SmolPhone: a smartphone with energy limits

Joseph Paturel, Martin Quinson, Simon Rokicki

SmolPhone: a smartphone with energy limits

Joseph Paturel
Univ. Rennes, IRISA, Inria, France.

Martin Quinson
Univ. Rennes, IRISA, Inria, France

Simon Rokicki
Univ. Rennes, IRISA, Inria, France

first.last@irisa.fr

***Abstract*—This paper explores the smartphone design space to achieve energy sobriety. At its core, the proposed platform has a compute-limited but energy efficient microcontroller (MCU). Another processor can be attached to execute legacy Linux applications. Some tasks are offloaded to the Wi-Fi network controller. Our design prefers an energy-efficient variant of the 4G cellular network, which is turned off as often as possible. We are in the process of designing a hardware evaluation board to implement the vision presented in this paper.**

I. Introduction

Commercialized in 1994, the IBM Simon was the first device that blended together a mobile phone and a Personal Digital Assistant (PDA). Latter recognized as the first smartphone, it included a calendar, an address book, a notepad and emails along with voice calls. One could argue that the smartphone history since then has consisted in growing the list of features added to this initial design. For example, the connectivity has seen tremendous improvements with Wi-Fi, GPS, Bluetooth, NFC and 5G. While historical devices could not handle any video, streaming and augmented reality applications are now commonplace. Modern devices also support AI acceleration features for voice recognition or image enhancement.

In pace with the feature offering, the smartphone hardware capabilities have dramatically increased over the last few decades. The Nokia 3310, released in 2000, had 16 Mb of storage with a 100 MHz ARM7 processor. The iPhone 3GS, released in 2009, had 256 Mb of RAM and 32 Gb of storage with a 600 MHz ARM8 processor and a GPU on chip. The iPhone 14, released in 2022, has 6 Gb of RAM, a SoC encompassing 6 compute core, 5 GPU cores, a neural processing unit and an image signal processor. These increases in hardware performance naturally comes with corresponding increases in resource consumption. For example, an early Nokia phone had a carbon footprint of 13.6 kg vs. 95 kg for the iPhone6 [13] and up to 116 kg for the iPhone 14 [1].

Battery life is perhaps the only feature that has been steadily decreasing over time. Old devices could easily be used two days on a single charge while many modern phone users struggle with one charge per day, despite the increase of battery capacity over time. In terms of design goals for the existing smartphones, the battery time falls clearly behind the increased feature set. Battery life is merely optimized as a second thought after the primary development efforts, despite the risks of rebound effects inherent to this approach.

This paper explores an alternate avenue in the smart phone design space: instead of increasing the number of features possibly at the cost of a reduced battery life, we aim for an increased battery life possibly at the cost of a reduced set of available features. Increasing the time between battery charges may also improve the overall device durability, as the performance decrease of batteries due to their aging is a function of charge cycles [6].

II. Smartphones energy tradeoffs

The most power-hungry components of a smartphone are the display, the cellular networking and the CPU [5]. A typical OLED screen consumes 800 mW, a 4G modem (LTE Cat 4) consumes between 600 mW (idle) and 1200 mW (transmitting) and a Wi-Fi modem consumes from 80 mW (idle) to 110 mW (transmitting) [8]. 5G modems consume over 3000 mW while transmitting [15]. The 4G variants intended for IoT sensors such as LTE Cat M1 can be used to reduce the power usage (transmitting at 200 mW [10]), noting that their networking performance is vastly reduced compared to LTE Cat 4 (from Mbps to kbps).

Several technologies are available for low power screens. Typical OLED screens consume from 3 mW/cm^2 for a black screen to 20 mW/cm^2 at maximal brightness [5]. Electrophoretic displays (EPD) typical of e-reader devices are compeling in this context because they are bi-stable: Displaying a still image does not consume any energy while changing the content consumes 10 mW/cm^2 at 2 Hz on the screen itself [7]. The screen controller induces an idle consumption of 2 mW/cm^2 (300 mW in total for 6 in) [14]. EPDs are unfortunately difficult to drive above 3 Hz [16], hindering their usability in a smartphone. Dot-matrix memory LCDs embed memory cells in an LCD to remove the need to constantly refresh the screen, reaching a consumption of 2 µW/cm^2 for a monochrome screen or 0.1 mW/cm^2 in full color, along with fast refresh rates [7].

