Energy Management on Handheld Devices

Marc A. Viredaz, Lawrence S. Brakmo, William R. Hamburgen
HP Laboratories Palo Alto
HPL-2003-184
August 28th , 2003*

handheld computing, battery-powered devices, energy management

Handheld devices are becoming ubiquitous and, as their processing and memory capacity increases, they are starting to displace laptop computers for certain tasks - much as laptop computers have displaced desktop computers in many roles. Current handheld devices are evolving either from the organizer or PDA side or from the cellular phone side and both lines are merging rapidly. However, whatever their origin, all handheld devices share the same Achilles' heel, the battery. This is because battery technology is not improving at the same pace as the energy requirements of handheld electronics. Therefore, energy management, once in the realm of desired features, has become an important design requirement and one of the greatest challenges in portable computing, and it will remain so for a long time to come.

* Internal Accession Date Only
Approved for External Publication
To be published in ACM Queue, October 2003
© Copyright ACM 2003

1 Introduction

The disparity between energy requirements and energy sources in handheld devices is the result of several factors. First, users expect more functionality from their devices (more performance and memory, better displays, wireless capabilities, etc.), and although partially offset by improvements in low-power electronics, this increased functionality carries corresponding increases in energy consumption. Second, as a consequence of their increased functionality, these handheld devices are displacing other pieces of equipment and as a result are seeing more use between battery charges. Finally, battery technology has not improved at the same pace as the increase in energy requirements.

Among today's rechargeable batteries, lithium-ion cells offer the highest capacity. Introduced commercially by Sony in 1991, their capacity has improved by about 10 % per year in recent years [8]. But this rate of improvement is levelling off, and even with alternate materials and novel cell structures, major future improvement in rechargeable batteries is unlikely.

That is why battery suppliers and mobile system designers expect the eventual emergence of fuel cell power sources and are investing accordingly. In its most basic form, a fuel cell combines hydrogen fuel with oxygen from the surrounding air to produce water and electricity. But hydrogen combusts easily and has low density, making it unattractive for most mobile applications. Instead, most efforts are directed towards using methanol as fuel and extracting the hydrogen with catalysts or high temperature reforming [9]. While methanol-based fuel cells promise an order-of-magnitude higher energy density than chemical rechargeable batteries, serious development issues remain. Still, several manufacturers (Intermec Technologies & MTI MicroFuel Cells, NEC, and Casio) have promised initial product introductions in 2004 – 2005. While some of these early fuel cells will be used in handheld systems, most will be for laptop computers where the fuel cell's hardware complexity can be amortized over a larger system. Widespread adoption of fuel cells in portable electronics is not expected until the end of the decade. Experience has shown that users' expectations evolve along with systems' capabilities. Even the adoption of fuel cells will probably not be enough to slake mobile users thirst for energy or diminish the importance of energy management.

Although the words power and energy are often used interchangeably in casual conversation, it is very important to understand the difference between these two concepts. The power used by a device is the energy consumed per time unit. Conversely, energy is the time integral of power. Since a battery stores a given quantity of energy, the goal of energy management (unfortunately, often referred to as power management) is to minimize the amount of energy required to satisfactorily perform each task. In some cases, minimizing the power also minimizes energy, but, as will be shown, this is not always the case. Some tasks will require less energy to complete when being performed at high speed and high power for a short duration rather than at low speed and low power for a longer period of time.

However, fixed-duration tasks — such as playing audio or video — form an important class where the energy required is directly proportional to the average power consumed (since the duration of the task is, by definition, constant). This class also includes waiting, whether waiting for user input

