# An Analysis of Power Consumption in a Smartphone

Aaron Carroll

*NICTA and University of New South Wales*

Aaron.Carroll@nicta.com.au

Gernot Heiser

*NICTA, University of New South Wales and Open Kernel Labs*

gernot@nicta.com.au

## Abstract

Mobile consumer-electronics devices, especially phones, are powered from batteries which are limited in size and therefore capacity. This implies that managing energy well is paramount in such devices.

Good energy management requires a good understanding of where and how the energy is used. To this end we present a detailed analysis of the power consumption of a recent mobile phone, the Openmoko Neo Freerunner. We measure not only overall system power, but the exact breakdown of power consumption by the device's main hardware components. We present this power breakdown for micro-benchmarks as well as for a number of realistic usage scenarios. These results are validated by overall power measurements of two other devices: the HTC Dream and Google Nexus One.

We develop a power model of the Freerunner device and analyse the energy usage and battery lifetime under a number of usage patterns. We discuss the significance of the power drawn by various components, and identify the most promising areas to focus on for further improvements of power management. We also analyse the energy impact of dynamic voltage and frequency scaling of the device's application processor.

## 1 Introduction

Mobile devices derive the energy required for their operation from batteries. In the case of many consumer-electronics devices, especially mobile phones, battery capacity is severely restricted due to constraints on size and weight of the device. This implies that energy efficiency of these devices is very important to their usability. Hence, optimal management of power consumption of these devices is critical.

At the same time, device functionality is increasing rapidly. Modern high-end mobile phones combine the functionality of a pocket-sized communication device with PC-like capabilities, resulting in what are generally referred to as *smartphones* [11]. These integrate such diverse functionality as voice communication, audio and video playback, web browsing, short-message and email communication, media downloads, gaming and more. The rich functionality increases the pressure on battery lifetime, and deepens the need for effective energy management.

A core requirement of effective and efficient management of energy is a good understanding of *where* and *how* the energy is used: how much of the system's energy is consumed by which parts of the system and under what circumstances.

In this paper we attempt to answer this question and thus provide a basis for understanding and managing mobile-device energy consumption. Our approach is to measure the power consumption of a modern mobile device, the Openmoko Neo Freerunner mobile phone, broken down to the device's major subsystems, under a wide range of realistic usage scenarios.

Specifically, we produce a breakdown of power distribution to CPU, memory, touchscreen, graphics hardware, audio, storage, and various networking interfaces. We derive an overall energy model of the device as a function of the main usage scenarios. This should provide a good basis for focusing future energy-management research for mobile devices.

Furthermore, we validate the results with two additional mobile devices at a less detailed level: the HTC Dream and Google Nexus One. Along with the Freerunner, these three devices represent approximately the last three to four years of mobile phone technology.

The paper is structured as follows. In Section 2 we describe our measurement platform and benchmarking methodology. Section 3 describes each experiment and presents the results, and in Section 4 we perform a coarse-grained validation of the results. We then analyse this data in Section 5. Section 6 surveys existing work. Finally, we conclude in Section 7.

## 2   Methodology

Our approach to profiling energy consumption is to take physical power measurements at the component level on a piece of real hardware. In this section, we describe the hardware and software used in the experiments, and explain our benchmarking methodology.

There are three elements to the experimental setup: the device-under-test (DuT), a hardware data acquisition (DAQ) system, and a host computer.

### 2.1   Device under test

The DuT was the Openmoko Neo Freerunner (revision A6) mobile phone. It is a 2.5G smartphone featuring a large, high-resolution touchscreen display, and many of the peripherals typical of modern devices. Table 1 lists its key components. The notable differences between our device and a modern smartphone are the lack of a camera and 3G modem.

| Component | Specification |
|---|---|
| SoC | Samsung S3C2442 |
| CPU | ARM 920T @ 400 MHz |
| RAM | 128 MiB SDRAM |
| Flash | 256 MiB NAND |
| Cellular radio | TI Calypso GSM+GPRS |
| GPS | u-blox ANTARIS 4 |
| Graphics | Smedia Glamo 3362 |
| LCD | Topploy $480 \times 640$ |
| SD Card | SanDisk 2 GB |
| Bluetooth | Delta DFBM-CS320 |
| WiFi | Accton 3236AQ |
| Audio codec | Wolfson WM8753 |
| Audio amplifier | National Semiconductor LM4853 |
| Power controller | NXP PCF50633 |
| Battery | 1200 mAh, 3.7 V Li-Ion |

Table 1: Freerunner hardware specifications.

This device was selected because the design files, particularly the circuit schematics [7], are freely available. This is critical for our approach to power measurement, which relies on understanding the power distribution network at the circuit level. For this reason, few other devices would be suitable.

The high-level architecture of the Freerunner is shown in Figure 1. The total system memory is split equally between two banks, one external RAM package, and one on-chip. All peripherals except the graphics chip communicate with the application processor (CPU) by programmed I/O over various serial buses.

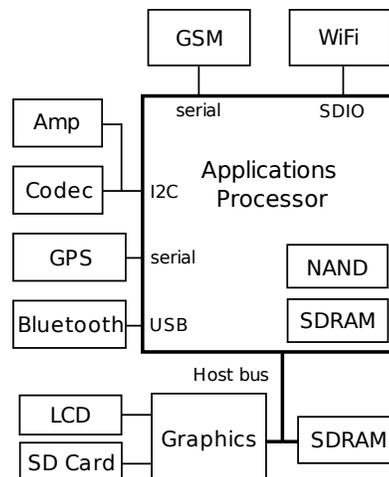The other devices studied, the HTC Dream (G1) and Google Nexus One (N1), are described in Section 4.



Figure 1: Architecture of the Freerunner device, showing the important components and their interconnects.

### 2.2   Experimental setup

To calculate the power consumed by any component, both the supply voltage and current must be determined.

To measure current, we inserted sense resistors on the power supply rails of the relevant components—this is relatively simple on the DuT selected, since most of them have been designed with placeholders for sense resistors, factory-populated with $0\,\Omega$. Where this was not the case, choke inductors could be reused in the same way. In both cases, we replaced the part with a current-sense resistor selected such that the peak voltage drop did not exceed 10 mV, which in all cases is less than 1 % of the supply voltage and therefore presents an acceptably small perturbation. With a known resistance and measured voltage drop, current can be determined by Ohm's law.

To measure the voltages, we used a National Instruments PCI-6229 DAQ, to which the sense resistors were connected via twisted-pair wiring. The key characteristics of this hardware are summarised in Table 2.

| Characteristic | Value |
|---|---|
| Max. sample rate | 250 kS/s |
| Input ranges | $\pm 0.2$ V, $\pm 1$ V, $\pm 5$ V and $\pm 10$ V |
| Resolution | 16 b |
| Accuracy | 112 $\mu$V @ $\pm 0.2$ V range |
| | 1.62 mV @ $\pm 5$ V range |
| Sensitivity | 5.2 $\mu$V @ $\pm 0.2$ V range |
| | 48.8 $\mu$V @ $\pm 5$ V range |
| Input impedance | 10 G$\Omega$ |

Table 2: National Instruments PCI-6229 DAQ specifications [6].