It is very common to offload computations to remote servers in an attempt to reduce the power consumption of the user's device [11]. Computations can either be offloaded to a cloud datacenter or closer to the user in an edge compute node. Either way, offloading proves challenging since the latency induced by the network may greatly hinder the performance. Moreover, the energy footprint of the communication often offsets any energy savings on the local CPU, even when the energy consumption of the remote computation is not taken into account.

Cloud-based executions yields better energy savings when the data is handled remotely *before* being downloaded on the

smartphone, instead of being uploaded from the smartphone to the cloud [2]. This approach is particularly adapted when the data to download needs to be filtered or transformed remotely.

III. The SmolPhone vision

Our goal is to design a "SmolPhone" device that offers many smartphone-like features within tight energy limits. The objective of this paper is not to describe a new product intended for a specific market, but rather to evaluate how a SmolPhone could be designed and built using recent technical and scientific advances.

Instead of offloading computations to remote servers for uncertain energy savings, the core of our proposal is to offload computations to other chips on the platform. Our design uses a small microcontroller (MCU) as the central compute core. Such MCUs usually only have hundreds of kb in RAM, limited computational power and no memory management unit (MMU). The lack of MMU forbids any OSes like Linux, so when the user requests a computation that requires MMU support (for instance to execute a legacy application), this computation is offloaded to a small application processor physically connected to the MCU. Our goal is to restrict the usage of the application processor to further save energy, while providing some access to applications that cannot be executed on the MCU. In some sense, this design decision extends over the classical Big-Little design [9] that sometimes offloads computation from a fast but energy hungry compute core to nearby slower but sparing cores. In comparison, we dub our design as *Tiny-Small*.

Because of the very limited performance of the central MCU, we want to leverage all computing power on the board. For example, the Wi-Fi function is typically implemented in a dedicated MCU with in-chip Wi-Fi modulation abilities such as on the ESP32 or BL602 chips. Similarly, cellular chips include an MCU that can decode mp3 (e.g., to play the ringtone), access a SD card and efficiently encode/decode audio streams. Since some dedicated chips can run arbitrary code on their MCU, our vision is to replace the large SoCs that are typical in smartphones with a federation of specialized chips. For example, the TCP stack will only be implemented in the networking chips, relieving the central MCU of all network protocol computations, and thus making it possible to implement the whole system on low-power and very limited hardware. This design borrows ideas both from the Computing Continuum paradigm (envisioning huge federations of computing nodes, from tiny microcontrollers up to the clouds [3]), and of the microservices paradigm (where applications are split in a set of interacting small software components [4]).

In addition, we plan to use an *input proxy*, a dedicated program located in the cloud whose goal is to reduce the load on the device as in [2]. For example, HTML pages accessed by the device would be rendered in the cloud *before* being downloaded, to alleviate the fact that the device's central MCU cannot host a complete HTML5 rendering engine. This mechanism could also be used to increase the accessibility of the web through other appropriate filters. The input proxy will also make it possible to shut down the network connectivity when the user is not pulling information from the Internet, as proposed in [18]. Push notifications such as incoming emails or chat messages could trigger SMS notifications from the proxy, that are conveyed on the cellular network even when the mobile data connectivity is turned off to save energy on the device. Upon reception, this notification would trigger a network activation on the device to fetch the incoming data.

We are currently designing a hardware prototype around a RP2040 dual-core microcontroller, a memory display, a Wi-Fi chip and a mechanical keyboard from Blackberry. Our design is modular: neither the application CPU nor the cellular modem are included on the board. They come instead as extension boards connected through USB or miniPCIe, so that several alternatives can be evaluated. For the application CPU we plan to use a Pi Zero processor, which is on of the smallest, lowest-power processors that can run Linux. In addition, the energy consumption of each component will be measured at 20 Hz or higher by the RP2040.

One may wonder whether an MCU is a sensible choice for the central element for the SmolPhone. First, these chips are highly optimized for energy consumption: a RP2040 consumes 100 mW per busy core at 133 MHz and can very quickly enter a sleep mode in which the power consumption is negligible [17]. In addition, modern MCUs actually exhibit performances that are comparable to ones of historical processors. A RP2040 achieves 246 CoreMark per core @133 MHz while the Pentium 1 CPU (released in 1993) achieved 281 CoreMark. The 2040 can easily be overclocked to 400 Mhz to achieve 859 CoreMark per core, while the Pentium II (1997) with its 1080 CoreMark used to run Windows 98. By comparison, the Pi Zero, which we plan to use as our application CPU for its MMU abilities, achieves 2084 CoreMark. This is why we think that this MCU will deliver the needed power to offer much more features than the IBM Simon.