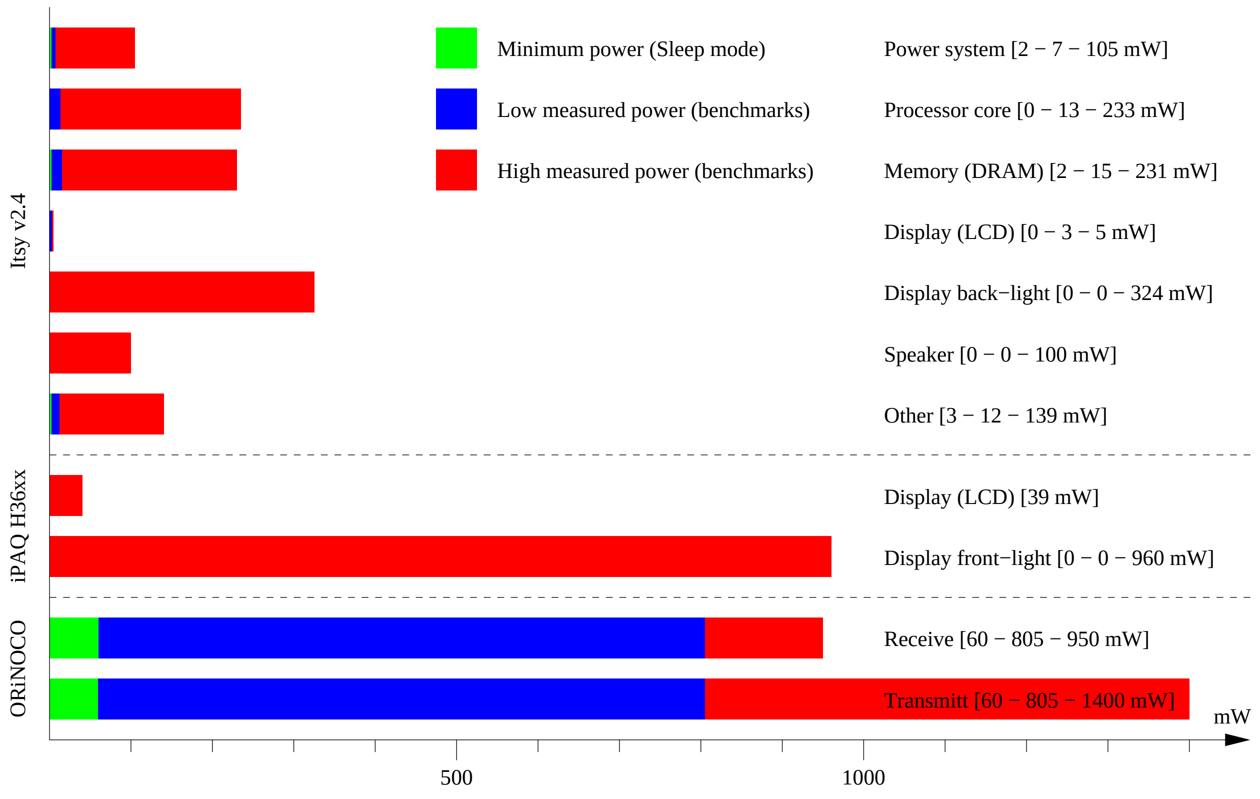

Figure 1: Minimum power (Sleep mode) and low and high power consumption of the Itsy v2.4 for a set of representative benchmarks. Display and front-light power of the Compaq iPAQ H36xx. Sleep (Doze), Idle, receiving, and transmitting power of the Proxim ORiNOCO PC Gold IEEE 802.11b wireless card.

when the device is on or keeping data in memory and the clock ticking when the device is off. For this class of tasks, focusing on minimizing power will minimize energy.

2 Energy usage in current handheld devices

In order to do useful energy management, it is important to understand how and where the energy is used in handheld devices. Unfortunately, there is little published data available. A power study [11] has been published for the Itsy [4], a prototype pocket computer developed by our team. In this study, the power usage of Itsy is broken down into its components for a variety of workloads ranging from idling to decoding and playing MPEG video. Figure 1 shows the minimum power consumed by several components or group of components (usually attained during Sleep mode) as well as the low and high power consumption figures for all the non Sleep-mode benchmarks used in this study. These power ranges are representative of real-life power consumption, since the benchmarks range from Idle mode to applications stressing the limits of most components.

In most respects, Itsy is very similar to its contemporary commercial handheld devices based on Intel's StrongARM processors. Therefore, its power usage should also be representative of these devices. However, there is one aspect where Itsy differs considerably from its commercial counterparts: the display. Itsy uses a passive-matrix gray-scale *liquid crystal display (LCD)*, optimized to be used without back-light in most lighting conditions (only under very dim lighting is the back-light necessary). On the other hand, many commercial handheld devices use active-matrix color LCDs. Although these LCDs are usually reflective or transflective and, hence, can be used with ambient light only, it has been found that most users turn the back- or front-light on, to at least low brightness, for more contrast and brighter more-saturated colors. To account for this difference, Figure 1 also presents the power for the display and front-light of the Compaq iPAQ H36xx, a device which is otherwise similar to Itsy.