The sense-resistor voltage drops were sampled differentially at the $\pm 0.2$ V input range. We used the same physical connections to measure supply voltages; these were taken relative to ground from the component side of the resistors, in the $\pm 5$ V range.

We were able to directly measure the power consumed by the following components: CPU core, RAM (both banks), GSM, GPS, Bluetooth, LCD panel and touchscreen, LCD backlight, WiFi, audio (codec and amplifier), internal NAND flash, and SD card. Since the graphics module had too many supply rails to measure directly, we instead used a combination of direct and subtractive measurements.

Power to the DuT was supplied through a bench power supply connected to the phone's battery terminals so we did not need to deal with battery management. This also prevents the OS's power policies from interfering with the benchmarks. Total system power consumption was measured at this point by inserting a sense resistor between the supply and the phone. For the G1 and N1 we measured total system power by inserting a sense resistor between the device and its battery.

Measuring backlight power required special attention, because its supply voltage (10–15 V, depending on the brightness) far exceeded the maximum range supported by our DAQ hardware. To resolve this, we pre-scaled the backlight voltage with some external circuitry, consisting of a high-input-impedance voltage follower feeding a fixed voltage divider. This brought the voltage within the $\pm 5$ V range.

### 2.2.1 Voltage regulation efficiency

Our measurement approach yields the power directly consumed by each component. However, a certain amount of additional power is lost in converting the supply (i.e. battery) voltage to the levels required by the components. We have not included this factor in the results reported, because the conversion efficiencies are unknown. However, based on the data sheet of a similar part (the NXP PCF 50606), the efficiency conversion is likely to be in the range of 75–85 %, depending on the current drawn.

Because of this, we differentiate between "total power", measured at the battery, and "aggregate power", measured as the sum of individual component measurements. The latter assumes no power is consumed in the non-instrumented components, and while we haven't been able to measure precisely what their contribution is, it is certainly less than 10 %, and probably within a few percent of the aggregate consumption.

One exception to this is the backlight boost converter, the efficiency of which we measured to be 67 %. We determined the cause of this poor efficiency to be heating in an external component. We found no evidence to suggest this is an issue for any of the other voltage regulators.

## 2.3 Software

The DuT ran the Freerunner port of the Android 1.5 operating system [1] using the Linux v2.6.29 kernel. Except for the CPU micro-benchmark, the kernel was configured with the `ondemand` frequency scaling governor, using 100 MHz and 400 MHz—the only two frequencies supported by both the hardware and OS.

On the host system we ran the power-data collection software which interfaced with the National Instruments DAQmxBase 3.3 library to collect raw data from the DAQ, aggregate it, and write the result to file for post-processing. Each data point collected was an average of 2000 consecutive voltage samples. We configured the tool such that a complete power snapshot of the system could be generated approximately every 400 ms.

The benchmarks were coordinated on the host machine, which communicated with the DuT via a serial connection. It was responsible for executing benchmarks on the DuT, synchronising the power measurement software with the benchmark, and collecting other relevant data.

## 2.4 Benchmarks

We ran two types of benchmarks. First, a series of micro-benchmarks designed to independently characterise components of the system, particularly their peak and idle power consumption.

Second, we ran a series of macro-benchmarks based on real usage scenarios. For low-interactivity applications (e.g. music playback), we simply launched them from the command line. For interactive applications, such as web browsing, we took a trace-based approach. A trace consisted of a sequence of input events, including a time-stamp, the name of the device providing the input (the touchscreen or one of two pushbuttons), and for touchscreen events, the coordinates of the touch. The Linux kernel provides this information by reading from the `/dev/input/event*` device files. To collect the trace, we used the target application normally, while in the background storing the input events to file. We then replayed the events under benchmarking conditions by writing the collected data to the `/dev/input/event*` files at the correct time.

Although this approach does bypass the hardware and interrupt paths that would usually be followed for a touchscreen event, our measurements showed the additional power to be negligible. The vast majority of energy required to handle a touchscreen event is consumed in delivering it from the kernel to software.
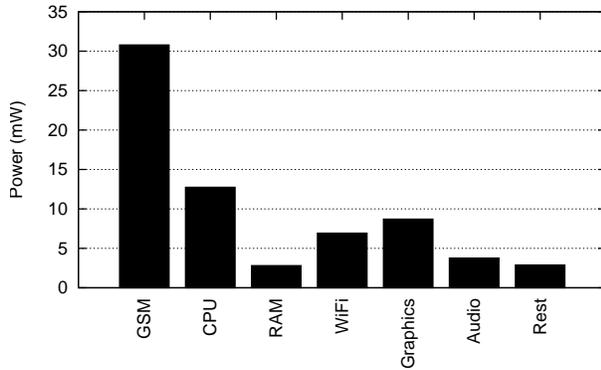
Figure 2: Power breakdown in the suspended state. The aggregate power consumed is 68.6 mW.



Figure 3: Average power consumption while in the idle state with backlight off. Aggregate power is 268.8 mW.

## 3 Results

### 3.1 Baseline cases

Prior to running any benchmarks, we established the baseline power state of the device, when no applications are running. There are two different cases to consider: *suspended* and *idle*. For the idle case, there is also the application-independent power consumption of the backlight to consider.

#### 3.1.1 Suspended device

A mobile phone will typically spend a large amount of time in a state where it is not actively used. This means that the application processor is idle, while the communications processor performs a low level of activity, as it must remain connected to the network be able to receive calls, SMS messages, etc. As this state tends to dominate the time during which the phone is switched on, the power consumed in this state is critical to battery lifetime.

The Android OS running on the application processor aggressively suspends to RAM during idle periods, whereby all necessary state is written to RAM and the devices are put into low-power sleep modes (where appropriate). To quantify power use while suspended, we forced the device into Android's suspended state and measured the power over a 120 second period. Figure 2 shows the results, averaged over 10 iterations. The average aggregate power is 68.6 mW, with a relative standard deviation (RSD) of 8.2 %. The large fluctuations are largely due to the GSM (14.4 % RSD) and graphics (13.0 %) subsystems.

The GSM subsystem power clearly dominates while suspended, consuming approximately 45 % of the overall power. Despite maintaining full state, RAM consumes neg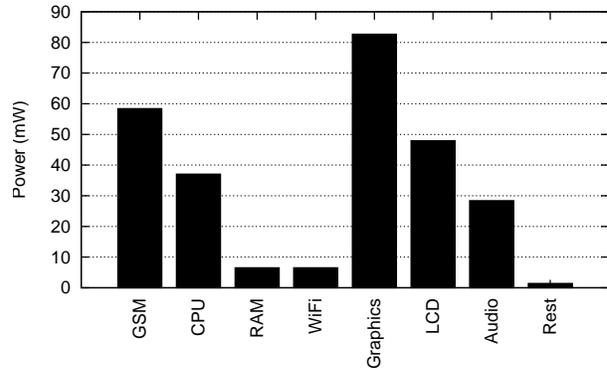ligible power—less than 3 mW. Note that the GSM subsystem in our device does not use system memory—it has its own bank of RAM which we include in the GSM power measurements.

