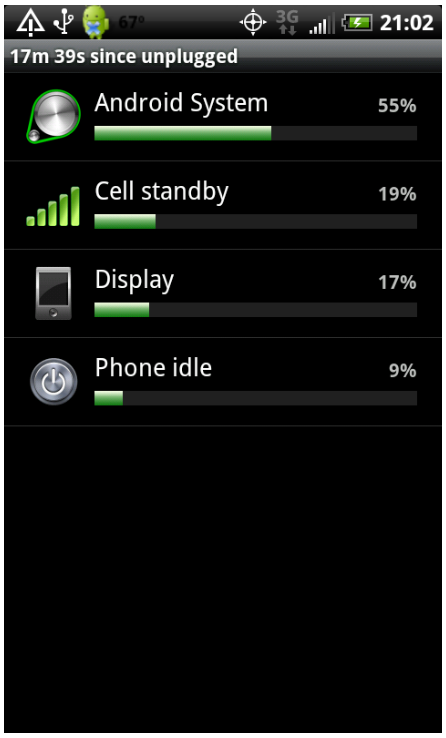
Intel's PowerTOP is a tool designed to monitor the causes of wakeups on a tickless system; by narrowing down the sources of wakeups, a user (or developer) can modify his system to avoid drivers and behaviors that result in poor power performance. PowerTOP, as a secondary function, can monitor various metrics of a CPU's sleep state ("P-states" and "C-states"); in the case of the Android platform, these apply less, as the ARM CPUs do not have this functionality.

PowerTOP gives a clearer picture of what is consuming power than a simple CPU graph, but no actual estimate of current power being consumed is available; the view that it provides is incomplete to fully diagnose a system with poor battery life.

2.2 Android built-in tools

The built-in Android application [1] is by far the most ubiquitous power monitoring application. (See Figure 2.) Unfortunately, its utility is substantially limited. It shows graphs detailing devices and significant applications that have been run on the phone and its contribution to the total energy consumed since the last charge cycle in percentage form; however, it does not show any indication of the current power draw of the device or even a prediction of the amount of battery life remaining.

The built-in application can be customized by vendors due to the open source nature of the platform to tweak the constantsfor each device, but there is evidence that the usage of this feature is imprecise at best. (Indeed, the data presented itself to be somewhat suspect; on the HTC Incredible, it showed a 4.3" AMOLED screen to be contributing only 7% to the total energy consumed.) We intend to build on the themes of this utility, but bring the mechanisms used to a higher degree of accuracy and calibration.


21:02

17m 39s since unplugged

Android System	55%
Cell standby	19%
Display	17%
Phone idle	9%

Figure 2: The Android platform's built in battery monitor.

3 Novel work

To measure the power draw of the device, a mechanism was built to analyze current flowing into the phone. This device involves using a lab supply to provide a "virtual battery" for the phone (to avoid having a physical battery buffer current spikes); current is measured and plotted versus time using a sense resistor and a small microcontroller (ATTINY) to log readings over time; a more in depth description of the hardware, as built, can be found in Section 4.

With the hardware analyzer, the power draw of each componenet of the phone will be isolated and measured. A more detailed testing plan is outlined below in Section 5; in brief, tests will be performed, comparing the individual immediate power consumption of each component on the system to a known baseline. (This assumes that system components are linearly independant in terms of power consumption.)

When the data collection process is complete, our analysis should ultimately result in a model of the power draw of the EVO 4G. In theory, the total power dissipation of the system should be the summation of all operating components; the final stage in the analysis will be to validate these results. Providing this model will pave a pathway for future research in runtime analysis of software-caused power usage above and beyond that provided by the existing Android stack.

3.1 Deliverables

At the conclusion of this project, we intend to have produced a power model equation calibrated for the HTC EVO 4G; this equation will detail the power requirements for each radio present on the phone, the screen, and the software stack running at various CPU frequencies. We will also have produced an easy-to-build power measurement apparatus customized for this phone.

4 Hardware design

The core work underlying this system is a *battery interposer* module that sits in between the target device and a benchtop power supply. The interposer serves three purposes: in brief, it *conditions* power to the device, it *measures* power consumption by the device, and it *emulates* a battery to allow the device to boot.