Another significant difference is that Itsy did not have any built-in wireless capability, while today many handheld users want wireless communication, either as an accessory or, for the latest handheld devices, built into the system. Figure 1 also shows the power consumption of a popular wireless (IEEE 802.11b) PC Card [10], which can be used as an accessory on handheld devices such as the Compaq iPAQ H36xx.

Finally, while it would have been preferable to present data for a more recent device with a more modern processor, no similarly comprehensive data has been published. Although based on newer technology, the architecture of these devices is similar enough to Itsy's that a power breakdown should not look qualitatively much different.

3 Energy-saving techniques

The most noticeable characteristic of Figure 1 is that the minimum power values for all components are at least an order-of-magnitude smaller that the maximum ones. It suggests that the obvious technique of turning off a sub-unit when it is unused can actually be very effective. Although conceptually simple, this scheme requires a very careful analysis of the *operating system (OS)*. As our experience with Itsy showed, achieving optimal energy savings requires well structured software and a lot of careful work.

In order to achieve this goal, system hardware should be designed as a collection of inter-connected building blocks that can function independently and be independently powered on and off. Current handheld devices only partially fulfill this requirement. Peripheral building blocks (communication ports, audio input and output, display, etc.) can usually be disabled or turned off when unused. However, it is generally not possible to turn off the central building blocks, such as the processor or memories, except by putting the whole system in Sleep mode.

With the trend of putting more and more functionality into the processor, it becomes important that new processor architectures keep internal building blocks independent from a power point-of-view. It would also be useful if integrated circuit manufacturers would specify the power behavior of sub-units. Todays specifications usually provide only information on how to enable or disabled a sub-unit or how to change its parameter (speed, etc.), but not how such actions will affect the

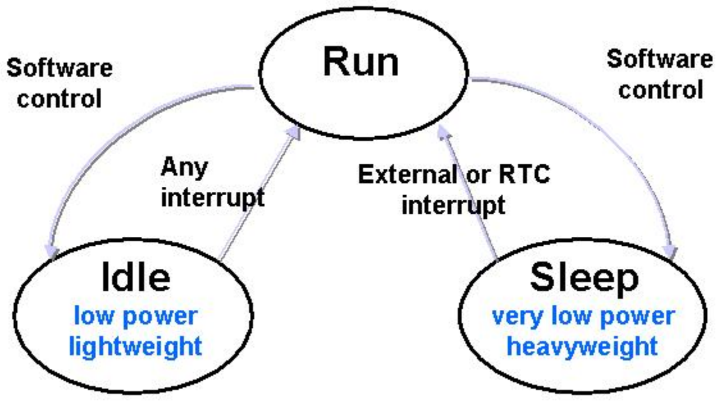

Figure 2: StrongARM SA-1100 power states.

power consumption. Without such data, software developers must resort to guesses or perform their own measurements.

Finally, one of the interesting findings of the Itsy power study is that the interplay of different sub-units with different power characteristics sometimes results in surprising and counter-intuitive behavior [11]. This suggests that real-time power measurement hardware might enable the software to make better energy-use decisions.

3.1 Processor

3.1.1 Power states

As shown in Figure 1, the processor can be responsible for a large percentage of the system's power consumption. The processors in today's handheld devices provide mechanisms to reduce their power usage by reducing the available performance. There are typically three major power states: *Run*, *Idle*, and *Sleep*. Each of the major states can itself consist of one or more minor states. For example, there may be multiple frequencies to choose from for the running state. Similarly, there may be multiple levels of the Sleep state, each with a different set of components turned off.

Figure 2 shows the major power states available on the StrongARM SA-1100, the processor used on Itsy. The Run major state has 20 minor states, two core frequencies for each of the 10 supported frequencies used by the rest of the processor. The Idle state has 10 minor states, one per supported frequency. The transitions between the Run state and the Idle or Sleep state are under software control. The transition from the Idle state to the Run state is triggered by an interrupt. The transition from the Sleep state to the Run state occurs as a result of a *real-time clock (RTC)* alarm or an external interrupt.

Three things need to be considered when looking at the energy implications of using a given processor state: (1) the performance available in this state, (2) the latency required to enter and

Table 1: Characteristics of the major power states on Itsy v2.4.