#### 3.1.2 Idle device

The device is in the idle state if it is fully awake (not suspended) but no applications are active. This case constitutes the static contribution to power of an active system. We run this case with the backlight turned off, but the rest of the display subsystem enabled.

Figure 3 shows the power consumed in the idle state. As with the suspend benchmark, we ran 10 iterations, each of 120 seconds in the idle state. Power consumed in this state was very stable, with an RSD of 2.6 %, influenced largely by GSM, which varied with an RSD of 30 %. All other components showed an RSD below 1 %.

Figure 3 shows that the display-related subsystems consume the largest proportion of power in the idle state—approximately 50 % due to the graphics chip and LCD alone, and up to 80 % with backlight at peak brightness. GSM is also a large consumer, at 22 % of aggregate power.

#### 3.1.3 Display

Figure 4 shows the power consumed by the display backlight over the range of available brightness levels. That level is an integer value between 1 and 255, programmed into the power-management module, used to control backlight current. Android's brightness-control user-interface provides linear control of this value between 30 and 255.

The minimum backlight power is approximately 7.8 mW, the maximum 414 mW, and a centred slider corresponds to a brightness level of 143, consuming 75 mW. The backlight consumes negligible power when disabled (as in the above idle benchmarks).
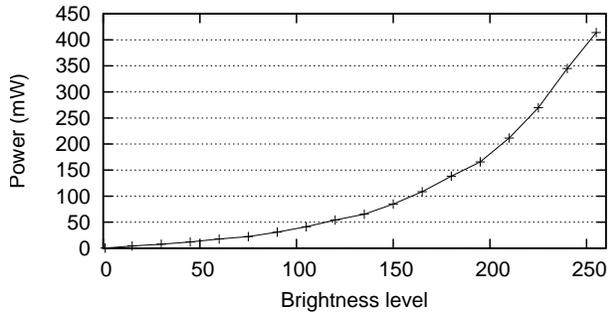
Figure 4: Display backlight power for varying brightness levels.



Figure 5: CPU and RAM power when running SPEC CPU2000 micro-benchmarks, sorted by CPU power.

We also measured how the content displayed on the LCD affected its power consumption: 33.1 mW for a completely white screen, and 74.2 mW for a a black screen. Display content can therefore affect overall power consumption by up to 43 mW.

## 3.2 Micro-benchmarks

As mentioned in Section 2.4, we used micro-benchmarks to determine the contribution to overall power from various system components. Specifically we used benchmarks to exercise the application processor (CPU and memory), the flash storage devices, and the network interfaces.

### 3.2.1 CPU and RAM

To measure CPU and RAM power, we ran a subset of the SPEC CPU2000 suite. There are several reasons for not running all benchmarks of the suite. Firstly, we could only use benchmarks which we could build and run on the Android OS, which rules out those written in C++ or Fortran, due to Android's lack of run-time support for these languages. They also needed to fit into the phone's limited memory and their execution times needed to be short enough to give reasonable turn-around. Finally, we were only interested in establishing the power consumption of CPU and memory, rather than making comparisons between different platforms' algorithms, hence completeness of the suite was not a relevant consideration.

From the candidates remaining according to the above criteria, we selected a set representing a good spectrum of CPU and memory utilisation, from highly CPU-bound to highly memory-bound. We determined memory-boundedness by running the entire suite on a server Linux system and comparing the slowdown due to frequency scaling. Snowdon et al. [9] show that this slowdown is primarily d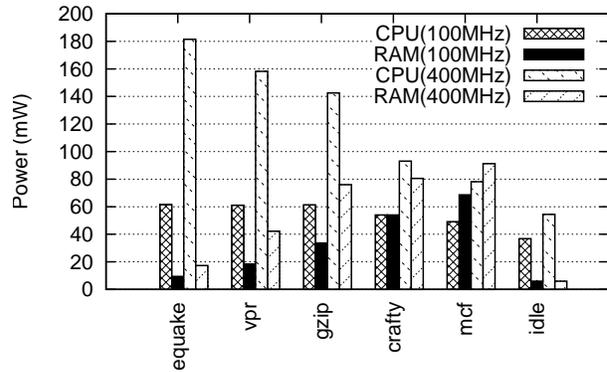ue to memory-boundedness. While we do not expect the benchmarks to behave similarly on the different platforms, our aim is only to select benchmarks with different characteristics.

The SPEC CPU2000 benchmarks ultimately selected are equake, vpr, gzip, crafty and mcf.

For each of the benchmarks, we measured the average CPU and RAM power at fixed core frequencies of 100 MHz and 400 MHz. We also measured power for the system in the idle state. Figure 5 shows these results, averaged over 10 runs. The RSD is less than 3 % in all cases.

For the idle, equake, vpr and gzip workloads, CPU power dominates RAM power considerably at both frequencies. However, crafty and mcf show that RAM power can exceed CPU power, albeit by a small margin.

Table 3 shows the effect of frequency scaling on the performance, as well as combined CPU and RAM power and energy of the benchmarks. The wide range of slowdown factors across the different benchmarks validates our selection of workloads as representing a range of CPU/memory utilisations.

| Benchmark | Performance | Power | Energy |
|-----------|-------------|-------|--------|
| equake    | 26 %        | 36 %  | 135 %  |
| vpr       | 31 %        | 40 %  | 125 %  |
| gzip      | 38 %        | 43 %  | 112 %  |
| crafty    | 63 %        | 62 %  | 100 %  |
| mcf       | 74 %        | 69 %  | 93 %   |
| idle      | -           | 71 %  | -      |

Table 3: SPEC CPU2000 performance, power and energy of 100 MHz relative to 400 MHz. Both CPU and RAM power/energy are included.
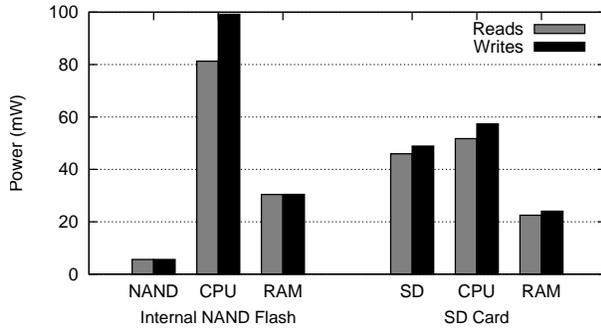
Figure 6: SD, NAND, CPU and RAM power for flash storage read and write benchmarks.



Figure 7: Power consumption of WiFi and GSM modems, CPU, and RAM for the network micro-benchmark.

### 3.2.2 Flash storage

Bulk storage on the Freerunner device is provided by 256 MiB of internal NAND flash, and an external micro Secure Digital (SD) card slot. To measure their maximum power consumption, we used the Linux `dd` program to perform streaming reads and writes. For reads we copied a 64 MiB file, filled with random data, to `/dev/null` in 4 KiB blocks. For writes, 8 MiB of random data was written, with an `fsync` between successive 4 KiB blocks to ensure predictability of writes. Between each iteration we forced a flush of the page cache.

