Battery-Free Game Boy

De Winkel, Jasper; Kortbeek, Vito; Hester, Josiah; Pawelczak, Przemyslaw

DOI
10.1145/3411839

Publication date
2020

Document Version
Final published version

Published in
Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies

Battery-Free Game Boy

JASPER DE WINKEL, Delft University of Technology, The Netherlands
VITO KORTBEEK, Delft University of Technology, The Netherlands
JOSIAH HESTER, Northwestern University, USA
PRZEMYSŁAW PAWEŁCZAK, Delft University of Technology, The Netherlands

We present ENGAGE, the first battery-free, personal mobile gaming device powered by energy harvested from the gamer actions and sunlight. Our design implements a power failure resilient Nintendo Game Boy emulator that can run off-the-shelf classic Game Boy games like Tetris or Super Mario Land. This emulator is capable of intermittent operation by tracking memory usage, avoiding the need for always checkpointing all volatile memory, and decouples the game loop from user interface mechanics allowing for restoration after power failure. We build custom hardware that harvests energy from gamer button presses and sunlight, and leverages a mixed volatility memory architecture for efficient intermittent emulation of game binaries. Beyond a fun toy, our design represents the first battery-free system design for continuous user attention despite frequent power failures caused by intermittent energy harvesting. We tackle key challenges in intermittent computing for interaction including seamless displays and dynamic incentive-based gameplay for energy harvesting. This work provides a reference implementation and framework for a future of battery-free mobile gaming in a more sustainable Internet of Things.

CCS Concepts: • **Human-centered computing** → **Handheld game consoles**; • **Hardware** → **Renewable energy**; • **Computer systems organization** → **Embedded systems**;

Additional Key Words and Phrases: Energy Harvesting, Intermittent Computing, Battery-free

ACM Reference Format:
Jasper de Winkel, Vito Kortbeek, Josiah Hester, and Przemysław Pawełczak. 2020. Battery-Free Game Boy. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 3, Article 111 (September 2020), 34 pages. https://doi.org/10.1145/3411839

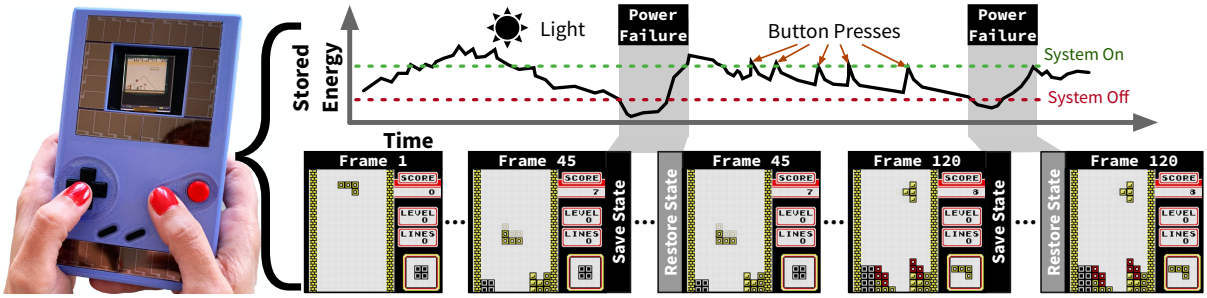


Fig. 1. Energy harvested from button presses and sunlight powers our custom handheld platform, ENGAGE, running a Nintendo Game Boy emulator which can play classic 8 bit games. ENGAGE efficiently preserves game progress despite power failures, demonstrating for the first time battery-free mobile entertainment.

Authors' addresses: Jasper de Winkel, Delft University of Technology, Delft, The Netherlands, j.dewinkel@tudelft.nl; Vito Kortbeek, Delft University of Technology, Delft, The Netherlands, v.kortbeek-1@tudelft.nl; Josiah Hester, Northwestern University, Evanston, IL, USA, josiah@northwestern.edu; Przemysław Pawełczak, Delft University of Technology, Delft, The Netherlands, p.pawelczak@tudelft.nl.

2474-9567/2020/9-ART111 $15.00
https://doi.org/10.1145/3411839

1 INTRODUCTION

This paper originates from the question: *can we game-on-the-go without batteries*? Batteries add size, weight, bulk, cost and especially inconvenience because of constant recharging—to any device. Energy can be generated by mashing buttons while gaming, and readily available energy from sunlight is all around us, so why not use this energy for battery-free mobile gaming! Significant challenges in software resiliency and efficiency, hardware operation and energy usage first need to be solved, but would represent a fundamental advancement over non-interactive (and not very fun) battery-free devices that currently exist.

Beyond the convenience and novelty factor of battery-free operation, the ecological impact of battery powered smart devices is troubling; technology advancements that reduce reliance on batteries in consumer devices would blunt the environmental impact of the trillion smart device [30, 49, 122] vision without losing the computational gains. With the emergence of low cost, small, and high-performance microcontrollers (MCUs), along with more efficient micro-energy harvesting devices that can harness the power of sunlight, motion, and heat, a new revolution in computing has come where Internet of Things devices are, in fact, increasingly leaving their batteries behind and relying only on ambient power from sunlight, motion, thermal gradients, and other modalities to power all operation [44, 89, 107].

Prototype battery-free devices have been used to make phone calls [126], deployed for machine learning [70], greenhouse monitoring [42], video streaming [92], eye tracking [73] and even built into a robot [150]. However, none of these techniques or prototypes have enabled *interactive battery-free devices*—like a smartwatch, in-place interactive display or even a **handheld video game console**. This is a critical gap in the research around battery-free devices, as these types of *reactive, interactive, and screen-focused* systems are a significant portion of the current and anticipated smart systems.

In this paper we focus specifically on this ignored part of the battery-free device ecosystem, *mobile gaming*, and use this application to elucidate the essential challenges that must be explored to get us to a future where reactive and user facing applications can also be battery-free. From a market perspective there is deep need to explore this area. The global gaming industry is massive and generates unprecedented revenues, which already exceeded 100 billion USD in 2016 [95]. Handheld console game sales constitute a large portion of the industry [95]. In terms of units sold, as of September 30, 2019 Nintendo sold 41.67 million units of its latest Switch console, since its release in March 3, 2019 [98] and these numbers continue to grow rapidly in the presence of a worldwide COVID-19 lockdown [100]. As a comparison, Nintendo's Game Boy handheld, shipped 118.69 million units since its official release in April of 1989.

To enable these types of devices, mobile gaming platforms must be re-imagined at the system and interactivity level. The main challenge is that energy harvesting is dynamic and unpredictable. This is intuitively apparent when considering a solar panel; a cloud, the time of day, weather conditions, movement and orientation of the panel, even the electrical load all change the amount of harvested energy. Because of this dynamism, these devices run out of energy and lose power frequently, only *intermittently* computing with the device having to wait seconds or minutes to gain enough energy to turn back on. This long recovery process can be energy and resource intensive, causing responsiveness delays. Worse, it can leave the game in an inconsistent state. Naturally, going through this entire re-loading process (from loading screen of a game to starting play) every time is burdensome, so just blindly replacing batteries in a game console with an energy harvester is not enough to ensure smooth game operation.

To address this challenge this paper presents a *framework of solutions based around energy-aware interactive computing* and a *reference implementation of a popular game console*—8 bit Nintendo Game boy [24, 99]—as a demonstration, see Figure 1. To reduce the unpredictability of energy harvesting, we take advantage of mechanical energy generated by "button mashing" of the console, harvesting this energy generated by actually playing a game on a handheld, and using it, along with solar panels, to power all operation. We design the system hardware

and software from the ground up to be energy-aware and reactive to changing energy situations to mitigate the issues caused by frequent power failures. Specifically we design a technique to create minimal *save games* that can be quickly created, updated, and saved to non-volatile memory before a power failure, then quickly restored once power returns. Unlike save games seen in traditional gaming systems, these capture the entire state of the system, so the player can recover from the exact point of power loss; for example mid-jump in a platform game; all this despite the device fully losing power.

Contributions. In this paper we present a practical, usable mobile gaming device, *Energy Aware Gaming* (abbreviated as *ENGAGE*). To date this is the first time full system emulation of a real world platform has been done battery-free, and the first intermittently powered interactive gaming platform. Our contributions follow:

(1) We introduce the concept of intermittently powered, battery-free mobile gaming;
(2) We develop an approach to failure resilient, memory-efficient, fast, whole system save games for interactive, display driven devices. A just-in-time differential checkpointing scheme is used based on the concept of tracking changed memory in *patches*;
(3) We develop a hardware platform as a reference implementation with a novel multi-input architecture for harvesting energy from button presses and sunlight. This device also enables any interactive-based system (not necessarily a game);
(4) As a stress test and demonstrative exercise of the promise of battery-free gaming, we use these systems and hardware to develop a full system Nintendo Game Boy emulator which plays unmodified Game Boy games despite power failures.

2 CHALLENGES

Personal, handheld gaming, has brought entertainment and fun to hundreds of millions of people in the past three decades. In the time of the COVID-19 pandemic, when so many are in lockdown, gaming is one of the activities that reduce stress and boredom [116, 118, 127]. The goal of this work is to develop the systems and hardware foundations for battery-free mobile gaming. This is motivated by two reasons: (i) the enhanced availability and usability of a platform that never needs to be recharged or plugged in—making the platform more convenient for the average user, and more accessible for everyone, and (ii) the need for alternative and sustainable forms of entertainment—a nod to the various industry consortia such as *Playing for the Planet* [108] which aim to reduce the gaming industries ecological impact. A battery-free handheld game console reduces ecological costs and disappointment, as it is always ready to be picked up and played *without needing to be recharged.*

Numerous explorations of battery-free smart devices address the calls for sustainable/carbon-neutral electronic device interaction and electronic design and computing [12, 65, 67, 85, 145] while preparing human-interactive electronics for the "post-collapse society" [131]. Other work has developed core systems [44], hardware [23, 28], and programming languages [82, 150] for serious systems focusing on solving the *intermittent computing problem* caused by energy harvesting and battery-free operation, where frequent power failures prevent a program from finishing a task (see Figure 2). In all cases the electronic device is powered by harvested energy from the environment [107] and stored in (super-)capacitors of much smaller energy density and size than batteries. None have yet explored the question of mobile handheld entertainment, going beyond the simple forms of battery-free gaming devices demonstrated commercially in early 1980's [26]. This is because making such a device is challenging due to complex system difficulties stemming from frequent power failures, listed below.

Challenge 1: Unpredictable Energy Harvesting. Environmental conditions change, this is exacerbated by mobile gaming. When players move from place to place, most forms of ambient energy change drastically (for instance, by moving from sun to shade), or increasing distance from a radio frequency power source. Without a more predictable source of power, it is hard to envision being able to play continuously without a battery.

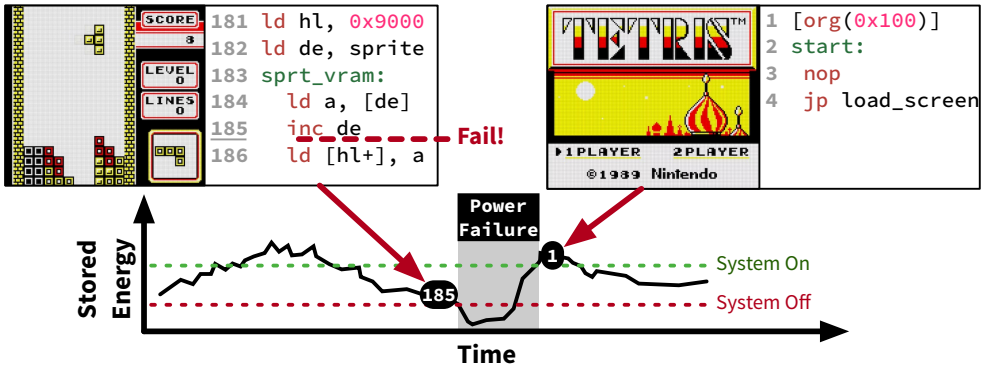


Fig. 2. Dynamic energy harvesting causes voltage fluctuations which cause frequent power failures. Shown is what would typically happen if a battery was removed from a Game Boy and replaced with solar panels. The game would play until energy is lost (i.e. at line 185) and then restart at the loading screen. *Intermittent computing techniques* seek to make it such that after the power failure, line 186 is then executed proceeding from the exact system state as before the failure.

Challenge 2: Keeping Track of System/Game State. Maintaining state of computation—let alone game state—through power failures from intermittent harvested energy is hard [77, 89]. Many software frameworks that support computation progress despite these power failures exist, saving state in non-volatile memory like FRAM and then restoring state after power resumes (see Figure 2), such as TICS [68], TotalRecall [144], and many others. Most systems trade memory efficiency for performance, this approach is opposite of that needed for gaming, where a display buffer and numerous sprites and large game state variables must be saved, requiring high memory efficiency.

