SmolPhone: a smartphone with energy limits

Joseph Paturel, Clément Quinson, Martin Quinson, Simon Rokicki

SmolPhone: a smartphone with energy limits

Joseph Paturel
Univ. Rennes, IRISA, Inria

Clément Quinson
MotionLab ML GmbH

Martin Quinson
Univ. Rennes, IRISA, Inria

Simon Rokicki
Univ. Rennes, IRISA, Inria

***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). Later recognized as the first smartphone, it included a calendar, an address book, a notepad and an email client along with voice call support. 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 to name a few. While historical devices could not handle any video, streaming and augmented reality applications, they are now commonplace. Modern devices also support AI acceleration allowing them to implement features such as voice recognition or image enhancement on-board.

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 come with corresponding increases in resource consumption. For example, an early Nokia phone had a carbon footprint of 13.6 kg against 95 kg for the iPhone6 [15] and up to 116 kg for the iPhone 14 [2].

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 smartphone design space: instead of increasing the number of features possibly at the cost of 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 [7]. While some efforts achieve a very low energy consumption allowing battery-free calls to be made, they do not offer any features a user might expect from a smartphone-like device [17]. This solution also relies on custom bridges between their infrastructure and the overall cellular network. On the other hand of the spectrum, other approaches are based around "real" smartphone hardware and lower power consumption by limiting the interaction between the user and the applications [20].

The objective of this work is not to describe a new product intended for a specific market, but rather to evaluate how such a platform could be designed and built using recent technical and scientific advances.

In the following paper we'll first discuss the energy trade-offs that one could make in order to improve re-balance the features/sustainability of a smartphone. In Section II, we will present our vision for what we call "SmolPhone", an energy limited smartphone research platform. We will then go over the current state of the projects and our ideas for its future.

II. Smartphone Design Energy Trade-offs

The most power-hungry components of a smartphone are the display, the cellular modem and the CPU [6]. 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) [9]. 5G modems consume over 3000 mW while transmitting [19]. The 4G variants intended for IoT sensors such as LTE Cat M1 can be used to reduce the power usage (transmitting at 200 mW [12]), 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 [6]. Electrophoretic displays (EPD) typical of e-reader devices are compelling 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 [8]. The screen controller induces an idle consumption

of $2\,\mathrm{mW/cm^2}$ ($300\,\mathrm{mW}$ in total for $6\,\mathrm{in}$) [18]. EPDs are unfortunately difficult to drive above $3\,\mathrm{Hz}$ [21], 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 $2\,\mu\mathrm{W/cm^2}$ for a monochrome screen or $0.1\,\mathrm{mW/cm^2}$ in full color, along with fast refresh rates [8].

It is very common to offload computations to remote servers in an attempt to reduce the power consumption of the user's device [13]. 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 [3]. This approach is particularly adapted when the data to download needs to be filtered or transformed remotely.

III. The SmolPhone Vision

In this project, we are tackling two key challenges:

- Minimizing System Power Consumption: We are exploring the extent to which we can reduce the system's power consumption by strategically shutting down the application CPUs, peripherals and phone connectivity. Our approach involves utilizing various sensors to precisely measure the costs associated with different operations and duty cycles in real-world usage scenarios.
- Trimming Phone Functionality While Preserving Core Features: Another aspect of our ongoing prototype development involves investigating how much we can trim down the phone's functionality while still maintaining its essential features. The aim here is to create a tool that allows us to experiment with different levels of service quality degradation to determine what is deemed acceptable and what is not.

By addressing these challenges, we aim to develop innovative solutions that optimize power efficiency and user experience on mobile devices, ultimately contributing to a more sustainable and user-centric mobile technology ecosystem.

Our goal is to design a device that offers many smartphone-like features within tight energy limits, we call it "SmolPhone".

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 the use of any OSes like Linux, so when the user requests a computation that requires MMU support (for instance to execute a legacy application), it is offloaded to a more power-hungry 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 opposes the classical Big-Little architecture [11] that sometimes offloads computation from a fast but energy-consuming compute core to a nearby slower but sparing cores. In comparison, we call our design *Tiny-Small*, where applications are offloaded to high-performance cores when needed and ran on energy-efficient ones by default.

Because of the very limited performance of the central MCU, we want to leverage all the computing power that can be found around the board. For example, Wi-Fi communications are typically implemented using a dedicated chip that embeds both a Wi-Fi modem as well as a support MCU such as the Espressif ESP32 [10] or the Bouffalo Lab BL602 [16]. Similarly, cellular chips often include a processor core and subsystem that can decode mp3 files (e.g., to play ringtones), access an SD card and encode/decode audio streams efficiently. Since some dedicated chips can run arbitrary code on their MCU, our vision is to replace the large System-on-Chips that are typical in smartphones with a collection of specialized chips. For example, the TCP stack will only be implemented in the networking chips, relieving the central MCU of all network protocol handling, 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 [4]), and of the microservices paradigm (where applications are split in a set of interacting small software components [5]).

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 [3]. 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 [23]. 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, these notifications would wake up the networking support system to fetch the incoming data.