Figure 6 shows the power consumed by the NAND flash and SD card, as well as the CPU and RAM, averaged over 10 iterations of each workload. Table 4 shows the corresponding data throughput, efficiency (including NAND/SD power and the CPU and RAM power to support it), and idle power consumption. The power and throughput RSD is less than 5 % in all cases.

The graphics module, which contains the physical SD card interface, showed a power increase of 2.2 mW (2.6 % above static) for writes, and a 21.1 mW increase (26 %) for reads.

| Metric | NAND | SD |
|---|---|---|
| Idle (mW) | 0.4 | 1.4 |
| Read | | |
|    throughput (MiB/s) | 4.85 | 2.36 |
|    efficiency (MiB/J) | 65.0 | 31.0 |
| Write | | |
|    throughput (KiB/s) | 927.1 | 298.1 |
|    efficiency (MiB/J) | 10.0 | 5.2 |

Table 4: Flash storage power and performance.

### 3.2.3 Network

In this benchmark we stressed the two main networking components of the device: WiFi and GPRS (provided by the GSM subsystem). The test consisted of downloading a file via HTTP using `wget`. The files contained random data, and were 15 MiB for WiFi, and 50 KiB for GPRS. The results of 10 iterations of the benchmark are shown in Figure 7.

WiFi showed a throughput of $660.1 \pm 36.8$ KiB/s, and GPRS $3.8 \pm 1.0$ KiB/s. However, they both show comparable power consumption far exceeding the contribution of the RAM and CPU. The increased CPU and RAM power for WiFi reflects the cost of processing data with a higher throughput. Despite highly-variable throughput, GSM showed a relatively consistent power consumption with an RSD of approximately 2 %.

To test the effect of signal strength on power and throughput, we re-ran the network benchmarks with the device shielded within a metal box of 2 mm thickness. Over GPRS, this resulted in an increase of GSM power of 30 %, but no effect on throughput. The shielding resulted in a reported signal strength drop of 10 dBm. Over WiFi, the signal strength dropped by only 2 dBm, and no effect on throughput or power consumption was observed.

### 3.2.4 GPS

To measure power consumption of the GPS subsystem, we enabled the module and ran the `GPS Status 2` Android application. Table 5 shows the power consumed by the GPS module in three situations; using only the internal antenna, with an external active antenna attached, and when idle (i.e. powered down).

We noticed that the energy consumption of the module is largely independent of the received signal—neither the number of satellites, nor the signal strength, had any appreciable effect.

This observation is contrary to the part's data sheet

| State | Power (mW) |
|---|---|
| Enabled (internal antenna) | $143.1 \pm 0.05\,\%$ |
| Enabled (external antenna) | $166.1 \pm 0.04\,\%$ |
| Disabled | 0.0 |

Table 5: GPS energy consumption.



Figure 8: Audio playback power breakdown. Aggregate power consumed is 320.0 mW.

[10], which specifies that power consumption should drop by approximately 30 % after satellite acquisition. It is unclear why we did not see such behaviour; perhaps due to the GPS module itself, or more likely an error in hardware integration or software. In addition, the power-management features of the device are not exploited by software. Thus, these figures should only be considered worst-case.

## 3.3 Usage scenarios

Here we show the results of using macro-benchmarks to determine power consumption under a number of typical usage scenarios of a smartphone. Specifically we examined audio and video playback, text messaging, voice calls, emailing and web browsing.

### 3.3.1 Audio playback

This benchmark is designed to measure power in a system being used as a portable media player. The sample music is a 12.3 MiB, 537-second stereo 44.1 kHz MP3, with the output to a pair of stereo headphones. The measurements are taken with the backlight off (which is representative of the typical case of someone listening to music or podcasts while carrying the phone in their pocket). However, GSM power was included, as the realistic usage scenario includes the phone being ready to receive calls or text messages.

Figure 8 shows the power breakdown for this benchmark at maximum volume, averaged over 10 iterations. The audio file is stored on the SD card. Between successive iterations we forced a flush of the buffer cache to ensure that the audio file was re-read each time.

The results show the audio subsystem (amplifier and codec) consuming 33.1 mW with an RSD of less than 0.2 %. Approximately 58 % of this power is consumed by the codec, with the remaining 42 % used by the amplifier. Compared with the idle state, this corresponds to a negligible change in codec power, with amplifier power increasing by 80 %. Overall, the audio subsystem accounts for less than 12 % of power consumed.

In addition to maximum volume, we also measured the system at 13 % volume. This showed little change—the audio subsystem power decreased by 4.3 mW 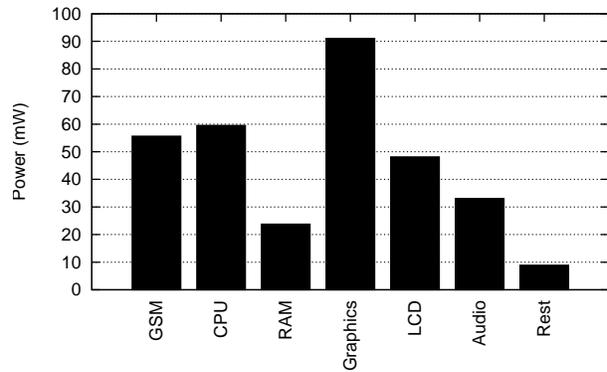(approximately 14 %), mostly in the amplifier. However, for unknown reasons, the power consumed by the graphics chip increased by 4.6 mW. As a result, the additional power consumed in the high-volume benchmark is less than 1 mW compared with the low-volume case.

Again, maintaining a connection to the GSM network requires a significant and highly variable amount of power, specifically $55.6 \pm 19.7$ mW in this case. While the MP3 file is loaded from the SD card, the cost of doing so is negligible at $< 2\,\%$ of total power.

### 3.3.2 Video playback

In this benchmark we measured the power requirements for playing a video file. We used a 5 minute, 12.3 MiB H.263-encoded video clip (no sound), and played it with Android's camera application. Again we forced a flush of the buffer cache between iterations. The power averaged over 10 iterations is shown in Figure 9.

Since the purpose of the macro-benchmarks is to analyse the full system, we have included backlight power in the results. However, rather than arbitrarily choosing a single brightness, we have plotted the results at 0 %, 33 %, 66 %, and 100 %, corresponding to the position of Android's brightness-control slider. These correspond to brightness levels of 30, 105, 180 and 255 respectively. GSM power is again included.

While the CPU is the biggest single consumer of power (other than backlight), the display subsystems still account for at least 38 % of aggregate power, up to 68 % with maximum backlight brightness. The energy cost of loading the video from the SD card is negligible, with an average power of 2.6 mW over the length of the benchmark.