Challenge 3: Enormous Variability of Games. These previous issues are compounded by the huge variability of games, both in terms of memory size, number of sprites, actions, difficulty, and even number of button presses per second. Each game is unique, and could pose difficulties when creating a general battery-free solution.

Challenge 4: Gaming's High Computational Load. To date, no full system emulation of any complex system has been attempted on battery-free, intermittently computing devices. Games and gaming platforms require more performant processors even when running natively—when running in emulation, this is compounded. All existing popular runtimes for intermittent computing are based on Texas Instrument's mixed-memory MSP430 MCU [130], which is order of magnitudes slower than the fastest ARM MCU on the market. To meet the high computational load of games, a practical runtime for ARM microconrollers must first be built.

Challenge 5: Capturing User Actions. Playing a video game means a system needs to be highly reactive: button presses post immediate updates to a screen. But *none of the existing battery-free interactive devices demonstrated this level of interactive complexity* with continuously reacting buttons and instantly-refreshing screen. Only simple touch button interfaces that do not need constant pressing for interaction are demonstrated with electronic screens that usually present static content not informed by user actions. Guaranteeing reactivity with unpredictable energy is difficult.

Challenge 6: Realistic Demonstration. The over-arching goal is to play a real, unmodified, video game on a battery-free console that everyone around the world knows (like Tetris)—in other words to be able to execute preexisting game code (or any existing code for that matter), *not to design a custom game* only to demonstrate the

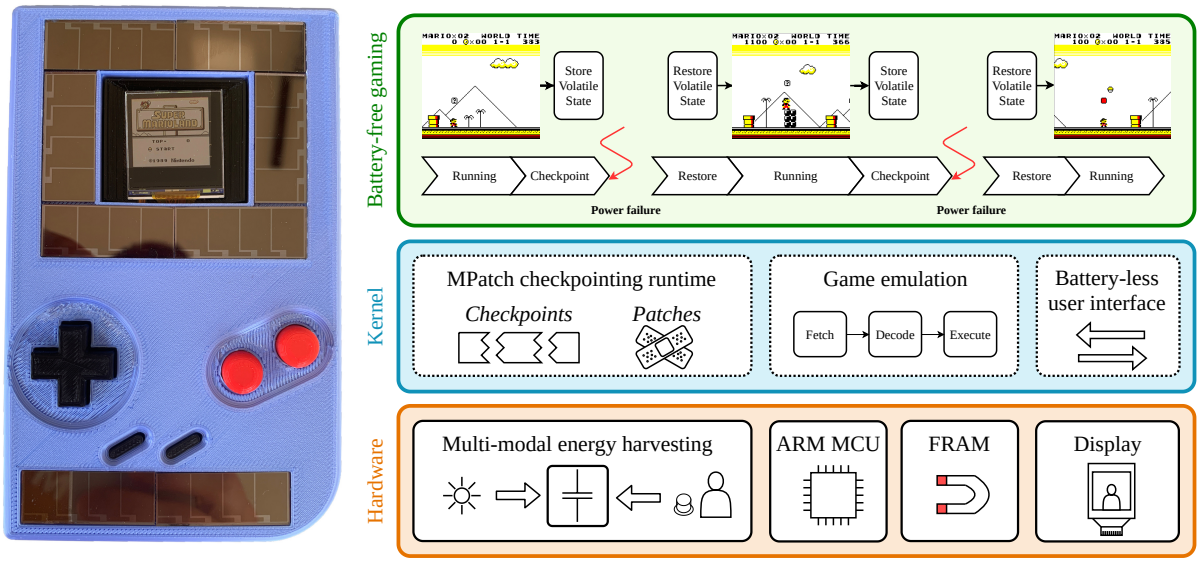


Fig. 3. ENGAGE hardware platform (left) and its internal architecture (right).

potential of battery-free gaming (we refer to extensive survey on this topic in Section 7). This could be possible only when all above challenges are addressed.

To tackle above *Challenge 1–6* we took one of the most popular gaming consoles of all-time [98]—original 8 bit Nintendo Game Boy [24, 99]—and redesigned its hardware-software, powering gameplay from the solar panels and button presses of the user, building the first ARM based intermittent computing hadware and runtime system, and doing the first full system emulation of a real world platform (Nintendo Game Boy) with intermittent computing techniques.

3 BATTERY-FREE HANDHELD GAMING

We designed the *Energy Aware Gaming* (ENGAGE) platform as a proof by demonstration that the discussed challenges could be overcome. The design and architecture of the ENGAGE platforms is shown in Figure 3. The *ENGAGE hardware* is the size and form factor of a Nintendo Game Boy, it is built around (i) user input via mechanical energy harvesting buttons (on the A, B, and D-Pad of the original Game Boy), (ii) a display, (iii) a slot for Game Boy game cartridges to be inserted, and (iv) energy harvesting circuitry from solar cells and the buttons which store energy in a small internal capacitor. The *ENGAGE kernel* consists of (i) a patch-based differential checkpointing system (denoted as MPatch) which handles low level memory movement and automatically saves and restores state of the entire system by efficiently moving necessary data to non-volatile memory (FRAM) and back (SRAM), and (ii) an extensively rewritten full-system Nintendo Game Boy emulator, which can run unmodified Game Boy games. ENGAGE is the first full system emulation, and the first gaming platform built for battery-free, energy harvesting, intermittently powered computing devices.

Usage and Impact. We intend to release the hardware designs, firmware and software as open-source, living repositories on Github [1]. We target a broad audience with our platform. Our battery-free platform (i) must be *open-source*—allowing hobbyists to expand it and improve it, including developing bespoke gaming libraries

completely separate from the Game Boy emulator, (ii) have comparable *gameplay and feel* as the original Nintendo Game Boy, and (iii) be able to play any popular retro game.

3.1 Key Ideas

Existing handheld gaming devices rely on large batteries because they need continuous high power to support high compute load, energy cost, and reactivity. We want to enable playing retro 8 bit console games, such as *Tetris* and *Super Mario Land*, on a battery-free console that is similar in user interface and gameplay to the original Nintendo Game Boy. Removing the battery and only using harvested energy causes intermittent operation, which leads to the challenges discussed in Section 2. The ENGAGE platform design navigates these challenges based on four key ideas.

Gather Energy from Gaming Actions and the Environment. ENGAGE harvests energy from button pressing/mashing and solar panels facing towards the player. As opposed to other techniques that rely on nearby dedicated wireless power generation, this approach allows for truly mobile gaming; anywhere a player can find light to see the screen and press/mash buttons. By pairing energy generation with the game actions, and the natural environment where gaming happens, ENGAGE overcomes challenges stemming from unpredictability of energy harvesting, and also ensures that energy is more likely to be available when a user initiates an action (since actions generate energy). This *addresses* **Challenge 1** and **Challenge 5**.

Track and Checkpoint Minimal State at the System Level. We must handle intermittent power failures to maintain state of play. Unfortunately, in games large amounts of memory is moved back and forth to the display, often in the form of sprites, with computation happening in between. Naively checkpointing the entire system state would be impractical, significantly increasing latency of operation. We note that while large memory movements happen, the changes in these memories are often small, meaning we can reduce checkpoints to only the changed memory, save that state just in time before a power failure, and then restore that state and resume game play. This *addresses* **Challenge 2**, **Challenge 3** and **Challenge 4**.

Use Processor Emulation to Play Retro Games. While ENGAGE could be used for custom gaming libraries made specifically for intermittent operation, the more challenging and interesting problem is full system emulation enabling the play of the thousands of existing games, and even home-brewed games. This also allows us to explore and understand the variability of real world games. This *addresses* **Challenge 3** and **Challenge 6**.

Speedup Intermittent Computing. We embrace ultra low powered, high performance ARM Cortex microcontrollers, and external FRAM memory to speed up computation. While a seemingly trivial technology advancement, with this approach we increase compute speed but increase our I/O burden for checkpointing, as the traditional MSP430 FRAM-enabled MCUs have internal FRAM memory accessible at CPU speeds. This is a different tradeoff space than any other intermittent hardware platform [23, 28, 43]. This *addresses* **Challenge 2** and **Challenge 4**.

3.2 ENGAGE Full System Nintendo Game Boy Emulator

A key part of our approach is running a full system emulation on ENGAGE hardware. To be able to run Nintendo Game Boy games an emulator is used to emulate the instruction set of the Game Boy processor, i.e. 8 bit 4.19 MHz custom-built Sharp LR35902 MCU with a processor closely based on the Z80 instruction set [24]. An emulator reads bitcode instructions and executes them in native code, mimicking the emulated CPU as closely as possible to ensure it executes in an identical fashion to the emulated CPU. With the restrictions of battery-free systems additional scenarios are introduced that normally do not exist, such as the loss of power while running a game and then attempting to restore the system to the state it lost power. Additionally emulation efficiency is of critical importance in regards of power consumption.

Table 1. Measurement statistics of all button pressing during a regular Game Boy game for four example games, showing variation between games depending on the type of game. Data was extracted by logging key presses during game play on a Game Boy emulator running on a stationary personal computer. Three similar three-minute playthroughs are averaged in the presented results.

Game name	Time between button presses (second)			Button presses per second
	Max	*Mean*	*Variance*	
Tetris	3.230	0.508	0.169	1.981
Space Invaders	3.542	0.372	0.129	2.715
Super Mario Land	12.46	0.652	1.091	1.543
Bomberman	7.534	0.765	0.762	1.313

The emulator allocates non-volatile and volatile Game Boy game memory within the memory space of ENGAGE, removing the need to keep cartridges continuously powered. Only upon loading a new game is the cartridge interface used to retrieve the non-volatile game data.

The process of emulating also requires emulation of the Game Boy I/O for the user to be able to interact with the device, most importantly the buttons and the screen. Changing behavior regarding interaction with the I/O might have an influence on the user experience and interaction with the device.

Emulating Button I/O. As energy can be harvested from the press of a button, the frequency of button presses determines the amount of energy generated. This button press frequency is game dependent: in-game-cut-scenes usually require no game-user interaction through the buttons compared to games like *Space Invaders* where buttons are continuously pressed. As a proof, in Table 1 we present statistics about the time between button presses in four popular Nintendo Game Boy games. The maximum time varies greatly depending on the presence of cut-scenes in the game. *Tetris* and *Space Invaders* have few, or short, cut scenes, and thus have a lower maximum time between presses and lower variance. On the other hand, *Super Mario Land* has an animation upon death and at the end of a level, and *Bomberman* has several cut-scenes, hence the higher maximum time between between presses and greater inter-press time variance.

This simple experiment shows that the maximum time between button presses directly pertains to the required energy buffer size, where button mashing games could run on smaller energy buffers increasing the reactivity of the platform by reducing the required charge time.

By changing emulator behavior of handling buttons more energy can be generated. To generate more energy we prevented the holding of buttons. For example, in *Super Mario Land* game it is common to hold the right button to keep moving, but in *Space Invaders* this is less common. This results in a trade-off space between energy generation and changes to the user interaction with Game Boy[1]. Finding the optimum between energy harvesting without changing the user interaction is therefore game specific. In Section 4 we further elaborate on our design choices regarding button emulation.

Emulating Screen Writes. The original Nintendo Game Boy does not employ a frame buffer. Instead, it uses a tile-based approach where tiles are rendered in a CRT-like fashion on the screen to save memory. This means when power to the system is lost, the state of the screen can not be restored since knowledge of the full screen state is not maintained. To combat this we employ a frame buffer and map the Game Boy tile-based rendering

[1]We note that the original Game Boy handles I/O through polling of I/O registers, meaning that every game can have a different handling of the I/O all together, since the Game Boy directly executes machine code from the game.

into this frame buffer. The buffer is then checkpointed to be able to restore the state of the screen upon power failure.

3.3 Gaming Through Power Failures

ENGAGE is protected from the loss of progress by the custom-designed runtime that guarantees data consistency despite power interrupts. The goal of this runtime is to save (i.e. to checkpoint) the current state of the emulator. This entails the current volatile memory content and the registers of both the host processor and the emulated system. Doing this will allow the system to continue execution from this point as if a power failure never happened.

There are multiple intermittent runtime systems (all of them are summarized in Section 7) which can be broadly divided into two classes: (i) those that use a *special (C program) code instrumentation* to guarantee correctness of computation despite power interrupts and (ii) those that use a *special version of the checkpointing*, of which a subset is designed for systems that use volatile memory—such as SRAM—as their main memory, and use a separate non-volatile memory that contains the checkpointed data. While designing ENGAGE we chose to use a checkpoint based system to allow emulation of arbitrary game code. We did not consider task-based runtimes simply because they are too complex to comprehend by a programmer and more difficult to design than a checkpoint-based system, see related discussion on this topic in [68]. But first and foremost, task-based system cannot execute a binary (machine) code, which ENGAGE is mostly executing.