State	Performance	Hardware latencies	Electrical power
Sleep	no instructions are executed, interrupts are detected and can awake the processor	entering Sleep: 150 μs exiting Sleep: 10 – 157 ms	7.18 mW
Idle	no instructions are executed, interrupts put the processor in the Run state	entering Idle: $<$ 10 cycles exiting Idle: $<$ 10 cycles	55.5 mW at 59 MHz 81.9 mW at 133 MHz 95.5 mW at 192 MHz
Run	performance is a function of processor frequency	N/A	depends on the workload

exit this state (usually from/to the Run state), and (3) the power consumed by the processor in this state.

From a software point-of-view, the Idle state is considered a lightweight state because of the small entry and exit latencies, while the Sleep state has much larger latencies and is considered a heavy-weight state. Table 1 shows the performance, latencies and power used of Itsy for each of the StrongARM SA-1100 states.

There are a couple of things to consider when looking at Table 1. One is to note that the Sleep state has two minor states, depending on whether or not the main clock is left enabled. Turning it off saves power but increases the latency to exit the Sleep state from 10 ms to 157 ms. Another important issue is that these latencies are only hardware latencies and, in the case of the Sleep state, one should consider the sometimes significant latencies resulting from saving and restoring OS state.

3.1.2 Frequency and voltage scaling

Dynamic frequency scaling (DFS) is a technique that seeks to reduce power consumption by changing the processor frequency based on the requirements of the executing application. For fixed-duration tasks, especially waiting, this usually results in a proportional reduction of energy use. For other tasks, due to the corresponding increases in execution time, DFS could either save or waste energy depending on second-order effects. To guarantee energy savings, the processor voltage must be changed at the same time as the frequency. This technique is referred to as *dynamic voltage-frequency scaling (DVFS)*. In the literature, it is also known as *dynamic voltage scaling (DVS)*.

The importance of DVFS arises from the fact that the power consumed by a processor is proportional to the frequency and to the voltage squared:

$$P \propto f \cdot v^2$$

As a first approximation, a given task (with no waiting) takes a fixed number of processor cycles, so the energy used per task is proportional to the voltage squared.

Older processors did not support voltage scaling (although the StrongARM processor was only specified to allow frequency scaling, several experimental systems, including Itsy, used it to implement DVFS). Newer processors like the Intel's XScale processors are specified to operate at a few different voltage points, with different maximum frequencies.

DVFS and DFS algorithms strive to change the frequency without affecting the user's perception of system response time. Frequency changes usually occur at a fixed rate, for instance every 10 ms, and are based on a prediction of the processing requirements (workload) until the next decision point. Although many algorithms have been proposed [12, 3], DVFS (or even DFS) is rarely implemented in general purpose commercial handheld systems.

Instead, handheld devices that support frequency scaling have taken a more static approach, changing processor frequency only as a result of major events such as a user request or low battery condition. There are many reasons for the lack of DVFS/DFS in commercial handheld systems, including the added complexity of correctly implementing DVFS/DFS and the potential for increased system response time. There are also side effects associated with frequency changes in many current processors. These processors are highly integrated and include many I/O components, such as LCD controller and serial ports, which can misbehave during or after a frequency change. To eliminate this problem, some (like the XScale processors) can be programmed with two core frequencies, where one is a multiple of the other, and can quickly switch between the two without affecting the I/O components.

3.2 Memory

One interesting finding of the Itsy power study [11] is that the energy cost of refreshing *dynamic random-access memory (DRAM)* is small compared to the cost of accessing it. Therefore increasing the size of the memory does not result in an important energy penalty. This is, however, not the case in Sleep mode. In Sleep mode, it would be beneficial to be able to selectively power down part of the memory. Although OSes have traditionally not implemented any mechanisms for this purpose, one could compress the memory content or otherwise free memory (e.g., require that some applications save their state and exit) to take advantage of such a feature.

Any techniques to reduce the number of memory accesses would also be beneficial. Traditionally, caches have been used for this purpose. However, caches are themselves memory arrays and the energy cost of accessing a cache will increase with its size. In the future, it is possible that cache sizes for handheld processors will be optimized to reduce energy consumption rather then access time.

3.3 Display

As illustrated by Figure 1, the display back- or front-light can easily dominate over the not insignificant power used by the display itself. Even at quarter brightness, the iPAQ H36xx's front-light is a major power consumer.

Some of this lamp power becomes the photons which make up the image and, hence, some power use is unavoidable in low lighting conditions. However, with good ambient lighting, the use of a back- or front-light could be completely avoided if LCDs had better reflective characteristics.