### 3.3.3 Text messaging

We benchmarked the cost of sending an SMS by using a trace of real phone usage. This consists of loading
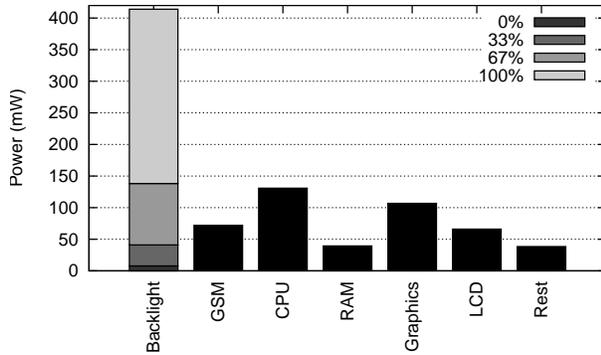
Figure 9: Video playback power breakdown. Aggregate power excluding backlight is 453.5 mW.
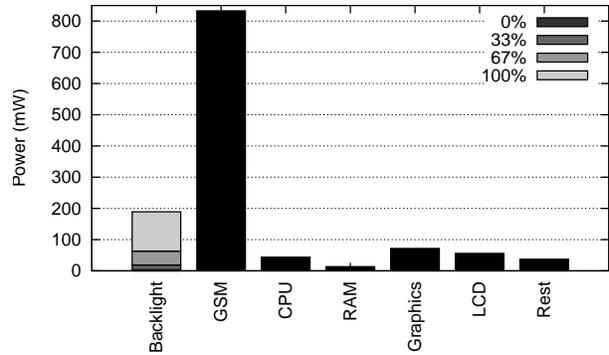


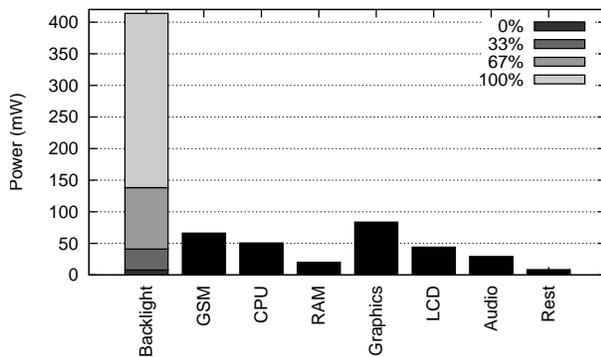Figure 11: GSM phone call average power. Excluding backlight, the aggregate power is 1054.3 mW.



Figure 10: Power breakdown for sending an SMS. Aggregate power consumed is 302.2 mW, excluding backlight.
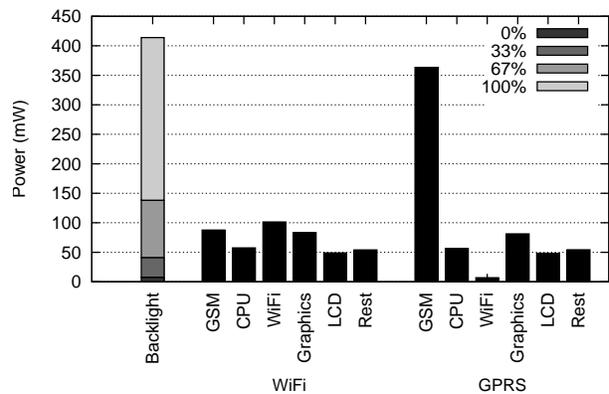


Figure 12: Power consumption for the email macro-benchmark. Aggregate power consumption (excluding backlight) is 610.0 mW over GPRS, and 432.4 mW for WiFi.

the contacts application and selecting a contact, typing and sending a 55-character message, then returning to the home screen; lasting a total of 62 seconds. To ensure the full cost of the GSM transaction is included, we measured power for an additional 20 seconds. The average result of 10 iterations of this benchmark are shown in Figure 10. Again, the power for four backlight brightness levels is shown.

Power consumed is again dominated by the display components. The GSM radio shows an average power of $66.3 \pm 20.9$ mW, only 7.9 mW greater than idle over the full length of the benchmark, and accounting for 22 % of the aggregate power (excluding backlight). All other components showed an RSD of below 3 %.

### 3.3.4 Phone call

Figure 11 shows the power consumption when making a GSM phone call. The benchmark is trace-based, and includes loading the dialer application, dialing a number, and making a 57-second call. The dialled device was configured to automatically accept the call after 10 sec-

onds. Thus, the time spent in the call was approximately 40 seconds, assuming a 7-second connection time. The total benchmark runs for 77 seconds.

GSM power clearly dominates in this benchmark at $832.4 \pm 99.0$ mW. Backlight is also significant, however note that its average power is lower than in other benchmarks, since Android disables the backlight during the call. The backlight is active for approximately 45 % of the total benchmark.

### 3.3.5 Emailing

For this benchmark, we used Android's email application to measure the cost of sending and receiving emails. The workload consisted of opening the email application, downloading and reading 5 emails (one of which included a 60 KiB image) and replying to 2 of them. The results of the benchmark are shown in Figure 12, averaged over 10 iterations.

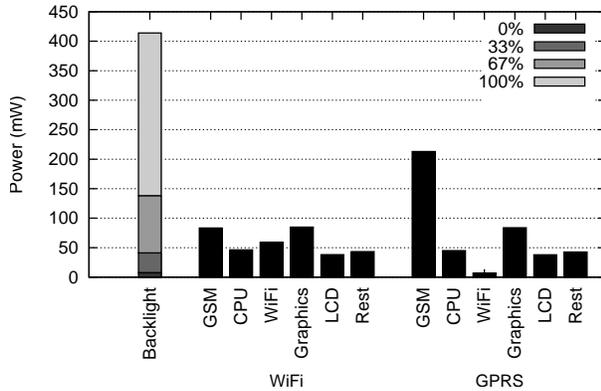The power breakdown between the GPRS and WiFi

Figure 13: Web browsing average power over WiFi and GPRS. Aggregate power consumption is 352.8 mW for WiFi, and 429.0 mW for GPRS, excluding backlight.



Figure 14: Display, button and keyboard backlight power on the G1.

benchmarks is comparable, except for the GSM and WiFi radios. Despite presenting identical workloads to the radios, GSM consumes more than three times the power of WiFi.

### 3.3.6 Web browsing

Our last benchmark measured the power consumption for a web-browsing workload using both GPRS and WiFi connections. The benchmark was trace-based, ran for a total of 490 seconds, and consisted of loading the browser application, selecting a bookmarked web site and browsing several pages. We used the BBC News website, which we mirrored locally to improve the reliability of the benchmark. After each run, the browser cache was cleared. The results averaged over 10 iterations are shown in Figure 13, including backlight power at 4 brightness levels.

GPRS consumes more power than WiFi by a factor of 2.5. The other components do not display any significant difference between the two benchmarks.

This benchmark, along with the emailing benchmark, are the only two where a more modern phone can be expected to show significantly different results. The much higher bandwidth supported by 3G protocols is likely to result in them being more power-hungry.

## 4 Validation

In this section, we measure the power consumption of two additional smartphones; the HTC Dream (G1), and the Google Nexus One (N1). Table 6 lists the key features of these devices.