The main requirement for ENGAGE is responsiveness, hence the checkpointing system needs to be as light-weight as possible. Naturally all of the checkpoint systems have some overhead, so when searching for a good solution we would like to minimize checkpoint size as much as possible—resulting in minimum overhead from data restoration. Checkpointing the entire system state, including game, and emulator, would be impossible. One core idea, proposed first by the DICE runtime [3], is *to checkpoint only parts of device memory that have been changed since the last checkpoint.* To check whether this idea applies to battery-free handheld gaming system we have performed a simple experiment. For four example Nintendo Game Boy games: (i) *Tetris*, (ii) *Space Invaders*, (iii) *Super Mario Land*, and (iii) *Bomberman*, we have measured to which memory regions of an MCU each game was writing to during one minute of game play. The result is presented in Figure 4. Indeed, we see that memory writes are very unevenly distributed for each game, hinting that such approach, which we broadly denote as *differential* checkpointing, is well suited for our ENGAGE needs.

The checkpoint runtimes, including differential ones, can be further divided into two unique classes: (i) *corruptible* and (ii) *incorruptible.*

- **Corruptible Checkpoint:** Such systems copy the current state of the MCU (memory, registers, etc.) to a predetermined location in non-volatile memory. This location is the same every time, as this eases the runtime development and reduces the non-volatile memory requirements. However, it is required that a checkpoint operation must guarantee to complete, otherwise part of the previous checkpoint may be overwritten with the current checkpoint[2]. Often these corruptible runtimes include a check whether a checkpoint was completed successfully, otherwise they start the program execution from the beginning. Such systems require exact prediction of the energy (required to perform a checkpoint) and the energy currently consumed by the complete system (to be able to guarantee that a checkpoint is only performed when its completion can be guaranteed). Such a requirement is *unrealistic* for a computing platform, such as ENGAGE, that includes many peripherals and components all connected to the same energy buffer, as correctly predicting the required energy—even the CPU alone—is difficult;

[2]If the system where to run out of power during the creation of a checkpoint, with the next checkpoint restoration a corrupt state will be restored leading to undefined behaviour—thus to a corrupt system.

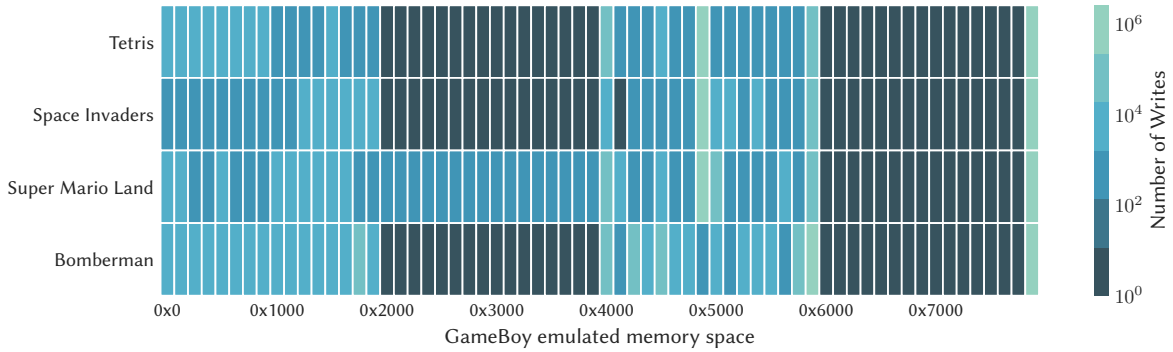


Fig. 4. Memory writes heat map of four popular 8 bit Game Boy games for one minute of play. Writes tend to cluster in a few large regions; tracking and checkpointing these regions would allow for performant intermittent execution. Note the log-scale of the number of writes.

- **Incorruptible Checkpoint:** Such systems take a different approach: at all times they *guarantee* that there is a valid checkpoint which can be restored. This means that a new checkpoint will never overwrite part of the previous checkpoint in non-volatile memory. Such guarantee is often implemented through double-buffering.

As of now, there are *no known incorruptible differential checkpoint systems*, and just one corruptible differential checkpoint system, DICE [3], refer also to Table 2 where existing intermittent runtimes are qualitatively compared from ENGAGE requirements point of view. Therefore, to realize a working ENGAGE we developed a new checkpointing runtime, denoted as MPatch, that performs incorruptible differential checkpoints. The proposed runtime is aided by a new concept of *patch checkpointing*, discussed below.

MPatch—a Patch Checkpointing Intermittent Runtime. Memory is constantly being modified during the execution of a program. However, as Figure 4 clearly illustrates, it is unlikely that during an on-period of any intermittently powered embedded system, including ENGAGE, all memory is modified. Therefore, when creating a checkpoint containing all the known or active memory regions of the system, one will inevitably copy memory locations that have not changed since the last checkpoint.

It is thus desirable to copy as little of the (embedded) system state as possible while keeping the checkpointing incorruptible. The most fundamental method to do this efficiently is to track which memory regions have been changed since the last checkpoint, in other words, to see memory modification *difference* in-between checkpoints. As mentioned earlier, the only checkpoint runtime that has employed this form of differential checkpoint so far was DICE [3], see again Table 2. It is, however, difficult to apply the techniques used by DICE while maintaining an incorruptible system (that uses double buffering). Specifically, assuming that only one of the buffers is active, if part of the checkpoint resides in the previous buffer, and yet another checkpoint occurs, then it is impossible to keep the incorruptibility trait with DICE without still copying all checkpoint data between the two buffers. Therefore, to achieve differential checkpointing that is incorruptible, a new system has to be designed, which resulted in MPatch.

MPatch Just-in-Time Checkpoints. As we have shown in Figure 4 not all of the emulator and display memory is written to at every MCU clock cycle. Hence we only checkpoint the modified memory regions, which we denote as *patches*. Then, we monitor the voltage level of the storage capacitor, as in other existing runtimes,

Table 2. Comparison of MPatch with state-of-the-art intermittent checkpointing runtimes.

System	Incorruptible	Differential	Just-in-time	Volatile main memory	ARM support
Mementos [109]	Yes ✓	No ✗	Yes ✓	Yes ✓	Yes ✓
Hibernus++ [8]	No[1] ✗	No ✗	Partially —	Yes ✓	No ✗
QuickRecall [57]	No ✗	No ✗	Yes ✓	No ✗	No ✗
Chinchilla [82]	Yes ✓	N/A	No ✗	No ✗	No ✗
Rachet [137]	Yes ✓	No ✗	No ✗	Yes ✓	Yes ✓
HarvOS [11]	No[1] ✗	No ✗	Yes ✓	Yes ✓	Yes ✓
TICS [68]	Yes ✓	N/A	No ✗	No ✗	No ✗
TotalRecall [144]	No[1] ✗	No ✗	Yes ✓	Yes ✓	No ✗
Elastin [19]	Yes ✓	N/A	No ✗	No ✗	No ✗
WhatsNext [35]	Yes ✓	No ✗	No ✗	Yes ✓	Yes ✓
DICE [3]	No [1] ✗	Yes ✓	Yes ✓	Yes ✓	Yes ✓
MPatch	Yes ✓	Yes ✓	Yes ✓	Yes ✓	Yes ✓

[1] These systems require perfect energy prediction to not get corrupted. Any changes in, for example, capacitor size [19], power consumption due to peripheral use, or harvested energy, **will** lead to incorrect predictions and therefore **corruption**.

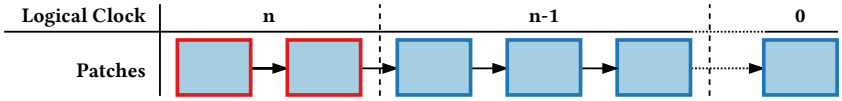


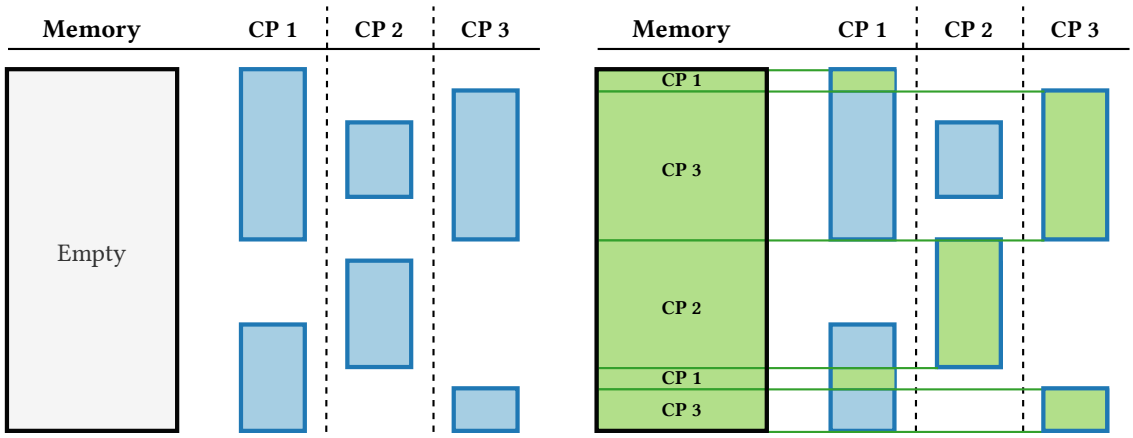
Fig. 5. MPatch stage operation. Patches outlined with red are staged, but not committed. Patches outlined with blue signify committed patches.

e.g. [3, 8, 57, 144] and only checkpoint the state when nearing a power failure—we call this *just-in-time checkpoint*. We purposefully do not perform checkpoints at an interval timer: game players are susceptible to lagging in a game, hence interval-based checkpointing (which introduces frequent fixed-interval delay) is not desirable.

Patch Handling. A patch is a non-volatile copy of a *consecutive* region of volatile memory that has changed since the last successfully created checkpoint. As different memory regions are modified during execution, multiple patches of different memory sections might be required for a complete checkpoint. During the restoration, the most recent patches (in combination with the pre-existing patches) are used to restore the volatile memory to the state it was in during the last checkpoint. By only storing the modified regions the checkpoint time is significantly reduced, as often only a small part of the memory is changed between the two consecutive checkpoints (we will investigate this further in Section 5).

As with traditional checkpoint-based systems that use double-buffering, an atomic variable n, determines which of the two buffers should be used to restore the system in case of a power failure [68, 109]. This variable n is changed—often incremented—to mark the completion of a checkpoint. The requirement on n is that for its increment, $n+1$, it holds that $(n \bmod 2) \neq (n+1 \bmod 2)$. MPatch patch management is also built around the atomic variable. However, MPatch extends the function of this variable to act as a *logical clock*, with the additional requirement that $n \neq n+1$.

We now define three fundamental patch operations (i) *Patch Stage*, (ii) *Patch Commit*, and (iii) *Patch Restore*.



(a) MPatch memory **before** restoration.

(b) MPatch memory **after** restoration. The restore sequence applies only the parts of patches that are required to reconstruct the memory.

Fig. 6. MPatch patch restore procedure after three successful checkpoints (CP). For the ease of illustration, we assume that the memory is initiated as empty; blue rectangles depict patches that have been successfully committed to non-volatile memory and green rectangles signify the parts of the patches that are applied during restoration.

- *Patch Stage:* When a patch is created, the required amount of non-volatile memory is allocated and the volatile-memory is copied to the patch. Next, the patch is *staged* by signing it with the current logical clock n added to the front of the *patch chain*, i.e. the list of patches, ordered from newest to oldest, that will be applied during restoration. Staged patches are outlined in red color in Figure 5. While a patch is staged it will be discarded if a power failure (and thus a restoration procedure) occurs.
- *Patch Commit:* When the logical clock n is incremented, all previously staged patches will become *committed*. These patches are outlined in blue in Figure 5. Committed patches will be considered during the *patch restore* procedure.
- *Patch Restore:* When ENGAGE inevitably fails due to a lack of energy, it should be restored to the last completed checkpoint. Patches hold copies of consecutive volatile memory regions and are linked together to form the patch chain. This moves the complication of deciding what *part* of the patch to apply, if any, to the restore operation. To reconstruct the state of the most recent checkpoint the (partial) content of multiple patches has to be combined. This reconstruction, due to the implicit ordering in the patch chain, starts from newest to oldest. For each patch, only the parts that were not already applied during the current restore operation are copied to volatile memory, as illustrated in Figure 6. In contrast, for a traditional incorruptible checkpoint runtime, restoring a checkpoint means reading the logical clock n and copying the checkpoint content from the selected buffer to the corresponding volatile memory and registers.

4 ENGAGE IMPLEMENTATION

We proceed with the implementation details of ENGAGE, following from the design description provided in Section 3. All hardware, software and tools, as well as documentation for ENGAGE are publicly available via [1].