While *cold cathode fluorescent lamps (CCFLs)* are typically used for illuminating color LCDs in handheld products, *light emitting diodes (LEDs)* are gaining a foothold. While today's white LEDs have less than half the efficiency of CCFLs, they are rapidly improving and dropping in price [5] and are being used where the LED's lower efficiency is offset by simplified light-guide optics and elimination of the CCFL's high-voltage power converter.

Independently from display hardware improvements, other energy-saving techniques might also be used. For instance, a typical back- or front-light illuminates an LCD uniformly. However, the full brightness of the light is only efficiently used for light pixels. For dark pixels, the light is mostly absorbed and the corresponding power wasted. Naehyuck Chang and his team proposed to dimming the back- or front-light while compensating by increasing the brightness and the contrast when displaying dark pictures [2]. They also proposed software techniques to reduce the power consumption of the LCD driver.

Organic light-emitting diode (OLED) displays are a promising new technology for the near future. However, since these displays are emissive, they can not make use of ambient light. However, a more pro-active form of energy management is possible for OLED displays: Subu Iyer et al. suggest darkening the parts of the screen that are not actively used by an application [7]. Although effective on laptop computers, it is unclear whether this technique would yield significant energy savings for handheld devices, where most applications use most of the screen. One could also choose window color schemes to reduce the number of bright pixels (e.g., light text over dark background).

Finally, further energy reduction could be achieved by changing the usage model for some handheld devices. A viewfinder or head-mounted display uses a tiny, display element to produce a large virtual image [6] and uses significantly less power than direct-view handheld displays.

3.4 Audio system

While microphone input uses little power, audio output — particularly using loudspeakers — can use appreciable power. In recent years, highly efficient switching (class D) amplifiers have been displacing traditional linear amplifiers on portable electronics for all but audiophile applications. The remaining power is mostly used to actually produce the sound and little can be done if speakers remain the favored audio-output interface. However, as with displays, energy could be saved by changing the usage model, in this case by using headsets.

3.5 Wireless networking

Wireless networking is becoming an important feature in current handheld devices. The most popular short-range wireless technologies are *Bluetooth (BT)* and IEEE 802.11b (or *WiFi*). Low-power BT is a short range technology, having a typical range of 10 meters, with a bandwidth of less than 1 Mbit/s. One of its main advantages is the low power consumption, on the order of 100 – 200 mW for transmission and reception, 10 – 20 mW in Idle mode, with even lower power modes (e.g., SNIFF, HOLD, or PARK) available during sporadic use.

IEEE 802.11b has a much longer range, normally about 100 meters, and a much higher bandwidth (11 Mbit/s) at the cost of much higher power consumption. For example, an IEEE 802.11b wireless PC Card was measured at about 60 mW in Sleep (Doze) mode, 805 mW in Idle mode, 950 mW while receiving, and 1400 mW while transmitting [10].

Today's most common technology for wide-area wireless networking is *General Packet Radio Service (GPRS)*. GPRS uses a mobile telephone network to send and receive information and has a theoretical maximum speed of about 170 kbit/s. Typical speeds are much less, in the range of 10 – 50 kbit/s. Higher speeds are expected in the future when *Enhanced Data rates for GSM Evolution (EDGE)* and *Universal Mobile Telephone System (3GSM)* are introduced.

Reducing energy consumption in a wireless subsystem is achieved by spending as much time as possible in low power states. However, as with processor state switching, it is important to take into account the energy cost of transitions into and out of these lower power states.

4 Current operating systems

Current operating systems usually implement three power states substantially equivalent to the Run, Idle, and Sleep states described earlier for Linux. However, they are often named differently. In particular, Windows CE 3.0 calls them *On*, *Idle* and *Suspend*; Symbian calls them *Run*, *Idle* and *Standby*; and Palm OS calls them *Run*, *Doze* and *Sleep*.

Most OSes use an Idle thread or process running at the lowest possible priority that is entered whenever there is no useful work to be done. The function of the Idle thread is to put the processor into its Idle state. The OS enters its Sleep state either through a user action such as pressing a button, or through an inactivity timer. Before the OS can put the processor into its Sleep state, it must first save some portion of the OS state, such as the value of some processor control registers. When the OS resumes, either by user action or by an RTC alarm (e.g., an appointment alarm), the OS must restore the processor registers that were saved.

OSes also support turning off individual components when they are not being used. For example, the audio subsystem could be turned off when no application is using it.

Linux is often used for prototyping thanks to its open-source model, which allows anyone to implement and explore novel energy management approaches.