4.1 Power conditioning

Power to the phone is regulated and conditioned from the bench supply using a simple linear regulator – specifically, a TO-220 package LM317T. The input voltage to the board is specified as being approximately 7V; the board outputs 3.9V to the phone. (This provides a comfortable dropout range for the LM317T, and suitable headroom for the op-amps, which are incapable of swinging rail-to-rail.) A linear regulator was determined to be sufficient; the device's average case current consumption is around 300mA, which would result in dissipating 930mW in the voltage regulator.

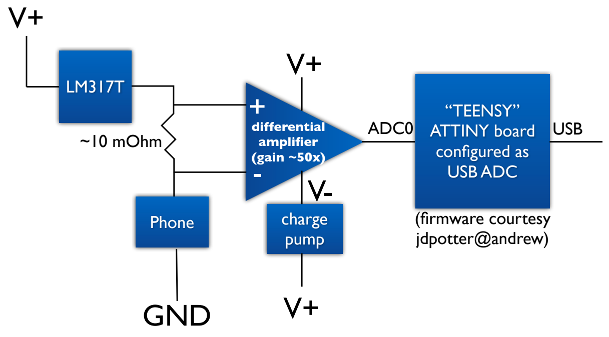

V+
LM317T
~10 mOhm
Phone
GND
V+
+
differential amplifier (gain ~50x)
-
V-
charge pump
V+
ADC0
"TEENSY" ATTINY board configured as USB ADC
USB
(firmware courtesy jdpotter@andrew)

Figure 3: Block diagram for the hardware components.

As the current consumed by the device increases, the power regulator maxes out at dissipating 3.875W if the load is drawing 1.25A. This results in linearity errors in the measurement modules from heat dissipated; the regulator's heat sink has been measured at in excess of 85°C at such high power levels. The linearity errors at these power levels were judged to be an acceptable source of error, especially in the face of the design time speedup and simplicity improvements from using the LM317T.

4.2 Power measurement

Power consumption on the phone is measured indirectly by measuring the voltage across a 10 mΩ sense resistor that is in series with the target phone. The sense resistor is placed on the high side of the phone (i.e., in series with the $V+$ lead, as opposed to the $V-$ lead); this was done to avoid issues that may arise when grounding the phone via a USB lead.

A simple differential amplifier circuit built from an LM741 is used to amplify the small voltages present across the sense resistor; at 1.25A, the voltage drop across the sense is only 10.25 mV. A gain of around 50x - 100x was chosen more or less arbitrarily given the parts available. Some attention should be given to how the LM741 is powered; since the LM741 is not capable of swinging its outputs from rail to rail, the power rails must be comfortably above the 3.9V input (and comfortably above the 5V maximum differential output), and comfortably below ground. For that reason, the 7V input to the measurement board is used as the $V+$ input to the op-amp. To generate the negative voltage for the op-amp, a Microchip TC7662B integrated charge pump controller was employed to generate -7V given the input 7V. The rails on the TC7662B were filtered using a randomly selected electrolytic capacitor.

Once the small voltage has been amplified, it is then passed through an RC filter into a "TEENSY" board featuring an ATMEGA32U4 USB microcontroller. The microcontroller is loaded with firmware (courtesy `jdpotter@andrew`) that samples the ADC0 input at a rate of approximately 15Hz, and reports the raw readings back over the USB interface; since the input samples at 15Hz, the RC filter is tuned to have a time constant of approximately 0.2 seconds, which should avoid sample aliasing issues. The TEENSY obtains its own power from

USB, and needs no regulation from the system.

4.3 Battery emulation

Since Android phones are capable of drawing more than the 2.5W specified by USB, the Android platform must always positively detect a battery's presence to buffer surges. The battery contains a small, undocumented, microcontroller attached to the two pins between $V+$ and $V-$ on the battery; presumably this microcontroller stores some state-of-charge information and can also read temperature from the battery. Without this microcontroller's presence, the platform will refuse to boot, and if the microcontroller is removed from a live system, the platform will immediately power down.

Since emulating this custom microcontroller is out of the scope of this project, the microcontroller from a real battery is used with a "piggy-back board" – a battery designed for the system is inserted into a board that connects through only the D_0, D_1, and $V-$ pins to the Android host. In this way, no power is supplied from the battery, but presence detection still functions.