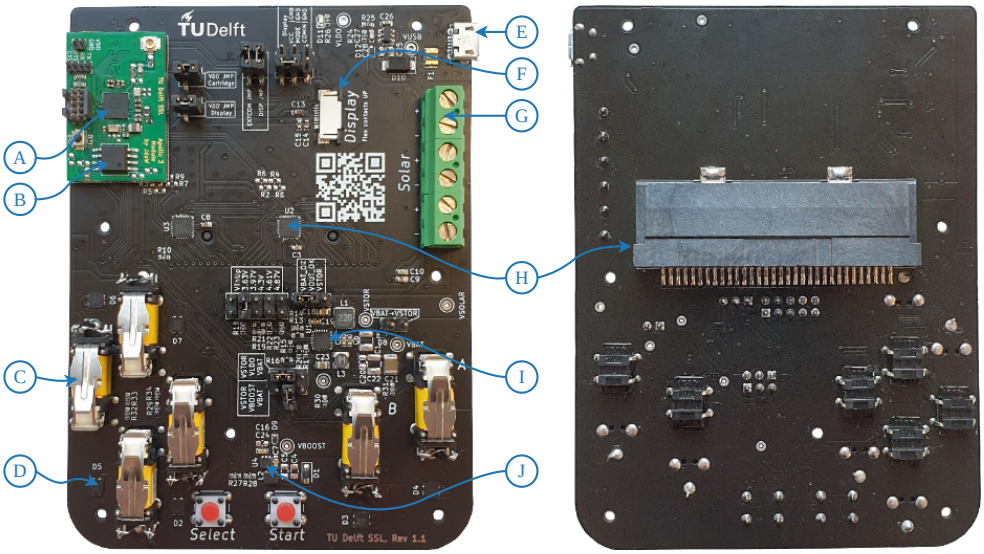


Fig. 7. Energy Aware Gaming (ENGAGE) fabrication details. The main components are: (A) Ambiq Apollo3 Blue ARM Cortex-M4 MCU, (B) Fujitsu MB85RS4MT 512 KB FRAM, (C) ZF AFIG-0007 energy harvesting switch, (D) Semiconductor Components Industries NSR1030QMUTWG low forward voltage diode bridge, (E) micro USB debugging port, (F) display connector, (G) solar panels connector, (H) cartridge interface, (I) Texas Instruments BQ25570 harvester/power management chip, and (J) Texas Instruments TPS61099 boost converter.

4.1 ENGAGE Hardware

We built a handheld, energy harvesting, battery-free hardware platform to enable the development and testing of our approach to battery-free mobile gaming. ENGAGE is built using the following components.

4.1.1 Processing and Memory. Stemming from the requirements (Section 2), for compatibility and popularity reasons, we build our ENGAGE around an ARM MCU architecture. However, none of the ARM architecture MCUs we are aware of contains on-chip fast, byte-addressable non-volatile memory—such as FRAM—serving as a main memory. Only slow and energy-expensive FLASH memory is present. Therefore we equip our battery-free console with external dedicated FRAM. Central to ENGAGE is the *Ambiq Apollo3 Blue ARM Cortex-M4 MCU* operating at a clock frequency of 96 MHz [5], chosen for its good energy efficiency. The Apollo3 runs the Game Boy emulator and MPatch software. External *Fujitsu MB85RS4MT 512 KB FRAM* [34] is connected through SPI to the MCU providing a fast and durable method of non-volatile storage for patch checkpoints, see Figure 7. With the availability of power switches within the MCU, it gates power to the screen and cartridge interface, see Figure 7.

To be able to read game cartridges, a cartridge connector is placed on the back of the ENGAGE platform. The cartridge interfaces with the MCU using *Semtech SX1503 I/O expanders* [120], in this case the extenders also translate the 3 V system voltage to 5 V logic required by the cartridge.

4.1.2 Mixed Source Energy Harvesting. We extract energy from two sources: (i) button presses of a regular Game Boy, using mechanical off-the-shelf button press harvesters, and (ii) a set of solar panels attached to the front of the Game Boy chassis. The selected buttons were *ZF AFIG-0007 energy harvesting switches* [153]. Six of these kinetic harvesting switches are used (see Figure 7) located at the position of the D-pad (four switches) and “A”,

"B" operation buttons (two switches). The energy harvesting switches generate energy by moving a magnet inside a coil. Since both up and downward motion generates energy, the output of the switches is rectified using a *Semiconductor Components Industries NSR1030QMUTWG low forward voltage diode bridge* [119] before being boosted using a *Texas Instruments TPS61099 boost converter* [129] to be stored in a small intermediate energy storage capacitor. When the intermediate energy storage reaches 4 V the system turns on and the buck converter of the power management chip steps down the voltage to 3 V to power the system. Additionally solar energy is harvested from eight small solar panels [103], each measuring 35.0×13.9 mm, affixed on the front of the ENGAGE chassis (see Figure 7).

Harvesting of solar energy is managed by a *Texas Instruments BQ25570 harvester/power management chip* [128], integrating both a buck and a boost converter. The harvester employs a boost converter and maximum power point tracking to harvest the solar energy and stores the harvested energy in the intermediate energy storage capacitor. All energy from energy harvesting is stored in a main 3.3 mF capacitor, chosen to enable even in the worst energy harvesting conditions a few seconds of game play before the system reaches a critical voltage and powers down to harvest more energy.

4.1.3 Ultra-low Power Display. Displaying game content consumes the most energy of any embedded platform. Making energy-efficient display is research on its own, which is beyond the scope of this work. At the same time, we want ENGAGE to be accessible to any hobbyist by being built out of easily available and inexpensive components. Therefore we have relied on super-low power state-of-the-art off-the-shelf commercial display. Note that we have excluded e-ink displays as their refresh rates are too low for a good gaming experience (especially with rapidly changing game states such as *Super Mario Land*).

We have chosen a low power non-back lit reflective *Japan Display LPM013M126A LCD* [56], noting that the Game Boy also does not have a backlight. The LCD measures 26.02×27.82 mm, which means the display is smaller compared to the original Game Boy screen by 47×43 mm. Our chosen display offers a greater resolution of 176×176 compared to 160×144 pixels of the original Nintendo Game Boy and is capable of displaying eight colors compared to the four shades of gray the Game Boy uses. Just like in case of cartridge interface, the MCU gates power to the screen.

4.1.4 Form Factor and Fabrication. For the same gaming feel we encapsulate the electronics of ENGAGE in a 3D-printed chassis reminiscent of the original Game Boy. The only differences are: (i) the removal of sound outlet, as *we do not support sound generation on an intermittent power supply*, (ii) addition of slits to house the solar panels, (iii) slit for the USB port to provide constant power supply to ENGAGE for debugging, and (iv) removal of slits for battery charge cable and an on/off switch—as they are obviously *not needed in a battery-free system*. Since the Apollo3 Blue MCU is only available in BGA packaging, we separated the MCU from the main PCB creating a separate module containing this MCU only—see the green PCB in the top left corner of Figure 7—to reduce the risk of soldering errors during small batch manufacturing. This module is connected through connectors to the main ENGAGE PCB. Complete fabricated PCB, front and back, are shown in Figure 7.

4.2 ENGAGE Emulator Implementation

As many Nintendo Game Boy emulators have already been written we have decided not to build yet another one and relied on the existing emulator implementation that targets a different MCU. Specifically, to run with ENGAGE we extensively modified and rewrote a pre-existing freely-available implementation of original Nintendo Game Boy emulator targeting a STM32F7 MCU [10]. All the modifications to this emulator, enabling to reproduce our work, are part of our open-source repository freely available to download from [1].

ENGAGE Screen Handling. Due to the availability of displaying colours on the chosen display we remapped the default gray-scale colour palette (white, light gray, dark gray and black) of the Game Boy to a colour version

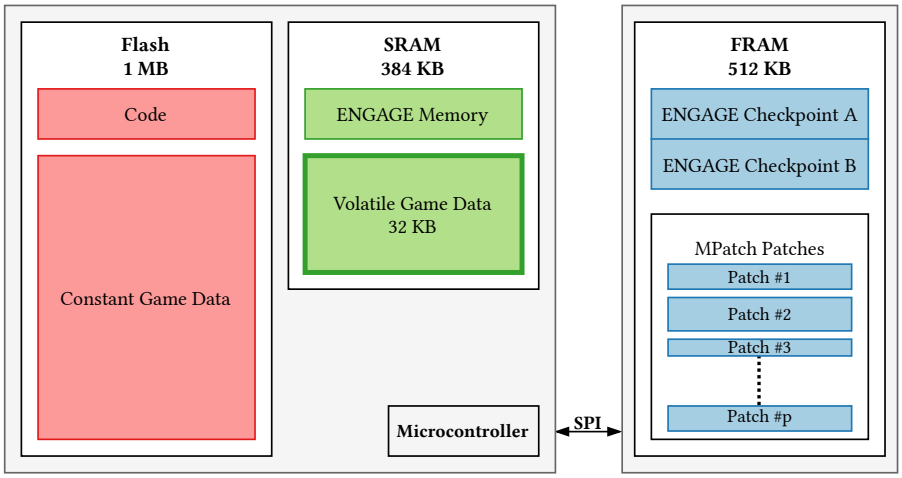


Fig. 8. ENGAGE physical memory structure. Constant game data is executed from Flash with its volatile memory in SRAM, avoiding overhead from accessing the external FRAM. Only checkpoints and patches are stored in external FRAM.

(white, yellow, red and black). This approach is similar to popular emulators that enable colour remapping by default, transforming Nintendo Game Boy games into the modern era where most games are rendered in colour.

ENGAGE Button Handling. In Section 3.2 we outlined the trade-off between energy generation and user satisfaction by altering the emulators handling of the buttons, for example removing the option to hold buttons for an extended duration of time. We limit the duration a button can be held to 300 ms. This is a duration similar to the button presses per second of *Space Invaders* as shown in Table 1. This approach forces the user to press buttons in a frequent pace but does not require excessive button pressing, balancing user satisfaction with energy generation. Due to the flexible nature of the ENGAGE platform future work can focus on user interaction with intermittent gaming devices. We will discuss this further in Section 6.1.

ENGAGE Memory Configuration. The Apollo3 ARM Cortex-M4 features flash and SRAM as on-board memory, where the Flash memory contains all the code (MPatch and Game Boy game emulator code) and non-volatile game data copied from the Game Boy game cartridge. SRAM contains memory of the whole ENGAGE platform and the volatile game memory—both separated from each other. Two buffers, *Checkpoint A* and *Checkpoint B*, for double buffering during checkpointing, as well as all patches created by MPatch reside in external FRAM. The complete memory map of ENGAGE is presented also in Figure 8.

4.3 MPatch Implementation

4.3.1 ENGAGE Core Checkpoints. MPatch is built upon a basic double-buffered checkpoint scheme which we denote as the **core checkpoint** system. The core checkpoint encompasses all the emulation management logic of ENGAGE, except for the emulated game memory, which is checkpointed using patches as described in Section 4.3.2. Specifically, the core checkpoint system checkpoints the .data, .bss and active stack sections of the MCU's volatile memory as well as the registers of the MCU, as can be seen in Algorithm 1. All this is double-buffered in the external non-volatile memory of ENGAGE. Naturally, this means that for every byte of volatile memory in the checkpoint, we need twice as much bytes in non-volatile memory. We remark that not all memory of ENGAGE is checkpointed. Specifically, we do not checkpoint memory buffers required for peripherals (as the peripheral state needs to be re-initialized every ENGAGE reboot). The restoration of a checkpoint will

Algorithm 1 Checkpoint Creation

```
1: function CheckpointCreate
2:     CoreCheckpoint                          ▷ Checkpoint memory not manged by MPatch
3:     PatchesCreate          ▷ Create and stage patches; see Section 4.3.2 and Algorithm 3
4:     RegisterCheckpoint                      ▷ Checkpoint the CPU registers
5:     RestorePoint            ▷ Continuation point after a restore operation
6:     if isNotRestore then
7:         CheckpointCommit                    ▷ Call function that commits the checkpoint
8:     end if
9: end function
```

Algorithm 2 Checkpoint Restoration

```
1: function CheckpointRestore
2:     CoreCheckpointRestore                   ▷ Restore memory not manged by MPatch
3:     PatchesRestore                ▷ Restore committed patches; see Algorithm 5
4:     PeripheralRestore                       ▷ Restore peripherals
5:     RegisterCheckpointRestore               ▷ Restore the CPU registers
6:     RestorePoint                ▷ Continue at the restore point; see Algorithm 1
7: end function
```

restore the state of the system to that of the last successful checkpoint. If the system does not experience a *first time boot*, the default memory initialization step (which traditionally runs before any user code) will be skipped. After this, the steps listed in Algorithm 2 are performed to continue executing as if no power failure had occurred. In line 3 of Algorithm 2 the MPatch patch restoration process is started to restore the emulated game memory which will be discussed further in Section 4.3.2.

We designed the core checkpoint system from the ground up, implementing special keywords enabling the exclusion of certain volatile memory parts from a checkpoint. Also, the core checkpoint provides hooks for every stage of the checkpoint for ease of extension, which is required to incorporate patches from MPatch.