We measure the full-system power of these platforms at the battery; per-component measurements are not possible because the necessary documentation (schematics,
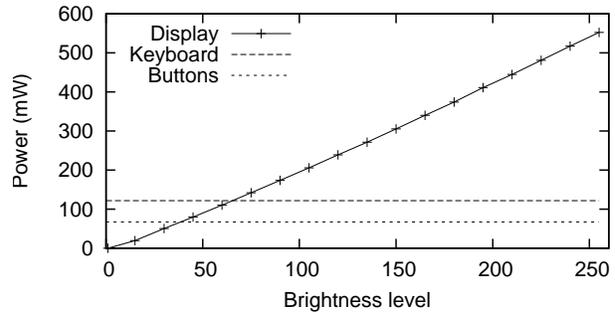
etc.) are not available to us. Moreover, there is no reason to expect these production devices would be capable of the type of instrumentation we have performed on the Freerunner, since the additional components and PCB area would increase the per-unit cost.

### 4.1 Display and backlight

Figure 14 plots the power consumption of the various backlights on the G1 as a function of brightness level. In addition to the LCD display backlight, the G1 features a backlit physical keyboard and buttons which are not present on either the Freerunner or the N1. These backlights do not have any brightness control, and contribute 189 mW when both enabled. The content of the LCD display can affect power consumption by up to 17 mW.

The Nexus One features an OLED display, and as such does not require a separate backlight like the Freerunner and G1. Furthermore, the effects of display content and brightness on power consumption are more tightly coupled. For instance, the OLED power consumption for a black screen is fixed, regardless of the brightness setting. For a completely white screen at minimum brightness, an additional 194 mW is consumed, and at maximum brightness, 1313 mW.

### 4.2 CPU

Figure 15 plots the G1 and N1 total system power under our SPEC CPU2000 workloads at the minimum and maximum frequencies supported by the respective device: 246 MHz and 384 MHz on the G1, and 245 MHz and 998 MHz on the N1. Table 7 shows the percentage slowdown, and reduction in full system power, due to frequency scaling. This benchmark was run with the display system powered down and all radios disabled.

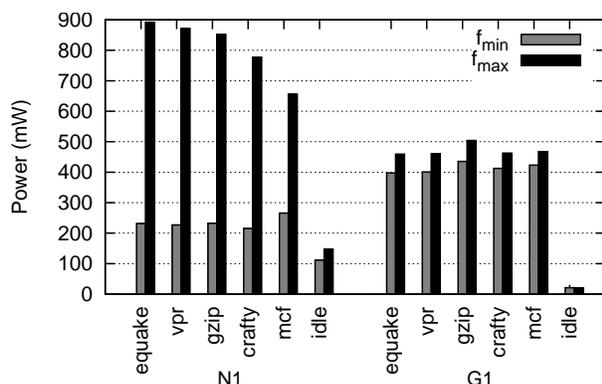|         | G1                    | N1                    |
| ------- | --------------------- | --------------------- |
| SoC     | Qualcomm MSM7201       | Qualcomm QSD 8250      |
| CPU     | ARM 11 @ 528 MHz       | ARMv7 @ 1 GHz          |
| RAM     | 192 MiB                | 512 MiB                |
| Display | 3.2" TFT, 320x480      | 3.7" OLED, 480x800     |
| Radio   | UMTS+HSPA              | UMTS+HSPA              |
| OS      | Android 1.6            | Android 2.1            |
| Kernel  | Linux 2.6.29           | Linux 2.6.29           |

Table 6: G1 and Nexus One specifications.



Figure 15: N1 and G1 system power for SPEC CPU2000 benchmarks.

|           | Performance (%) |     | Power (%) |     |
| --------- | --------------- | --- | --------- | --- |
| Benchmark | G1              | N1  | G1        | N1  |
| equake    | 67              | 25  | 87        | 26  |
| vpr       | 68              | 25  | 87        | 26  |
| gzip      | 71              | 25  | 86        | 27  |
| crafty    | 76              | 25  | 89        | 28  |
| mcf       | 84              | 54  | 91        | 41  |

Table 7: SPEC CPU2000 performance and average system power of 246 MHz relative to 384 MHz on the G1, and 245 MHz relative to 998 MHz on the N1.

## 4.3 Bluetooth

As noted earlier, we were unable to get Bluetooth working reliably on the Freerunner phone. To get an idea of Bluetooth power consumption, we re-ran the audio benchmark on the G1 with the audio output to a Bluetooth stereo headset. The power difference between this and the baseline audio benchmark should yield the consumption of the Bluetooth module, because (as shown in our Freerunner benchmarks) the power consumed by the audio subsystem is almost entirely static.

|                   | Power (mW) |           |
| ----------------- | ---------- | --------- |
| Benchmark         | Total      | Bluetooth |
| Audio baseline    | 459.7      | -         |
| Bluetooth (near)  | 495.7      | 36.0      |
| Bluetooth (far)   | 504.7      | 44.9      |

Table 8: G1 Bluetooth power under the audio benchmark.

Table 8 shows the total and estimated Bluetooth power consumption for the audio benchmarks. In the "near" benchmark, the headset was placed approximately 30 cm from the phone, and about 10 m in the "far" benchmark.

## 4.4 Benchmarks

Table 9 shows total system power consumption for the Freerunner, G1, and Nexus One for a selection of our benchmarks. The power consumption of the backlight (OLED for the N1) has been subtracted out, since it is highly dependent on the user's brightness setting. Table 10 shows the additional power consumption of the OLED display at minimum and maximum brightness levels.

The lower power consumption of the G1 in the idle, web and email benchmarks can be attributed to the excellent low-power state of its SoC and effective use of it by software. This can be seen in the SPEC benchmarks, where the idle system consumes less than 22 mW; the idle CPU power must be lower still.

The power disparity for the phone call benchmark is likely due to power consumed by the non-radio components of the system. The G1 and Nexus One phones enter a suspended state during the call, offloading all functionality to the UMTS module. In contrast, the Freerunner remains in a fully-active state throughout. The power consumption of the GSM subsystem alone (832.4 mW) is comparable to the G1 and N1 system consumption. Due to lack of freely-available documentation, it is not clear whether the Freerunner's GSM chipset lacks this feature, or if it is not supported in software.

| | Average System Power (mW) | | |
|---|---|---|---|
| Benchmark | Freerunner | G1 | N1 |
| Suspend | 103.2 | 26.6 | 24.9 |
| Idle | 333.7 | 161.2 | 333.9 |
| Phone call | 1135.4 | 822.4 | 746.8 |
| Email (cell) | 690.7 | 599.4 | - |
| Email (WiFi) | 505.6 | 349.2 | - |
| Web (cell) | 500.0 | 430.4 | 538.0 |
| Web (WiFi) | 430.4 | 270.6 | 412.2 |
| Network (cell) | 929.7 | 1016.4 | 825.9 |
| Network (WiFi) | 1053.7 | 1355.8 | 884.1 |
| Video | 558.8 | 568.3 | 526.3 |
| Audio | 419.0 | 459.7 | 322.4 |

Table 9: Freerunner, G1 and N1 system power (excluding backlight) for a number of micro- and macro-benchmarks.