4.4 Completed system

The prototype system was built on a piece of copper-clad perfboard ("donutboard"). In order to fit the assembled hardware into the phone's battery header, no solder could exist on the bottom of the board (indeed, nothing at all would be permitted to extend through the bottom of the board). To assemble the system, then, all components *and wires* must exist on the top side, which is the copper-clad side. The first prototype system can be seen in Figure 4.

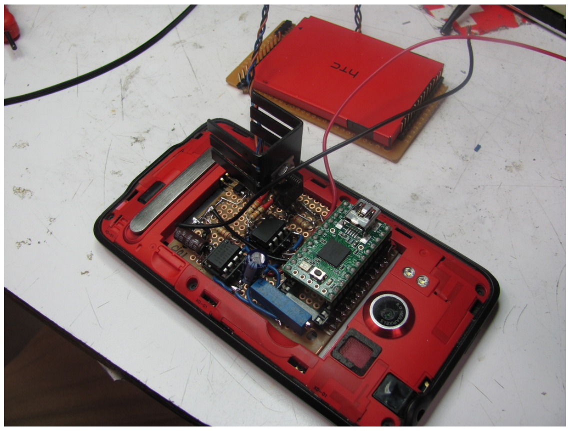


Figure 4: The first assembled prototype system loaded into an HTC EVO 4G.

5 Data collection methodology

To test the following components of the HTC EVO 4G, the phone is first set to the baseline configuration as described below. Afterwards, each component will be individually powered on and placed in various states under various loads to see its contribution to power consumption.

5.1 Baseline

The baseline configuration for the phone will have the phone drawing as little power as possible; as few of the system peripherals as possible will be enabled. Specifically, the CPU will be set to the lowest DVFS setting (240MHz, with all system voltages determined by the kernel's default DVFS tables); the display's backlight will be turned off; the CDMA radio will be set to "airplane mode"; the WiMAX, 802.11n, and Bluetooth radios will be disabled; and the system will be set to attempt to draw no power from the USB. All running applications should be "force-stopped" before the tests begin.

Preliminary measurements of this state indicate that the system draws 110mA in this mode; more detailed measurements are yet to come.

5.2 Communication radios

To analyze the power requirements of the various radios present on the phone, power consumption will be measured with each radio individually set to each the following states:

- off
- on, and searching for signal
- on, idle
- on, transferring at various data rates covering light, medium and heavy loads (normalized to the capability of the radio)

These states will be tested (as applicable) for each of the radios on the system – specifically:

- CDMA: voice
- CDMA: 3G data
- WiMAX (4G data)
- 802.11n (Wi-Fi)
- Bluetooth

5.3 Display backlight

The display's backlight is analyzed in a similar fashion; measurements with the backlight off and on at various brightness settings will be taken. The backlight on the HTC EVO 4G can be linearly modulated between the values of 1 and 255; a value of 0 turns off the backlight (and display controller) fully. While the testbench is running, the Android software stack believes that the display is turned off, and will make no attempts to interfere with testbench control of the backlight.

5.4 Application processor

Power consumption on the processor will be analyzed by varying the running DVFS setting and stressing different subsystems on the CPU's applications processor (AP) core. For all discrete voltage/frequency steps the following benchmarks will be performed:

- **LINPACK[3]**: floating point unit (FPU) stress test
- **Dhrystone[4]**: integer arithmetic stress test
- **STREAM[5]**: memory stress test
- power correlation with CPU wakeups per second from sleep

The application processor does not provide a continuously variable DVFS set; the available settings between 240MHz and 998MHz are quantized in increments of roughly 38MHz. The AP voltage cannot be varied independently of the frequency setting; the voltage is linearly scaled between 1.050V and 1.300V depending on the frequency.

5.5 Miscellaneous

The HTC EVO 4G has a smattering of other peripherals; in brief, they include:

- the LED flash and flashlight modules for the camera;
- the backlight for the capacitative keys;
- the touch input module;
- and the system status LEDs.

The obvious isolation methodology can be used to determine how much power each of these devices consumes.