4.3.2 Patch Checkpoints Implementation. The emulated memory, i.e. the memory used by the Game Boy games, is a region in SRAM accessed only by emulated read and write instructions form the emulator. Leveraging this fact makes tracking modification to the emulated memory straightforward, and doing so has little impact on the overall performance. ENGAGE tracks these modifications, and when a checkpoint is created, this information is used to create the required patches as can be seen in Algorithm 3. Tracking of these modifications is done using the memory protection unit of the MCU. Upon writing to a region of emulated game memory, the memory protection unit triggers an interrupt allowing the memory region to be marked as modified. After a region is marked as modified the interrupt for the region is disabled. This results in an efficient method of tracking memory writes since the introduced overhead is only present during the first write after a reboot. The memory protection unit features eight regions which each have eight sub-regions, for a total of 64 sub-regions. We equally divided the memory space of the emulated Game Boy memory between these sub-regions resulting in patches containing 32 kB / 64 = 512 B of emulated memory.

Content of a Patch. In addition to the copy of a volatile memory region, a patch contains accompanying metadata required to successfully manage and restore a patch. This metadata is: (i) the *value of the logical clock n* from when the patch was staged, (ii) the *interval of the volatile memory* that is stored within the patch, (iii) the

Algorithm 3 Patch Creation

```
1: function PatchesCreate
2:     while p ← ModifiedMemory do          ▷ For each of the modified regions of memory
3:         PatchStage(address_start, address_end)   ▷ Create and stage the patch; see Algorithm 4
4:     end while
5: end function
```

Algorithm 4 Patch Staging

```
1: function PatchStage(address_start, address_end)
2:     patch ← AllocatePatch(address_start, address_end)   ▷ Allocate memory for a patch
3:     PatchCreate(patch)   ▷ Copy the volatile memory region into the non-volatile patch
4: end function
```

next patch in the patch chain, (iv) the *metadata to build an augmented interval tree* to speed up the restoration procedure, which will be discussed later in this section.

Patch Allocation. Patch sizes are allowed to differ, therefore some form of dynamic memory allocation is required. This brings challenges, as dynamic allocation leads to fragmentation, which is undesirable in an embedded system. Therefore patches are allocated using a *fixed-size block allocator* [64]. These allocated blocks are chained together to create enough room required to store the volatile memory within the non-volatile blocks. Each block contains: (i) a link to the *next block* in the chain, and (ii) a link to the *next free block* in the chain. All blocks are stored and managed in non-volatile memory. This creates challenges when trying to synchronize its non-volatile and volatile state. If these are not kept in sync, blocks will be lost, and the system may become corrupt. Additionally, write-after-read (WAR) violations [27] should be avoided when interacting with the non-volatile state. These two separate links in a block are required to eliminate one of these WAR violations, this violation could also be eliminated by introducing forced checkpoints, as inserting a checkpoint will break a WAR violation [27]. The total memory overhead of a patch in ENGAGE as it is currently implemented is 29 B. By excluding the interval tree required for the metadata, this can be reduced further to 17 B, but this would require an additional dynamic memory allocator to allocate this memory in volatile memory during a restoration (e.g. standard heap). For the final version used in ENGAGE, this was deemed undesirable, and therefore we integrated the interval tree metadata within non-volatile patches.

Patch Restoration. Restoring patches involves first discarding all staged—but not yet committed—patches, and then iterating through the patch chain while applying only the regions of a patch that were not previously applied during the restoration process. To keep track of the regions of volatile memory that were already restored we maintain an augmented *interval tree* during the restoration process. After a patch is applied, its range is added to the interval tree, and when a patch is applied, the interval tree is queried to detect overlaps. If there are no overlaps, the path is applied (i.e. written to the corresponding region in volatile memory). However, if the patch region overlaps with any region in the interval tree, the patch is split-up and all sub-patches are attempted to be applied. The complete algorithm for patch restoration is shown in Algorithm 5, with its accompanying patch apply algorithm shown in Algorithm 6.

Memory Recovery. One of the features of MPatch is its constant time patch creation while being incorruptible. However, patches that are no longer useful, i.e. that will not be applied during restoration, should be deleted. To avoid WAR violations, removing a patch (reclaiming its memory), consists of two operations. Firstly, the patch is freed, and secondly, the patch is deleted. Between these two operations, a checkpoint of only the MPatch

Algorithm 5 Patch Restoration (note: $low(p)$, $high(p)$ denote the low, high component of range p, respectively)

```
1: function PatchesRestore
2:     DiscardUncommitted                          ▷ Call function that discards uncommitted patches
3:     while p_apply ← next(PatchChain) do         ▷ Extract next patch from patch chain to restore it
4:         PatchApply(p_apply, low(p_apply), high(p_apply))   ▷ Apply patch; see Algorithm 6
5:         IntervalInsert(low(p_apply), high(p_apply))        ▷ Insert the patch range into the interval tree
6:     end while
7: end function
```

Algorithm 6 Patch Apply (note: $low(p)$, $high(p)$ denote the low, high component of range p, respectively)

```
1: function PatchApply(p_apply, low, high)
2:     if p_overlap ← IntervalOverlap(low, high) then   ▷ Check for overlapping region in interval tree
3:         if low < low(p_overlap) then
4:             PatchApply(p_apply, low, low(p_overlap) − 1)   ▷ Recursively apply patch with a new, partial, range
5:         end if
6:         if high > high(p_overlap) then
7:             PatchApply(p_apply, high(p_overlap) + 1, high) ▷ Recursively apply patch with a new, partial, range
8:         end if
9:     else
10:        Write(p_apply, low, high)   ▷ Write patch content between low and high to the volatile memory
11:    end if
12: end function
```

management state is made containing patch and block allocation related metadata. During the deletion of a patch special care is taken to avoid WAR violations when modifying non-volatile memory in the patch chain. Memory recovery is not needed during every time a checkpoint is created or restored, is automatically done when there is no more non-volatile memory available to allocate a patch.

5 ENGAGE EVALUATION

We built ENGAGE as a proof by demonstration that battery-free mobile gaming was possible. In this section we demonstrate that the system can play unmodified retro games despite intermittent power failures. We analyze the real-world execution of the platform while playing *Tetris* in different lighting scenarios (i.e. with different energy scarcity) to show the effect of energy availability. We then benchmark the ENGAGE hardware platform for power consumption and, investigate the performance of the MPatch system. We find that in well-lit environments playing games that require at least moderate amounts of clicking, play is only slightly interrupted by power failures (less than one second of failure per every ten seconds of play). Our measurements of MPatch across four different games show that checkpoints are fast (less than 50 ms and restoration time after a power failures is not noticeable (average of 140 ms).

5.1 End-to-End ENGAGE Performance

First, we look at the typical play of ENGAGE executing an example Nintendo Game Boy game *Tetris*, chosen due to its requirement for moderate/high button presses and a small number of cut-scenes. We show how the system operates only on harvested energy. We execute two experiments, each in different lighting conditions: (i) 'daylight' with approximately 40 klx and (ii) 'shade' with approximately 20 klx, where a gamer plays ENGAGE

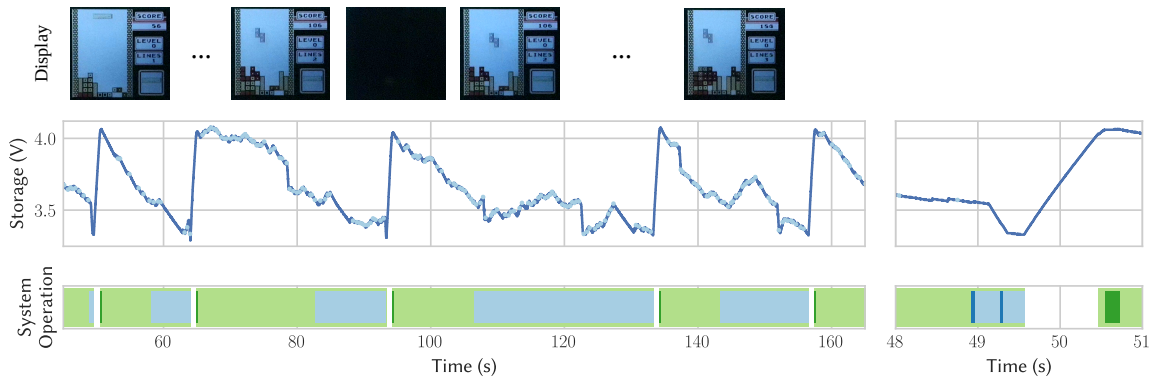


Fig. 9. End-to-end evaluation of ENGAGE operating in 'daylight' (approximately 40 klx during *Tetris* gameplay using harvested energy only. Storage capacitor voltage is shown, overlaid by unique button presses (marked as light blue dots). Additionally the following system events are shown at the bottom of the figure: initialization time (marked in dark green), system on time (marked in light green), low energy state (marked in light blue, denoting moments of ENGAGE periodically checkpointing due to critical system voltage) and checkpoint time (shown in dark blue in the separate zoomed-in window on the right). The actual game frames are shown on top, taken from recording the ENGAGE display during the evaluation scenario. The scenario shows that user interaction prolongs the on time of ENGAGE, by pressing buttons during gameplay—achieving ten seconds or more of on time with small off times. We consider this to be a playable *Tetris* scenario.

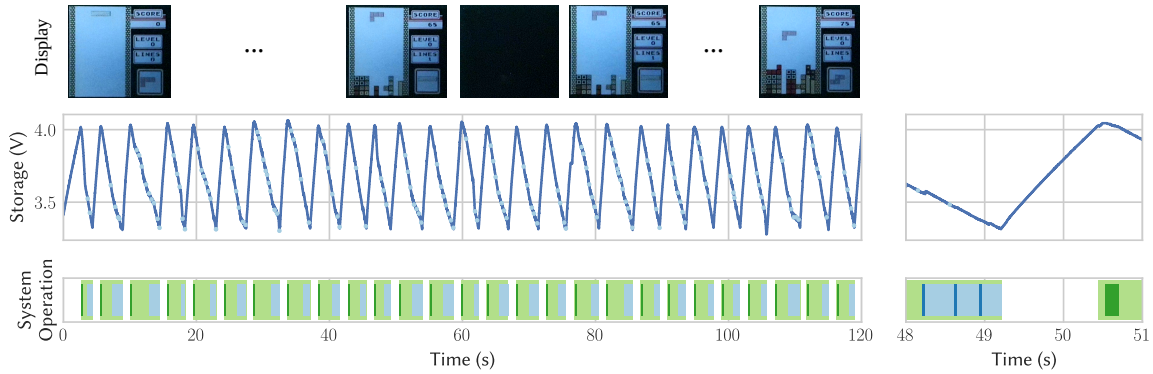


Fig. 10. End-to-end evaluation of ENGAGE operating in 'shade' (approximately 20 klx. Description of figure elements is the same as in Figure 9. With less energy available to ENGAGE as in the scenario in Figure 9, on times are reduced to around 3.5 s, with off times of more than a second. This scenario creates a noticeable impact to the user experience.

fully untethered, operating on harvested energy only. In the experiment the voltage of the main supply capacitor of ENGAGE is recorded together with various debugging signals indicating different system states. The system state and button presses are recorded using a Saleae logic pro 8 logic analyzer [113]. The ENGAGE platform was placed in a light box with two remotely controllable lights generating the two different light exposure conditions. The luminance of both scenarios was verified using a UNI-T UT383 lux meter [133].

In the first scenario ('daylight', Figure 9) we show a period of execution with both little and many button presses. Here clearly the contribution of the energy harvesting by the switches is shown, significantly prolonging the on time of the device (marked in green). The figure shows the complete sequence from startup until the ENGAGE reaches a critically low energy level when it starts checkpointing. Due to the variability in the incoming energy pattern, ENGAGE can spend some time in this state, since it always needs to account for the worst-case scenario of no additional incoming energy. This scenario results in on times of ten seconds or more with small off times of less then a second, making it a very playable experience.

In the second scenario ('shade', Figure 10) we halved the amount of light the solar panels are exposed to compared to 'daylight', a more challenging condition for ENGAGE. This reduces on times to around 3.5 s with off times of more than a second. Despite the system still functioning correctly the lack of incoming energy becomes noticeable and even button mashing cannot compensate for the lack of energy. As with any energy harvesting platform, the limits of operation are defined to a major degree by the available energy in the environment. Full-system emulation is challenging and energy-intensive, but the game is still playable and functional; just with longer intermittent outages. We note that the downward peaks of storage voltage in Figure 9 and Figure 10 are caused by the energy harvester: during maximum power point tracking no energy is harvested causing the quick drop in the storage capacitor voltage.

Full-System Restoration Time. We have also measured end-to-end time of ENGAGE restoration: from the moment of applying power to the MCU to the moment of executing game code within the Game Boy emulator. In the case of *Tetris* this is 264 ms. The other games we tested resulted in comparable restore times, the main difference resulting from MPatch operations, as is further described in Section 5.3.