5 Conclusion

As people are increasingly relying on handheld devices, good energy management is becoming necessary to squeeze the most out of a battery. A plethora of ideas have been proposed and tested by researchers in academia and in industry, but few have made their way into commercial products. The field is still in its infancy and interesting ideas continue to blossom.

Power/energy studies, which have been sadly lacking, are necessary to understand how energy is actually used. Very little data is available and more published studies are badly needed. Another requirement is better tools to track how energy is used [1], in particular the capability to evaluate power consumption in real time.

There is also a chasm in that power measurements have traditionally been in the hands of hardware engineers, while energy management is usually the responsibility of software developers. Hardware engineers need to provide energy-tracking tools that software developers can easily use. At the same time, software engineers need to acquire a better understanding of how the hardware uses energy. In many respects, the energy-management field is at the same stage as performance evaluation decades ago, when a similar gulf existed between hardware engineers who understood ”where cycles were being used” and software engineers who wrote compilers and applications. Perhaps within a few years ”energy-management engineers” will be as numerous as performance engineers are today and will help bridge this still yawning gap.

References

[1] Fay Chang, Keith I. Farkas, and Parthasarathy Ranganathan. *Energy-Driven Statistical Sampling: Detecting Software Hotspots.* In B. Falsafi and T.N. Vijaykumar (eds.), *Power-Aware Computer Systems*, vol. 2325 of *Lecture Notes in Computer Science*, pp. 110 – 129. Springer-Verlag, Berlin Heidelberg (D), 2003. Revised papers from PACS ’2002.

[2] Inseok Choi, Hojun Shim, and Naehyuck Chang. *Low-Power Color TFT LCD Display for Hand-Held Embedded Systems.* In *Proceedings of the 2002 International Symposium on Low Power Electronics and Design*, pp. 112 – 117, Monterey, CA (USA), Aug. 2002. ACM, IEEE, ACM Press.

[3] Kinshuk Govil, Edwin Chan, and Hal Wasserman. *Comparing Algorithms for Dynamic Speed-Setting of a Low-Power CPU.* In *Proceedings of the First International Conference on Mobile Computing and Networking*, pp. 13 – 25, Berkeley, CA (USA), Nov. 1995. ACM, NASA, ACM Press.

[4] William R. Hamburgen, Deborah A. Wallach, Marc A. Viredaz, Lawrence S. Brakmo, Carl A. Waldspurger, Joel F. Bartlett, Timothy Mann, and Keith I. Farkas. *Itsy: Stretching the Bounds of Mobile Computing. Computer*, 34(4):28 – 36, Apr. 2001. IEEE Computer Society.

[5] Yoshiko Hara. *White LED Lamp Market Brightens. EE Times*, July 2002.

[6] *The Fusion of PDA Portability and Laptop Utility*. White paper, Interactive Imaging Systems, Rochester, NY (USA), 2001.

[7] Subu Iyer, Lu Luo, Robert Mayo, and Parthasarathy Ranganathan. *Energy-Adaptive Display System Designs for Future Mobile Environments*. In *Proceedings of the First International Conference on Mobile Systems, Applications, and Services*, pp. 245 – 258, San Francisco, CA (USA), May 2003. ACM, USENIX.

[8] David Linden and Thomas B. Reddy. *Secondary Batteries — Introduction*. In David Linden and Thomas B. Reddy (eds.), *Handbook of Batteries*, chapter 22, pp. 22.3 – 22.24. McGraw-Hill, New York, NY (USA), 3rd edition, 2002.

[9] Michael J. Riezenman. *Mighty Mites*. *IEEE Spectrum*, 40(6):30 – 33, June 2003.

[10] Eugene Shih, Paramvir Bahl, and Michael J. Sinclair. *Wake on Wireless: An Event Driven Energy Saving Strategy for Battery Operated Devices*. In *Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking*, pp. 160 – 171, Atlanta, GA (USA), Sept. 2002. ACM, ACM Press.

[11] Marc A. Viredaz and Deborah A. Wallach. *Power Evaluation of a Handheld Computer*. *IEEE Micro*, 23(1):66 – 74, Jan. – Feb. 2003.

[12] Mark Weiser, Brent Welch, Alan Demers, and Scott Shenker. *Scheduling for Reduced CPU Energy*. In *Proceedings of the First Symposium on Operating Systems Design and Implementation*, pp. 13 – 23, Monterey, CA (USA), Nov. 1994. USENIX.