The size and complexity of the needed software present challenges for our SmolPhone design. Larger and more complex systems were designed for older devices, but they usually required an inordinate development effort. In addition, embedded programming poses specific challenges due to the lack of an operating system. Recent works show the benefices of the Rust language in this context: compile-time verification partially compensates the lack of a MMU and runtime safety guards [12]. Furthermore, resolving memory management issues during the compilation results in more optimized applications, in particular when these applications perform complex data management. Our effort will also build upon many current efforts in the Rust community to provide industrial-class solutions for embedded development.

Given all the above, we believe that our proposed approach will be feasible both in terms of hardware components and software implementation. Much work remains to define and implement a *sufficient* set of smartphone features on our SmolPhone. The device will be fully open-sourced (hardware and software) to ease the exploration of the tradeoffs towards digital sobriety by ourselves and others.

References

[1] Apple. Product Environmental Report of the iPhone 14. Technical report, 2022.

[2] M. V. Barbera, S. Kosta, A. Mei, V. C. Perta, and J. Stefa. Mobile offloading in the wild: Findings and lessons learned through a real-life experiment with a new cloud-aware system. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pages 2355–2363, 2014.

[3] P. Beckman, J. Dongarra, N. Ferrier, G. Fox, T. Moore, D. Reed, and M. Beck. *Harnessing the Computing Continuum for Programming Our World*, chapter 7, pages 215–230. John Wiley & Sons, Ltd, 2020.

[4] B. Butzin, F. Golatowski, and D. Timmermann. Microservices approach for the internet of things. In *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–6, 2016.

[5] A. Carroll. *Understanding and reducing smartphone energy consumption*. PhD thesis, University of New South Wales, Sydney, Australia, 2017.

[6] M. Cordella, F. Alfieri, C. Clemm, and A. Berwald. Durability of smartphones: A technical analysis of reliability and repairability aspects. *Journal of Cleaner Production*, 286:125388, 2021.

[7] T. Grosse-Puppendahl, S. Hodges, N. Chen, J. Helmes, S. Taylor, J. Scott, J. Fromm, and D. Sweeney. Exploring the design space for energy-harvesting situated displays. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16, page 41–48, New York, NY, USA, 2016. Association for Computing Machinery.

[8] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. A close examination of performance and power characteristics of 4g lte networks. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, MobiSys '12, page 225–238, New York, NY, USA, 2012. Association for Computing Machinery.

[9] X. Li, G. Chen, and W. Wen. Energy-efficient execution for repetitive app usages on big.little architectures. In *Proceedings of the 54th Annual Design Automation Conference 2017*, DAC '17, New York, NY, USA, 2017. Association for Computing Machinery.

[10] P. Masek, M. Stusek, K. Zeman, R. Drapela, A. Ometov, and J. Hosek. Implementation of 3gpp lte cat-m1 technology in ns-3: System simulation and performance. In *2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, 2019.

[11] Q.-H. Nguyen and F. Dressler. A smartphone perspective on computation offloading—a survey. *Computer Communications*, 159:133–154, 2020.

[12] F. Nilsson and S. Lund. Abstraction layers and energy efficiency in tockos, a rust-based runtime for the internet of things. 2018.

[13] J. Suckling and J. Lee. Redefining scope: the true environmental impact of smartphones? *The International Journal of Life Cycle Assessment*, 20(8), 2015.

[14] Waveshare. 6inch e-ink display specifications. https://www.waveshare.com/6inch-HD-e-Paper-HAT.htm.

[15] D. Xu, A. Zhou, X. Zhang, G. Wang, X. Liu, C. An, Y. Shi, L. Liu, and H. Ma. Understanding operational 5g: A first measurement study on its coverage, performance and energy consumption. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '20, page 479–494, New York, NY, USA, 2020. Association for Computing Machinery.

[16] B.-R. Yang. *E-Paper Displays*. Wiley– SID Series in Display Technology. Wiley, 2022.

[17] S. Zhao, P. V. Rengasamy, H. Zhang, S. Bhuyan, N. C. Nachiappan, A. Sivasubramaniam, M. T. Kandemir, and C. Das. Understanding energy efficiency in iot app executions. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 742–755, 2019.

[18] L. Zhong, B. Wei, and M. J. Sinclair. Smert: Energy-efficient design of a multimedia messaging system for mobile devices. In *Proceedings of the 43rd Annual Design Automation Conference*, DAC '06, page 586–591, New York, NY, USA, 2006. Association for Computing Machinery.