5.2 ENGAGE Power Consumption and Energy Generation

We have measured ENGAGE's power consumption, looking into overall power consumption whilst first measuring the consumption of MCU together with the FRAM and display. The MCU and FRAM combined consume 11.15 mW and the screen consumes 344.31 µW during game execution taking a ten second average. During idle time the screen only consumes 3.90 µW, resulting in a combined system average power consumption of 11.50 mW. As a comparison, the original Nintendo Game Boy consumes 232.08 mW during game execution, varying slightly per game and cartridge architecture. While not necessarily a useful or meaningful number, we conclude that our platform is *more than 20 times* more power-efficient than the original Nintendo Game Boy (representing normal technology advancement, but noting that ENGAGE is an emulator). The measurements were conducted using a Fluke 87V [33] multimeter and the X-NUCLEO-LPM01A [123] programmable power supply source with power consumption measurement capability.

Energy Generation. Then, to give more insight in the energy harvesting on the ENGAGE platform we have measured the amount of energy the solar panels generate using a Fluke 87V [33] multimeter and compare this to the energy generated from the buttons. For the buttons we use the minimal energy generation figures from the specification of the harvester [153, summary] as a worst case scenario[3]. Assuming that a single button press generates a minimum of 0.66 mJ and knowing the amount of button presses per game is specific to the game as per Table 1, we can assume the buttons generate between 0.66 mJ for one press per second and 1.98 mJ for three presses per second. At 40 klx and 20 klx, the solar panels generated an average of 10.14 mW and 8.33 mW, respectively, i.e. less than the required system average power consumption of 11.50 mW. We can conclude that ENGAGE is mostly powered by solar panels and supplemented by the button presses although the button presses can significantly increase the on-time of the platform, as shown in Section 5.1.

[3]Harvesting energy from the button energy harvesters is highly dependent on numerous factors such as the force applied and the manner of pressing the button hence the choice for the minimal figure.

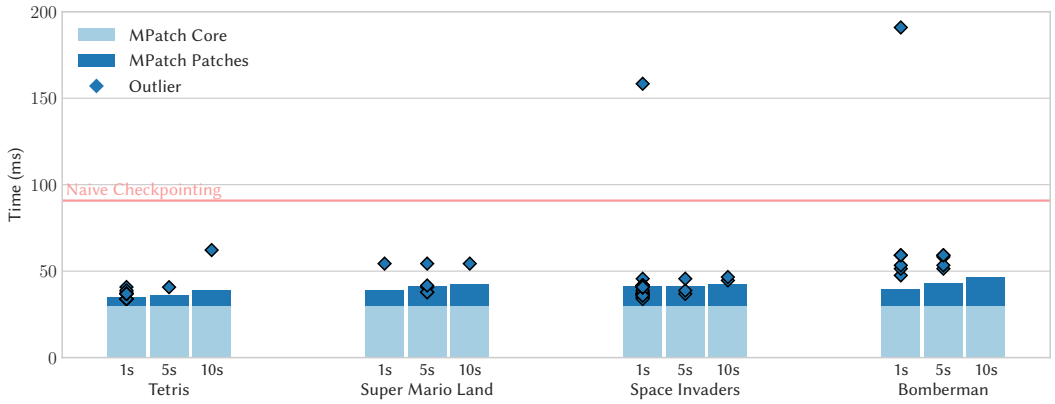


Fig. 11. MPatch checkpoint time comparison of approximately two minutes of game play per game using three different on times (1 s, 5 s, and 10 s) between successive checkpoints. ENGAGE has noticeably better performance than naive system, across all on times and games.

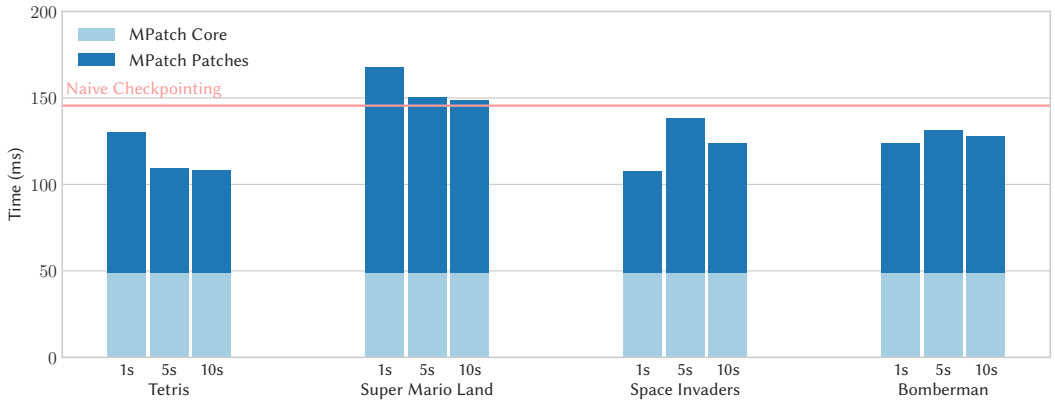


Fig. 12. Restoration time comparison of after approximately two minutes of game play per game using three different on times (1 s, 5 s, and 10 s) between successive checkpoints. ENGAGE has comparable or better performance than naive system, across all on times and games.

5.3 MPatch Performance

To better understand and quantify the effect of patches on the checkpoint and restore time, we evaluate MPatch against a *naive* approach—comparable in operation to Mementos [109]—where all active memory in the system is copied to non-volatile memory during a checkpoint, even if it was not modified since the last checkpoint. We compare these two strategies, MPatch and *naive*, by running multiple different games on ENGAGE. These games include: (i) *Tetris*, (ii) *Super Mario Land*, (iii) *Space Invaders*, and (iv) *Bomberman*. These games represent a wide variety of play styles, developers, and even release dates.

5.3.1 MPatch Checkpoint Time. To measure only the impact of the MPatch patch checkpoints, we disable the just-in-time checkpoints—used in Section 5.1—and run the system on constant power during these measurements. Instead, we perform a checkpoint every c execution cycles of the emulator, we chose three different values for c, which correspond to different *on times*, i.e. 1 s, 5 s, and 10 s. During normal operation checkpoints will only be created when the voltage reaches a critical threshold, as seen in Section 5.1, these fixed on times represent a simplified scenario where the critical voltage threshold is reached after the specified on time. The *on time* affects the number and size of the checkpoints, as it allows for more memory writes between two consecutive checkpoints. The on time does not affect the *naive* checkpoint, as it always checkpoints all memory, with the only variable size being the system stack of ENGAGE. However, because of the way ENGAGE works—as an emulation loop—the system stack size is virtually constant.

During the emulation of each game, with the three different on times, we measured the cost of each component of the checkpoint using the same logic analyzer as used in experiments in Section 5.1. A checkpoint of ENGAGE consists of a core checkpoint of ENGAGE (Section 4.3.1) and additionally patches created by MPatch. The core checkpoint includes the management of both the emulator and the emulated memory, but excludes the emulated memory itself. This emulated memory is the largest memory component of the system, and therefore also the largest component of a naive checkpoint. For this reason we checkpoint this part of the system using MPatch, as the other components of ENGAGE are virtually constant in the amount of memory that is modified and are thus covered by the core checkpoint.

Figure 11 illustrates the naive checkpoint time as the horizontal line, the average checkpoint time of a core checkpoint (light blue bar), the differential component of a checkpoint using MPatch (dark blue bar), and the outliers (blue diamonds). As can be seen, the cost of the core checkpoint is around 30 % of the complete *naive* checkpoint, the rest being the emulated memory. However, when using MPatch to checkpoint the emulated memory, the core checkpoint dominates the total checkpoint time. In total MPatch is on average *more than two times faster* than the *naive* approach. This confirms our hypothesis that only a small amount of emulated memory is modified during execution. This reduction in checkpoint time directly leads to a lower energy requirement for each checkpoint and leaves more time for game emulation. Interestingly this assumption seems to hold even when the on time approaches 10 s, which is substantial for intermittent devices. Some outliers take longer than a *naive* checkpoint, this is due to a periodically performed memory recovery procedure (Section 4.3.2)—which was introduced to keep the creation of patches constant while keeping the system incorruptible.

5.3.2 MPatch Restoration Time. We also evaluate restoration time of patch checkpointing of MPatch, in a similar manner as in the previous section (i.e. the same set of games, comparison against three other reference mechanisms). The results are presented in Figure 12.

Restoring a patch-based checkpoint requires more time than the creation of a patch, as described in Section 4.3.2, due to the need to apply only the parts of the patches that are required, and because all the volatile memory has to be restored. Additionally, the restoration procedure must take into account all the committed patches when trying to restore the volatile memory, as each of these might hold some region that was only checkpointed using that specific patch. Therefore it is not directly influenced by the on-period, but influenced by the time since a *memory recovery*. Nevertheless, as can be seen in the figure, MPatch often *reduces the restoration time* compared to the *naive* restoration. We can also conclude from this that tested games often only modify a portion of their memory (in this case the emulated memory), as can also be seen in Figure 4.

6 DISCUSSION AND FUTURE WORK

Our evaluation of ENGAGE has shown that retro games are playable without batteries, making a next step in self-sustainable gaming made first decades ago by e.g. Bandai Corporation's LCD Solarpower game series [26]. Although the core gameplay mechanisms of the mobile handheld gaming have been successfully implemented,

i.e. interaction with a screen-displayed data (for the original Nintendo Game Boy), other forms of interaction that make game experience complete are waiting to be researched and implemented.

6.1 Limitations, Alternatives and Future Work

Of course ENGAGE is just a first step in the direction of battery-free gaming and the proposed platform has still many limitations that need to be addressed. First, our battery-free platform *plays no sound*. We agree that no sound play is the main hurdle of complete game immersion. How to make sound enjoyable despite power supply intermittency is the core research question, but at the same time (in our opinion) an exciting research area. Some approaches to the sound problem we anticipate as worth-considering are (i) to include separate storage for sound buffering and play, following the architecture of [23, 42], (ii) introduce superficial pauses in the original game tone—effectively making the game sounds identical to the original battery-based game but punctured by silence at pre-selected moments—to make sound interrupts less irritating during gameplay, or (iii) to create intermittent system-specific game sounds—sounds that inform the user that the system is about to die or has just become operational again—to enrich battery-free gameplay.

Second, *playing in the dark has not been addressed*, as screen we used has no back light[4]. In the context of battery-free gaming, provision of back light for screen is very difficult. Simply, lack of light reduces amount of energy from harvesting, which in consequence reduces chance to perform *any* task—let alone supporting LEDs that are the most energy-consuming components of any embedded system.

Third, *haptics for battery-free games needs deeper investigation*. The energy harvesting buttons we have used in our prototype [153] are designed for industrial *sporadic* single-press cases (think of a battery-free wireless light switch). In frequent pressing cases these switches are much sturdy than the original Game Boy buttons, which for gamer can be a distracting feature. This necessitates a quest for more natural press buttons with equal (or better) energy harvesting. Furthermore, solar panels have to be placed on console's chassis such that their obstruction by fingers is minimized (as in our prototype). This, however, might downgrade the aesthetics of the device or require to make it bigger than necessary.

Fourth, *networking with battery-free game consoles* is another important point to consider, which was not addressed by us. Original Game Boy had an ability to connect, via cable, to another Game Boy for tandem gaming. This cable connection can be actually used for energy sharing and balancing between two consoles. Wireless networking of battery-free is an ongoing research task, which we did not want to cover with this work (for state of the art battery-free networking overview we refer to Section 7).

Fifth, *we cannot claim that all games will have the same playability* when ported to the intermittently-powered domain. Only when the off-times are negligible for the player we can safely assume that any existing game could be played intermittently/battery-free. Negligible off-times will cause no irritation to the person who is accustomed to always-on style of play. This observation would hold for any game system—not only classical (but old) Nintendo GameBoy we used as a basis for ENGAGE, but also recent systems such as PlayStation Portable or Nintendo Switch. An open research question is to find how long this off time is (less than a second or maybe less than a millisecond)? Our intuition says that this time is game-dependent and the longer the off times are present in a battery-free console, the set of games that can be ported to the battery-free platform gets smaller. Games that do not need frequent button pushes intuitively would be less irritating to play intermittently (e.g. *Chess*) or *Solitaire*), refer also to qualitative comparison of 8-bit Nintendo Games portability in Table 3. However, this creates an interesting paradox of button-based interaction. More button presses during the game result in more energy supplied to the game console, see also Table 1 and Section 3.2 (in extreme case games that are based on button bashing, such as classical *Track & Field* arcade game from Konami Corporation, gamer would be able to continuously generate energy purely from gameplay). At the same time less button presses result in less

[4]To be fair, the original Game Boy had the same deficiency, so do some of the upcoming gaming consoles [104].

Table 3. This table describes the difficulty (or irritability) of playing types of Nintendo GameBoy games on intermittent power, assuming the intermittent effect is noticeable to the player and that enough energy is available for some level of play.