One may wonder whether an MCU is a sensible choice for the central element of the SmolPhone. First, these chips are highly optimized for energy consumption: a RP2040 consumes $100\,\mathrm{mW}$ per busy core at $133\,\mathrm{MHz}$ and can very quickly enter a sleep mode in which the power consumption is negligible [22]. In addition, modern MCUs actually present performance figures that are comparable to ones of full-blown

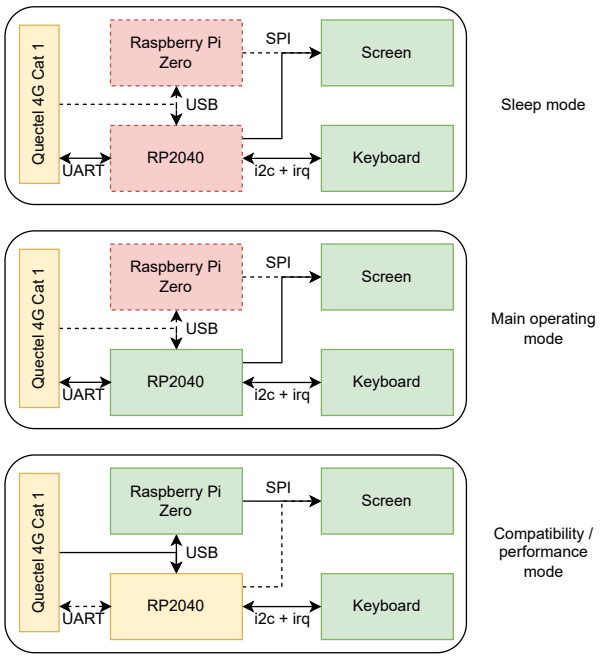

Fig. 1. First version of the SmolPhone prototype. The system has different operating modes which depend on its utilization (Sleep, Main operating, and Compatibility). For each mode, components and busses are either turned 'off' (red - dashed), or 'on' (green - plain), or used partially (yellow - plain).

application processors of years past. 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 Raspberry 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 power needed to offer many 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 benefits of the Rust language in this context: compile-time verification partially compensates the lack of a MMU and runtime safety guards [14]. Furthermore, resolving memory management issues during 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.

The device will be fully open-sourced (hardware and software) to ease the exploration of the tradeoffs toward digital sobriety by ourselves and others.

IV. Current State and Future Work

We are currently designing a hardware prototype around a RP2040 dual-core microcontroller, an LCD memory display, a Wi-Fi chip, and a mechanical keyboard from a Blackberry Q20 phone. 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 other communication busses, so that several alternatives can be evaluated. For the application CPU we plan to use a Raspberry Pi Zero processor, which is a small, low-power processor that can run Linux and Android applications in a sandbox [1]. The modem used is a Quectel EC200, a low-power 4G modem.

The system has three operating modes. In sleep mode, both the RP2040 and the Raspberry Pi Zero are turned off, waiting for a wake-up signal coming from the UI or the network. The modem only listens for GSM and can wake the system up if the cloud-based *input-proxy* sends a signal. In the main operating mode, only the RP2040 is turned on and offers the main functionality needed by the end user. The compatibility/performance mode is used when the user wants to execute a legacy application, or any task too heavy for the RP2040. The RP2040 only acts as a driver for the keyboard and the main CPU takes control of the system.

The goal is to maximize the smartphone functionalities handled by the main operating mode, reducing the active time of the Raspberry Pi Zero.

Additionally, to help us model and record the energy consumption of each feature/component of the SmolPhone, the current drawn by every major component on the board is recorded. This will allow us to correlate the use of features to their energy impact and understand what needs to be prioritized in the design of applications (does it matter if one receives their text-message notification at the very second it was received for example).

Another important aspect of any modern electronic device is security. While we can rely on Rust to ensure memory safety on the RP2040, and sandboxing for applications running on the CPU, we still need to look into the communication channels between the different board components. The interfaces used to connect the application processor to the low-power MCU should be secure in case personal tokens or any kind of credentials are exchanged between the two.

V. Conclusion

In this paper, we have introduced our vision for an energy-limited smartphone-like device: the SmolPhone. At its core, the SmolPhone philosophy is based on a power consumption conscious connection of execution layers that can be turned on and off function of the level of features an application needs. We have presented a solution that relies on firmware and OS-level mechanics, enabling an efficient energy usage and studies the benefits of trimming down the user experience. Our future work include a security survey of our solution and an integration of bio-sourced components into the hardware of the device.

References

[1] Waydroid, android in a linux container. https://waydro.id/.

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

[3] 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.

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] B. Lab. Bl602/604 product details. https://en.bouffalolab.com/product/?type=detail&id=1.

[11] 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.

[12] 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.

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

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

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

[16] E. Systems. Esp32 wi-fi and bluetooth mcu. https://www.espressif.com/en/products/socs/esp32.

[17] V. Talla, B. Kellogg, S. Gollakota, and J. R. Smith. Battery-free cellphone. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(2), jun 2017.

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

[19] 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.

[20] J. Xu, S. Zhu, A. Balasubramanian, X. Bi, and R. Shilkrot. Ultra-low-power mode for screenless mobile interaction. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, UIST '18, page 557–568, New York, NY, USA, 2018. Association for Computing Machinery.

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

[22] 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.

[23] 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.