## 5 Analysis

### 5.1 Where does the energy go?

Our results show that the majority of power consumption can be attributed to the GSM module and the display, including the LCD panel and touchscreen, the graphics accelerator/driver, and the backlight.

In all except the GSM-intensive benchmarks, the brightness of the backlight is the most critical factor in determining power consumption. However, this is a relatively simple device from a power-management perspective, and largely depends on the user's brightness preference. Our results confirm that aggressive backlight dimming can save a great deal of energy, and further motivates the inclusion of ambient light and proximity sensors in mobile devices to assist with selecting an appropriate brightness. Moreover, the N1 OLED results show that merely selecting a light-on-dark colour scheme can significantly reduce energy consumption.

The GSM module consumes a great deal of both static and dynamic power. Merely maintaining a connection with the network consumes a significant fraction of total power. During a phone call, GSM consumes in excess of 800 mW average, which represents the single largest power drain in any of our benchmarks. Unfortunately, a phone-call-heavy workload presents little scope for software-level power management. Dimming the backlight during a call, as Android does, is clearly good policy, saving up to 40 % power even with the large GSM consumption.

Overall, the static contribution to system power consumption is substantial. In all of our usage scenarios, except GSM phone call, static power accounts for at least 50 % of the total. If the backlight is included, this fig-

| | OLED Power (mW) | |
|---|---|---|
| Benchmark | Min. | Max. |
| Idle | 38.0 | 257.3 |
| Phone call | 16.7 | 112.9 |
| Web | 164.2 | 1111.7 |
| Video | 15.1 | 102.0 |

Table 10: Additional power consumed by the N1 OLED display at maximum and minimum brightness.

ure rises substantially. This leads us to the conclusion that the most effective power management approach on mobile devices is to shut down unused components and disable their power supplies (where possible).

The RAM, audio and flash subsystems consistently showed the lowest power consumption. While our micro-benchmarks showed that the peak power of the SD card could be substantial ($\approx 50$ mW), in practice the utilisation is low enough such that on average, negligible power is consumed. Even video playback, one of the more data-intensive uses of mobile devices, showed SD power well under 1 % of total power. RAM has similar characteristics; micro-benchmarks showed that RAM power can exceed CPU power in certain workloads, but in practical situations, CPU power overshadows RAM by a factor of two or more. Audio displayed a largely static power consumption in the range of 28–34 mW. Overall, RAM, audio and SD have little effect on the power consumption of the device, and therefore offer little potential for energy optimisation.

### 5.2 Dynamic voltage and frequency scaling

Our CPU micro-benchmarks show that dynamic voltage and frequency scaling (DVFS) can significantly reduce the power consumption of the CPU. However, this does not imply reduced energy overall, because the run-time of the workload also increases. Our results show (Table 3) that only highly memory-bound workloads (namely `mcf`) exhibit a net reduction in CPU/RAM energy.

However, such a simplistic analysis assumes that after completing the task, the device consumes zero power. Clearly this is not a realistic model, particularly for a smartphone. To correct for this, we can "pad" each of the measurements with idle power [5] in order to equalise the run times, according to the following equation:

$$E = Pt + P_{\text{idle}}\left(t_{\text{max}} - t\right)$$

where

$E$    is the equivalent energy consumed for the benchmark;

| Benchmark | % Energy | | |
|---|---|---|---|
| | Freerunner | G1 | N1 |
| equake | 95.5 | 126.0 | 75.6 |
| vpr | 95.8 | 124.5 | 75.9 |
| gzip | 95.8 | 120.1 | 77.7 |
| crafty | 95.5 | 115.6 | 77.3 |
| mcf | 94.9 | 105.3 | 65.9 |

Table 11: SPEC CPU2000 percentage total system energy consumption of the minimum frequency compared with the maximum frequency, padded with idle power.

$P$    is the average power over the run-time of the benchmark;

$t$    is the run-time of the benchmark;

$P_{\mathrm{idle}}$    is the idle power;

$t_{\mathrm{max}}$    is the maximum run-time of the benchmark over all frequencies.

Table 11 shows the energy consumed for each of the SPEC benchmarks at the lowest frequency, compared to the highest frequency, padded with idle power.

The results show that the practical benefits of DVFS depend largely on the CPU hardware (particularly idle power), and to some extent, the workload.

On the G1, which has a good low-power idle mode, reducing frequency always results in increased energy usage. It appears that DVFS on this platform is completely ineffective.

On the Freerunner, DVFS only yields a marginal energy reduction of approximately 5%—a saving of at most 20 mW. However, the N1 shows considerable advantages to using DVFS, saving up to 35%, corresponding to an average power reduction of 138 mW. Whether or not to use DVFS on these two platforms is a policy decision, since reducing frequency can affect user experience.

Much of the energy reduction on the Freerunner can be attributed to the high idle power. For a system going into suspend (rather than idle) after completing the workload, DVFS no longer offers an advantage. However, on the N1 this is not the case: DVFS is still effective, even if transitioning into a very-low power state. This is due to the processor's high efficiency at low frequencies, which can be seen in Figure 15.

In the case of an idle system, reducing frequency can result in an energy saving, and at worst has no effect. Our results show that DVFS reduces idle CPU/RAM consumption by about 30% on the Freerunner. However, in absolute terms, this is less than a 20 mW saving: 6.5% of an idle system. On the N1, this saving is approximately 36 mW. On the G1, frequency scaling during idle periods is ineffective due to the processor's low-power idle state, which is used aggressively.

## 5.3 Energy model

We can express the results of Section 3 in a scenario-based energy model of the Freerunner device, which shows the energy for each usage scenario as a function of time:

$$
\begin{aligned}
E_{\mathrm{audio}}(t) &= 0.32W \times t \\
E_{\mathrm{video}}(t) &= (0.45W + P_{\mathrm{BL}}) \times t \\
E_{\mathrm{sms}}(t) &= (0.3W + P_{\mathrm{BL}}) \times t \\
E_{\mathrm{call}}(t) &= 1.05W \times t \\
E_{\mathrm{web}}(t) &= (0.43W + P_{\mathrm{BL}}) \times t \\
E_{\mathrm{email}}(t) &= (0.61W + P_{\mathrm{BL}}) \times t
\end{aligned}
$$

The equations give the energy consumed in Joules when the time is supplied in seconds. $P_{\mathrm{BL}}$ is the backlight power (in watts), scenarios without a $P_{\mathrm{BL}}$ term are assumed to run with backlight off.

## 5.4 Modelling usage patterns

To investigate day-to-day power consumption of the device, we define a number of usage patterns. *Suspend* represents the baseline case of a device which is on standby, without placing or receiving calls or messages. The *casual* pattern represents a user who uses the phone for a small number of voice calls and text messages each day. *Regular* represents a commuter with extended time of listening to music or podcasts, combined with more lengthy or frequent phone calls, messaging and a bit of emailing. The *business* pattern features extended talking and email use together with some web browsing. Finally, the *PMD* (portable media device) case represents extensive media playback. The parameters of these patterns are summarised in Table 12. In each case, GPRS is used for data networking.