Game name	Type	Button presses	Intermittent play	Comments
Baseball	Sports	Very High	Hard	Reaction time is part of the game
Super Mario Land	Platformer	High	Hard	Button press order is crucial
Tetris	Puzzle	High	Medium	Tile rotation is often infrequent
Solitaire	Cards	Low	Easy	No penalty for missing a press
WordZap	Puzzle	Low	Easy	Easy with "no solving time" penalty
Chess	Strategy	Very Low	Easy	Most time spent on thinking

energy being created, causing reduction in continuous duration of play. To verify the above claims *detailed user studies considering large pool of gamers and games* need to be performed, where users play different games with artificially-induced intermittent operation (varying duration of on and off times).

Sixth, *screen retention needs to be introduced* (keeping screen state in-between on times) which our ENGAGE has not implemented yet. This simple extension would significantly reduce perceived negative effect of intermittent operation (think of a *Chess* game where state of the screen does not change much when the player is thinking and often user would not distinguish between off time and regular game operation, see again Table 3).

Finally, the overarching goal is to be able to *play state of the art 21 century handheld game consoles battery-free*, such as Playstation Vita Nintendo Switch—going beyond 8 bit architecture. This however requires years of research and can only be achieved by further advances in intermittently-powered software frameworks and ultra-low power electronics, which hopefully this work made a first step in achieving this goal.

General Software Framework for Battery-free Games. The goal of being able to run any existing or future game battery-free requires the introduction of general software framework for such games, going beyond checkpoining mechanism or a driver design presented in this paper, which is of course tailored towards the 8 bit Game Boy emulator. We envision a game engine, inspired by game engines of existing video games, such as the *Source* game engine [136] used in first-person shooter games such as *Counter-Strike*, that supports battery-free interaction abstracting underlying frameworks for intermittent operation form an actual game design.

Further Reduction of Game Console Carbon Footprint. We have made a first step towards making Game Console fabrication more environmentally friendly, however this is just a first step. Needless to say, original game cartridges of Nintendo Game Boy contain battery, and our console is based on many electronic components that are responsible for large CO_2 emissions in production [38], not to mention chassis that is made of plastic. More radical ideas need to be exploited, such as on the electronic level design with minimum amount of components (e.g. crystal-free design), going beyond policy changes in electronic fabrication enlisted, e.g. in [36].

Behavior Nudges to Generate More Energy. Many types of games have natural gaming mechanics that could be leveraged to increase energy harvesting actions. *Dance Dance Revolution*, *Bop-It*, and others, exploring this gaming induced behavior change for increasing energy is an interesting research direction. For example, a specific rapid button pressing sequence can trigger new game events (new levels, extra game points, etc.). Then, there are great user interfaces for battery-free interaction, for instance a crank[5], that can be researched further.

Native Execution. We chose the hard path: running a game emulator on an intermittent platform. This was to demonstrate the range of capabilities available to intermittent computing, and to leverage the vast amount of

[5]Which is already used in the upcoming post-retro *Playdate* console [104], which sadly is not used for internal battery charging.

pre-built games that can play unchanged on the platform. However, one could imagine that native gameplay would significantly increase the performance of the platform, by orders of magnitude, since a single emulated instruction has significant overhead over native code for the platform. This could be accomplished by compiling game binaries to native ARM code, or by leveraging a bespoke gaming API from bare-metal C code. The latter is intriguing as an exercises to take advantage of the unique aspects of intermittently-powered and battery-free gaming, where the situation and context, as well as the gameplay, will effect how much energy is harvested. Game mechanics leveraging this system attribute might increase engagement.

6.2 Gaming and the Environment

Electronic games are an important part of the world's economy [95]. First and foremost they are crucial to mental well being of many people around the world. Especially in the time of the COVID-19 pandemic, when millions of people are stranded at home, various forms of electronic gaming are one of the activities that reduce stress and boredom due to lockdown implemented by most of world's governments [116, 118, 127]. At the same time, the electronic gaming industry is and important job creator, and although being a financially non-struggling industry, to say the least [95, 100, 118], is also actively supported by international governments': as an latest example refer to CD Projekt—creator of *The Witcher* video game series—and its list of European Union-funded projects the company participated in [15]. At the same time it is apparent that gaming industry contributes significantly to global warming. In the United States alone gaming is responsible for "24 MT/year of associated carbon-dioxide emissions equivalent to that of 85 million refrigerators" [90]. To tackle that challenge the gaming industry is joining various industry consortia such as *Playing for the Planet* [108] aiming at reducing its ecological impact. Independently, some national governments aim at influencing the gaming industry requesting content providers to throttle-down data rate of streaming services with too high demand [36]—resulting in smaller electricity consumption of data centers.

But all the above actions to address climate impact of the gaming industry do not tackle the effect of battery-based/handheld/mobile gaming (the above-mentioned study of [90] explicitly excludes such devices from the analysis). Beyond any doubt handheld gaming devices, while extremely popular [98], contribute independently to increased worldwide CO_2 emissions. While we are not aware of any detailed studies on the carbon footprint of popular handheld gaming consoles, such as Nintendo Switch[6], its impact is beyond negligible. For example, Nintendo was *the least* environmentally-friendly company of Greenpeace 2010 *Guide to Greener Electronics* ranking [152], while none of the video-game oriented companies are listed among the world's most sustainable corporations in year 2020 [25].

There are numerous components that gaming handheld console/mobile phone is made of that cause substantial environmental impact [38] so removing some of them without compromising the usability would by highly appreciated from the environment point of view. A first potent candidate for such removal is a battery. Production of batteries has great environmental impact by itself and many research projects are devoted to making *batteries-only* more sustainable [31]. But even if most of the goals of more sustainable batteries are met by 2030, they will *still* have to be produced, collected and recycled. And while we conjecture that majority of console game players do not consider reliance on batteries as a problem[7] it is the *environmental responsibility of the electronic designers* to address the battery issue for the users of handheld gaming consoles.

[6]None of the handheld gaming platforms are listed in the Electronic Product Environmental Assessment register [39]; the closest study of environmental impact of Nintendo Switch we are aware of is given in [69]. As a reference, in-depth analysis of carbon footprint of one of the most popular non-handheld gaming console, Sony's Play Station 4, is available in [38]. To quote from this study: "(s)ince the PlayStation 4's release in 2013, approximately 8.9 billion kilograms of carbon dioxide have been generated and subsequently released into the atmosphere".
[7]Actually, in many cases game console players are close to a power socket playing their games tethered, making a problem of battery replacement or recharge even less profound.

7 RELATED WORK

Battery-free Sensors. Long before our idea of a batter-free gaming console, non-gaming embedded platforms were realized in a battery-free manner—making these sensor more environmentally-friendly. The first such battery-free platforms were wireless sensors [106]. First battery-free sensors were based on the idea of computational RFID tags: programmable RFID tags with on-board sensors (such as accelerometers or temperature sensors) communicating with the outside world by radio frequency backscatter to a RFID reader. WISP [114, 135] and Moo [134] are the first realization of such RFID tags. Since the introduction of WISP and Moo many research groups have focused on making battery-free backscatter communication more efficient [146], for instance, by making it free from dedicated energy sources [105], by enabling communication with non-backscatter networks such as IEEE 802.11 [63] or LoRa [125], or by improving backcatter-based networks—either based on standard RFID protocols [80], or based on dedicated backscatter network stack [41]. A separate line of research focused on introducing camera-based image processing to backscatter-based sensors. First, a backscatter-based battery-less cameras, as an extension to WISP platform, has been demonstrated in [93, 94], later followed by a dedicated (non-WISP) backscatter-based system [92, 112]. Additionally, non-radio frequency backscatter systems based on passive visible light communication backscatter, such as PassiveVLC mote [148], have also been demonstrated. It is important to remark that the biggest drawback of backscatter-based systems is the reliance on external energy source (itself powered by batteries or power line) that downscales the benefit of removing battery from a complete system.

Additionally, battery-free sensors that communicate using non-backscatter, i.e. active, communication techniques also become actively researched. These include simple sense and transmit sensor powered by ambient temperature differences [156], UFoP [42] and Capybara [23]—energy-harvesting storage-adaptive sensors, Battery Free Phone [126], SkinnyPower—wearable sensor powered by intra-body power transfer [121], Camaroptera—image-inferring sensor [96], SoZu—battery-free activity detector [155], or Botoks—time-aware wireless sensor [28]. Non-wireless/non-communicating battery-free sensors include CapHarvester—local energy monitor powered by harvesting stray voltage from AC power lines-[40], self-powered step motion counter [60], Saturn—battery-free microphone [6], and active radio battery-less eye tracker [73].

Battery-free Interactive Devices. It is imperative to extend battery-less devices beyond a simple 'sense-and-transmit' functionality (as summarized above) demonstrating simple forms of user interaction. The same RFID technology that lay the foundation for battery-free sensing was also used to demonstrate battery-less interaction. Such systems include RFID-based tags displaying external information [102], elderly monitoring based on embedded-in-clothes RFID tags [58], surface shape detection [59], speech recognition [142], augmented reality with (i) unmodified RFID tags [72] and (ii) modified RFID tags (to enable touch sensing) [47], interactive building block system with augmented RFID tags[8] [48, 76] or finger gesture measurement [62]. It needs to be emphasized that any RFID tags-based interaction is very sensitive to interference and signal mis-matches as demonstrated in [141].

Separately from RFID-based battery-free interactive devices, non-RFID counterparts are also actively researched. Most of these devices focus on remote device control through touch. Examples of such devices are capacitance-based touch sensor (although communicating with FM radio receiver through backscatter) [140], Ohmic-Sticker—force-to-capacitance sensor attachable to laptop touchpad [51], aesthetically pleasing self-powered interactive surfaces based on photovoltaic cells [87] and self-powered gesture recognition based on (i) photovoltaic panels [79][9], (ii) photodiodes [74] and (iii) capacitance sensing [132]. E-ink battery-free wearable displays embedded in clothes, energized by NFC-enabled smartphones were demonstrated in [29].

[8] A similar concept for NFC-based tags has been presented in [14].
[9] System claims to be battery-less, while in evaluation a battery-based version was used.

Another approach for battery-free embedded devices is to equip the area where the sensor resides in some form of wireless power transfer system. Many end-to-end wireless power solutions can be found in the literature, including recent systems build on top of capacitive power transfer [154], magnetic resonant coupling [124], quasistatic cavity resonance [115], lasers [54] or distributed RF beamforming [32]. As in the case of backscatter-based sensors, wirelesly-powered sensors require external (complex, bulky and still having not fully resolved safety issues) infrastructure. This limits applicability of this approach to ubiquitous battery-free gaming.

Battery-free Gaming. An ultimate form of interaction is through a gaming system. A first, commercial battery-free/solar-powered gaming platform was Bandai's LCD Solarpower [26], released already in 1982, that enabled manipulation of hard-coded elements on a liquid crystal display. Unfortunately, Bandai's console and modern existing academic-grade battery-free gaming systems are limited to a simple game forms, such as attachable touch pad extenders for better (but still battery-powered) mobile game experience [18, 151] (similar to an earlier referred design [51]), extra controllers for smartphones based on its front/rear cameras [149], or based on RFID technology that requires heavy-lifting of battery-less features by an expensive RFID reader using either (i) computational RFID tags [134, 135] as for instance in [86], or (ii) using commercial off-the-shelf RFID tags as in [71]. Battery-free non-RFID touch pad extender for the introduction of physical manipulation into touch screen-based games was prototyped in [97]. Battery-free gaming aimed at children includes system based on rubbing/touching electrostatic surfaces to power simple electronics [16, 17, 61] and attachable energy harvester mote for learning and understanding concepts of energy generation and consumption [111].

New Electronic Game Forms. Battery-less handheld gaming console, ENGAGE, presented in this paper introduce a novel form of self-powering play, where user (to continue playing a normal electronic handheld game platform) is (sometimes) required to push buttons to continuously power a device. This is a twist on movement-inducing (exer)games [9, 53] such as Pokémon GO [66] where movement is required only to *perform better* in game instead of *perform better and continue* to play. This is a new form of game interaction that use the human body as an immanent component of gaming experience, as advocated in [91]. We note that novel forms of games with dedicated hardware (albeit battery-powered) are introduced, where the energy of the body is used to introduce a novel form of interaction. A recent example of such game is based on swallowable temperature measurement pills [75] or through-body electric field propagation [138, 139].

Gaming as a Behavioral Intervention. Research community is in constant search for new forms of gaming interaction and our battery-less gaming console aims at defining yet another gaming behavior. Such new forms of gaming are for instance, 'idle games' [4] or new game forms with custom-made haptics, such as virtual reality games for blind people [117]). Design challenges in behaviour-inducing games (such as exergames referred earlier) have been discussed recently in [66].