6 Preliminary results

As of October 29, 2010, the hardware needed to measure power on the system has been fully designed and built. The target Android device has been successfully booted using with the probe in place, and various power control mechanisms on the phone have been experimented with; specifically, a set of commands has been developed to modify the CPU frequency, the backlight, and the set of LEDs on the system.

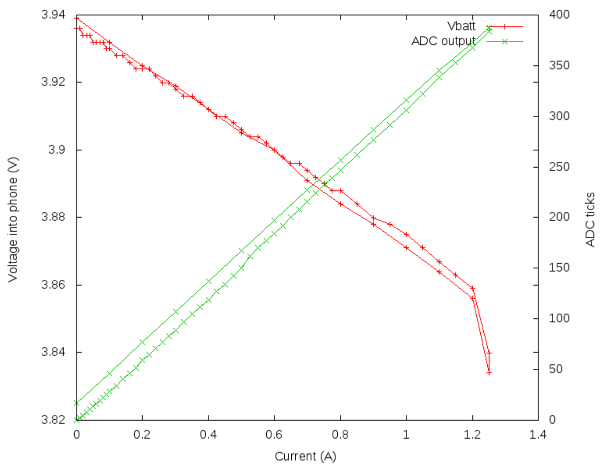


Figure 5: Calibration data from the ADC.

The ADC readings have been calibrated to real current and voltage values (and hence power readings) with the aid of an electronic load. The electronic load was set to a range of constant current values, starting at no current, and ending at 1.25A. The resulting ADC values (and voltage at the load) were graphed with respect to the delivered current; the results can be seen in Figure 5.

Two interesting misbehaviors of the current hardware were noted. First, when the system heats up as a result of heavy load, the ADC reading drifts higher; this problem is particularly notable after the system makes a sustained excursion above 0.75A or so and then drops back below 0.25A. When this occurs, the ADC may read an offset as high as 80mA above what it should. We do not anticipate this to be a substantial threat to the results that we acquire; the impact of this can be mitigated by simply measuring high power and low power modes independently.

More serious of a misbehavior is the ability of the 3G data radio to interfere with the analog input stage. When the 3G radio is transmitting at high data rates, the output of the op-amp begins to swing negative, instead of reading the true current through the sense; the ADC, then, reads zero. If this becomes a problem in our experiments, we intend to wrap the device in polyethylene wrap and then in an aluminum or copper foil. With the foil grounded, a Faraday cage effect should take place to negate the electromagnetic interference, and the current measurement circuitry should function during radio transmission.

The board has been successfully used to retrieve data from a running system, but no formal experiments have yet been performed to calibrate either a baseline or a running system; only informal "curiosity experiments" have been run. Given these, though, the board has been validated for data collection *in situ*. For mobility-related experiments in longer-term data collection, a portable power pack may be proposed; a standard 7.2V NiMH battery pack should suffice to run the device over the course of a day.

7 Schedule and milestones

Over the remainder of this project, we intend to meet the following milestones:

- **Oct. 11, 2010** – Hardware designed and parts collected.
- **Oct. 26, 2010** – Hardware built; initial results gathered (checkpoint 2)
- **Nov. 12, 2010** – Data collection complete; begin analysis.
- **Dec. 2, 2010** – Poster Presentation.
- **Dec. 3, 2010** – Written report written.

References

[1] Android Open Source Project Contributors, *android.os.BatteryStats*, source from `http://android.git.kernel.org/?p=platform/frameworks/base.git;a=blob;f=core/java/android/os/BatteryStats.java;hb=HEAD`. Accessed October 29, 2010.

[2] Intel Corporation, *"LessWatts.org – Saving Power on Intel systems with Linux – PowerTOP,"* on the Internet at `http://www.lesswatts.org/projects/powertop/`. Accessed October 29, 2010.

[3] Burkardt, John, *"The LINPACK Benchmark,"* on the Internet at `http://people.sc.fsu.edu/~jburkardt/c_src/linpack_bench/linpack_bench.html`. Accessed October 29, 2010.

[4] Weicker, Reinhold P., *"Dhrystone Benchmark: Rationale for Version 2 and Measurement Rules,"*, SIGPLAN Notices 23,8, August 1988.

[5] McCalpin, John D., *"Memory Bandwidth and Machine Balance in Current High Performance Computers,"* IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter, December 1995.