The Freerunner uses a battery of 1.2 Ah capacity, which is approximately 16 kJ. Table 13 shows the power use, and resulting battery life corresponding to the above use patterns. We assume that in all cases requiring backlight, illumination level is set at approx 66%, corresponding to 140 mW. In all other cases, backlight is assumed off.

The table shows that total battery life varies by almost a factor of 2.5 between use cases. It shows that GSM is the dominating energy drain, followed by CPU and graphics.

| Workload | SMS | Video | Audio | Phone call | Web browsing | Email |
|----------|-----|-------|-------|-----------|--------------|-------|
| Suspend  | -   | -     | -     | -         | -            | -     |
| Casual   | 15  | -     | -     | 15        | -            | -     |
| Regular  | 30  | -     | 60    | 30        | 15           | 15    |
| Business | 30  | -     | -     | 60        | 30           | 60    |
| PMD      | -   | 60    | 180   | -         | -            | -     |

Table 12: Usage patterns, showing total time for each activity in minutes.

| Workload | Power (% of total) | | | | | | | Battery life [hours] |
|----------|-----|-----|-----|----------|-----|-----------|------|----------------------|
|          | GSM | CPU | RAM | Graphics | LCD | Backlight | Rest |                      |
| Suspend  | 45  | 19  | 4   | 13       | 1   | 0         | 19   | 49                   |
| Casual   | 47  | 16  | 4   | 12       | 2   | 3         | 16   | 40                   |
| Regular  | 44  | 14  | 4   | 14       | 4   | 7         | 13   | 27                   |
| Business | 51  | 11  | 3   | 11       | 4   | 11        | 10   | 21                   |
| PMD      | 31  | 19  | 5   | 17       | 6   | 6         | 14   | 29                   |

Table 13: Daily energy use and battery life under a number of usage patterns.

## 5.5   Limitations

Our work has a number of limitations which need to be kept in mind when using our results.

The biggest one is that the Freerunner is not a latest-generation mobile phone, but is a few years old. The main feature it is lacking is a 3G cellular interface, which supports much higher data rates than the 2.5G GPRS interface. Our validation results show that this higher data rate does not appreciably affect power consumption in practical situations.

Further, the application processor is based on a relatively dated ARMv4 architecture, however it is clocked at a rate consistent with 2009-vintage smartphones. The difference in power consumption compared with more modern processors can traced largely to idle power; in other respects, the age of the CPU is not a substantial limitation.

## 6   Related Work

Mahesri and Vardhan [4] perform an analysis of power consumption on a laptop system. Their approach to component power measurement is driven partially by direct power measurement, but largely by deduction using modelling and off-line piece-wise analysis. They show that the CPU and display are the main consumers of energy for their class of system, and that other components contribute substantially only when they are used intensively. Their results mirror our observations that RAM power is insignificant in real workloads.

Bircher and John [2] look at component power estimation using modelling techniques. They demonstrate

an error of less than 9 % on average across all tested subsystems, including memory, chipset, disk, CPU, and I/O.

In a later work, Bircher and John [3] measure the power consumption of the CPU, memory controller, RAM, I/O, video and disk subsystems under a number of workloads. Their results show that CPU and disk consume the majority of the power, with the RAM and video systems consuming very little. However, under the SPEC CPU suites, they show that RAM power can indeed exceed CPU power for highly memory-bound workloads.

Sagahyroon [8] perform an analysis similar to ours on a handheld PC. They show significant consumption in the display subsystems, particularly in backlight brightness. Unlike our results, theirs suggest that the CPU, and its operating frequency, is important to overall power consumption. They also show significant dynamic power consumption in the graphics subsystems.

## 7   Conclusions and Future Work

We performed a detailed analysis of energy consumption of a smartphone, based on measurements of a physical device. We showed how the different components of the device contribute to overall power consumption. We developed a model of the energy consumption for different usage scenarios, and showed how these translate into overall energy consumption and battery life under a number of usage patterns.

The open nature of the Openmoko Neo Freerunner smartphone is what allowed us to perform such a detailed analysis and breakdown of its power consumption. This is not possible to the same degree on a typical commercial device.

We have compared the detailed measurements with a coarse-grained analysis of more modern phones, and shown the results to be comparable.

The ultimate aim of this work is to enable a systematic approach to improving power management of mobile devices. We hope that by presenting this data, we will enable such future research, both in our lab as well as by others.

## Acknowledgments

## Availability

Relevant software and data is available at `http://ertos.nicta.com.au/software/`.

## References

[1] ANDRIOD ON FREERUNNER COMMUNITY. 2009. `http://code.google.com/p/android-on-freerunner/`.

[2] BIRCHER, W. L., AND JOHN, L. K. Complete system power estimation: A trickle-down approach based on performance events. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software* (San Jose, CA, USA, Apr. 25–27 2007), IEEE Computer Society, pp. 158–168.

[3] BIRCHER, W. L., AND JOHN, L. K. Analysis of dynamic power management on multi-core processors. In *Proceedings of the 22nd International Conference on Supercomputing* (Island of Kos, Greece, June 2008), pp. 327–338.

[4] MAHESRI, A., AND VARDHAN, V. Power consumption breakdown on a modern laptop. In *Proceedings of the 2004 Workshop on Power-Aware Computer Systems* (Portland, OR, USA, Dec. 2004), B. Falsafi and T. N. Vijaykumar, Eds., vol. 3471 of *Lecture Notes in Computer Science*, Springer, pp. 165–180.

[5] MIYOSHI, A., LEFURGY, C., HENSBERGEN, E. V., RAJAMONY, R., AND RAJKUMAR, R. Critical power slope: understanding the runtime effects of frequency scaling. In *Proceedings of the 16th International Conference on Supercomputing* (New York, NY, USA, June 2002), ACM Press, pp. 35–44.

[6] NATIONAL INSTRUMENTS CORPORATION. *NI 622x Specifications*, June 2007. 371290G-01.

[7] OPENMOKO, INC. GTA02 — Neo Freerunner schematics. `http://downloads.openmoko.org/developer/schematics/GTA02`, Aug. 2008. A7RC0.

[8] SAGAHYROON, A. Power consumption in handheld computers. In *Proceedings of the International Symposium on Circuits and Systems* (Dec. 2006), pp. 1721–1724.

[9] SNOWDON, D. C., LE SUEUR, E., PETTERS, S. M., AND HEISER, G. Koala: A platform for OS-level power management. In *Proceedings of the 4th EuroSys Conference* (Nuremberg, Germany, Apr. 2009).

[10] U-BLOX AG. *ATR0630 Data Sheet*, July 2006. GPS.G4-X-06009-P2.

[11] WIKIPEDIA. Smartphone. `http://en.wikipedia.org/wiki/Smartphone`. Last visited January 2010.