Considering classical gaming behavioral studies we can refer to game design that activate children to play outdoors [101], study on the effect of 'gamification' of cognitive tasks [143] or observation of gaming experience as an indication of cognitive abilities [52]. We are not aware of any non-orthodox gaming behavioral studies.

A separate line of research, although not strictly related to games, touches upon behavioral change of battery-powered smartphones usage. These studies include crowdsensing of battery usage for suggestion of better user behaviour extending battery lifetime [21], optimization of frame rate for mobile (smartphone-based) games saving energy on frame rendering [50], or a proposal for new form of interaction with mobile devices with turned-off screen to conserve energy [147]. Our battery-free game console is the first study that considers a behavioral intervention for *battery-free* device.

Sustainable Design of Interactive Devices. Design of any future interactive devices must consider sustainability and reuse, as advocated already a decade ago in [12, 85]. The same plea, but in the context of pervasive devices, was presented in [55]. Since almost a decade many studies call for sustainable 'upstream' HCI by making

conscious choices in HCI design process in selecting materials that are sustainable, recyclable and reusable [65] or using post-apocalyptic terms— HCI "designed for use after the industrialized context has begun to decay" [131]. We are unaware of any studies on whether the (handheld) gaming community considers sustainable gaming as important, let alone existing, problem. Loosely related study to our posted problem is the study on the motivations behind leading green households [145].

Intermittent Computing Systems. The goal of intermittent computing frameworks is to guarantee correctness and completion of the computation of battery-less energy harvesting embedded platforms *despite* frequent power interrupts[10]. Such framework is essential for the usability of battery-free gaming platform.

From the publication of a first framework supporting intermittently-powered devices, Mementos [109]—voltage threshold-triggered checkpoining system, more efficient checkpoint systems are being published. These include Hibernus++ [8] and QuickRecall [57] (just like Mementos, both hardware-activated chceckpoints), Chinchilla [82], Rachet [137] and HarvOS [11] (all three compiler-instrumented checkpoints), TICS (time-aware checkpoints) [68], TotalRecall (checkpoints using volatile memory) [144], Elastin (adaptive checkpoints) [19], DICE (differential checkpoints) [2, 3] and WhatsNext (checkpointing augmented with approximate computing) [35]. A second class of systems include runtimes based on specially instrumented code (by form for a *task*) such as Dino [78], Chain [22], Alpaca [81], MayFly [45], InK [150], Coati [110], CoSpec [20] and Coala [84]. A third class of intermittent computation support systems are hardware-assisted systems such as Clank [46] that check for memory inconsistencies. Important to recall are workload-specific computation systems such as on-device inference on intermittently-powered devices with off-line and on-line learning, see [37] and [70], respectively.

A separate stream of work targets peripheral support for intermittently-powered devices, such as Restop (through dedicated middleware) [7], Samoyed (through just-in-time checkpoints) [83] and Karma (supporting parallel or asynchronous peripheral operations) [13], or targeting handling of dedicated peripherals such as e-displays (to improve their update rate) [88].

8 CONCLUSIONS

This paper presented a first working example of a battery-free gaming console, and the first full system emulation on intermittent power: ENGAGE. We demonstrate we can port existing battery-based gaming platforms—such as in our case 8 bit Nintendo Game Boy—to the battery-free domain. With this platform we have shown that deeply interactive devices, like gaming platforms, are possible to create without batteries, and in spite of frequent power failures. We developed a novel hardware and software platform to facilitate this new class of device: (i) a hybrid energy harvesting device tailored towards battery-free gaming and (ii) a new system for persistent computation across power failures based on a novel concept of patch checkpointing of volatile memory state into non-volatile memory regions. ENGAGE represents a bright future of deeply interactive, maintenance and battery-free devices.

ACKNOWLEDGMENTS

We thank our anonymous reviewers for their useful comments. In addition, we would like to thank Izabela Grudzińska for help in preparing the photographs for the paper. This research was supported by Netherlands Organisation for Scientific Research, partly funded by the Dutch Ministry of Economic Affairs, through TTW Perspective program ZERO (P15-06) within Project P1 and P4, and by the National Science Foundation through grants CNS-1850496 and CNS-2032408. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] 2020. ENGAGE Open Source Repository. https://github.com/tudssl/engage. (July 2020). Last accessed: Jul. 22, 2020.

[10]For a good overview of intermittent computing concepts we independently refer to [44, 77, 89].

[2] Saad Ahmed, Muhammad Hamad Alizai, Junaid Haroon Siddiqui, Naveed Anwar Bhatti, and Luca Mottola. 2018. Poster Abstract: Towards Smaller Checkpoints for Better Intermittent Computing. In *Proc. IPSN* (Nov. 11–13). ACM/IEEE, Porto, Portugal, 132–133.

[3] Saad Ahmed, Naveed Anwar Bhatti, Muhammad Hamad Alizai, Junaid Haroon Siddiqui, and Luca Mottola. 2019. Effient Intermittent Computing with Differential Checkpointing. In *Proc. LCTES*. ACM, Phoenix, AZ, USA, 70–81.

[4] Sultan A. Alharthi, Olaa Alsaedi, Zachary O. Toups, Joshua Tanenbaum, and Jessica Hammer. 2018. Playing to Wait: A Taxonomy of Idle Games. In *Proc. CHI* (April 21–26). ACM, Montréal, QC, Canada, 210:1–210:13.

[5] Ambiq Micro Inc. 2018. Apollo3 Blue Ultra-Low Power Microcontroller. https://ambiqmicro.com/static/mcu/files/Apollo3_Blue_MCU_Data_Sheet_v0_11_0.pdf. (2018). Last accessed: Apr. 25, 2020.

[6] Nivedita Arora, Steven L. Zhang, Fereshteh Shahmiri, Diego Osorio, Yicheng Wang, Mohit Gupta, Zhengjun Wang, Thad Eugene Starner, Zhonglin Wang, and Gregory D. Abowd. 2018. SATURN: A Thin and Flexible Self-powered Microphone Leveraging Triboelectric Nanogenerator. *ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 2 (June 2018), 60:1–60:28.

[7] Alberto Rodriguez Arreola, Domenico Balsamo, Geoff V. Merrett, and Alex S. Weddell. 2018. RESTOP: Retaining External Peripheral State in Intermittently-powered Sensor Systems. *Sensors* 18, 1 (2018), 172.

[8] Domenico Balsamo, Alex S. Weddell, Anup Das, Alberto Rodriguez Arreola, Davide Brunelli, Bashir M. Al-Hashimi, Geoff V. Merrett, and Luca Benini. 2016. Hibernus++: a Self-calibrating and Adaptive System for Transiently-powered Embedded Devices. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* 35, 12 (2016), 1968–1980.

[9] Soumya C. Barathi, Daniel J. Finnegan, Matthew Farrow, Alexander Whaley, Pippa Heath, Jude Buckley, Peter W. Dowrick, Burkhard C. Wünsche, James L. J. Bilzon, Eamonn O'Neill, and Christof Lutteroth. 2018. Interactive Feedforward for Improving Performance and Maintaining Intrinsic Motivation in VR Exergaming. In *Proc. CHI* (April 21–26). ACM, Montréal, QC, Canada, 408:1–408:14.

[10] Uwe Becker. 2017. Gameboy-Emulator per STM32F746 (in German). https://mikrocontroller.bplaced.net/wordpress/?page_id=1290. (Nov. 2017). Last accessed: May 5, 2020.

[11] Naveed Bhatti and Luca Mottola. 2017. HarvOS: Efficient Code Instrumentation for Transiently-powered Embedded Devices. In *Proc. IPSN* (April 18–20). ACM/IEEE, Pittsburgh, PA, USA, 209–219.

[12] Eli Blevis. 2007. Sustainable Interaction Design: Invention & Disposal, Renewal & Reuse. In *Proc. CHI* (April 28 – May 3). ACM, San Jose, CA, USA, 503–512.

[13] Adriano Branco, Luca Mottola, Muhammad Hamad Alizai, and Junaid Haroon Siddiqui. 2019. Intermittent Asynchronous Peripheral Operations. In *Proc. SenSys* (Nov. 10–13). ACM, New York City, NY, USA, 55–67.

[14] Lars Büthe, Michael Hardegger, Patrick Brulisauer, and Gerhard Tröster. 2014. RFID-Die: Battery-free Orientation Sensing Using an Array of Passive Tilt Switches. In *Proc. UbiComp Adjunct*. ACM, Seattle, WA, USA, 215–218.

[15] CD Projekt. 2020. European Union Projects. https://www.cdprojekt.com/en/capital-group/eu-projects. (April 2020). Last accessed: Apr. 28, 2020.

[16] Arunkumar Chandrasekhar, Gaurav Khandelwa, Nagamalleswara Rao Alluri, Venkateswaran Vivekananthan, and Sang-Jae Kim. 2018. Battery-Free Electronic Smart Toys: A Step toward the Commercialization of Sustainable Triboelectric Nanogenerators. *Sustainable Chemistry and Engineering* 6, 5 (April 2018), 6110–6116.

[17] Arunkumar Chandrasekhar, Gaurav Khandelwal, Nagamalleswara Rao Alluri, Venkateswaran Vivekananthan, and Sang-Jae Kim. 2017. Sustainable Biomechanical Energy Scavenger toward Self-Reliant Kids' Interactive Battery-Free Smart Puzzle. *Sustainable Chemistry and Engineering* 5, 8 (June 2017), 7310–7316.

[18] Tzuwen Chang, Neng-Hao Yu, Sung-Sheng Tsai, Mike Y. Chen, and Yi Ping Hung. 2012. Clip-on Gadgets: Expandable Tactile Controls For Multi-touch Devices. In *Proc. MobileHCI* (Sept. 21–24). ACM, San Francisco, CA, USA, 163–166.

[19] Jongouk Choi, Hyunwoo Joe, Yongjoo Kim, and Changhee Jung. 2019a. Achieving Stagnation-Free Intermittent Computation with Boundary-Free Adaptive Execution. In *Proc. RTAS*. IEEE, Montréal, QC, Canada, 331–344.

[20] Jongouk Choi, Qingrui Liu, and Changhee Jung. 2019b. CoSpec: Compiler Directed Speculative Intermittent Computation. In *Proc. MICRO*. ACM, Columbus, OH, USA, 399–412.

[21] Yohan Chon, Gwangmin Lee, Rhan Ha, and Hojung Cha. 2016. Crowdsensing-based Smartphone Use Guide for Battery Life Extension. In *Proc. UbiComp*. ACM, Heidelberg, Germany, 958–969.

[22] Alexei Colin and Brandon Lucia. 2016. Chain: Tasks and Channels for Reliable Intermittent Programs. In *Proc. OOPSLA* (Oct. 30 – Nov. 4). ACM, Amsterdam, The Netherlands, 514–530.

[23] Alexei Colin, Emily Ruppel, and Brandon Lucia. 2018. A Reconfigurable Energy Storage Architecture for Energy-harvesting Devices. In *Proc. ASPLOS* (March 24–28). ACM, Williamsburg, VA, USA, 767–781.

[24] Rodrigo Copetti. 2019. Architecture of Consoles: GameBoy. https://copetti.org/projects/consoles/game-boy. (Feb. 2019). Last accessed: May 3, 2020.

[25] Corporate Knights. 2020. 2020 Global 100 Ranking: Index of the World's Most Sustainable Corporations. https://www.corporateknights.com/reports/2020-global-100. (Winter 2020). Last accessed: Apr. 30, 2020.

[26] Bandai Corporation. 1982. LCD Solarpower Handheld Electronic Games Series. https://en.wikipedia.org/wiki/Bandai_LCD_Solarpower. (1982). Last accessed: Sep. 22, 2020.

[27] Marc de Kruijf and Karthikeyan Sankaralingam. 2013. Idempotent Code Generation: Implementation, Analysis, and Evaluation. In *Proc. CGO*. ACM/IEEE, Shenzhen, China.
[28] Jasper de Winkel, Carlo Delle Donne, Kasım Sinan Yıldırım, Przemysław Pawełczak, and Josiah Hester. 2020. Reliable Timekeeping for Intermittent Computing. In *Proc. ASPLOS* (March 16–20). ACM, Lausanne, Switzerland, 53—-67.
[29] Christine Dierk, Molly Jane Pearce Nicholas, and Eric Paulos. 2018. AlterWear: Battery-Free Wearable Displays for Opportunistic Interactions. In *Proc. CHI* (April 21–26). ACM, Montréal, QC, Canada, 210:1–210:13.
[30] Economist Intelligence Unit. 2020. The IoT Business Index 2020: a Step Change in Adoption. https://learn.arm.com/rs/714-XIJ-402/images/economist-iot-business-index-2020-arm.pdf. (Feb. 2020). Last accessed: May 7, 2020.
[31] Kristina Edström (Executive Publisher). 2020. Horizon 2020 EU Program Battery 2030+: Inventing the Sustainable Batteries of the Future: Research Needs and Future Actions. https://battery2030.eu/digitalAssets/861/c_861350-l_1-k_roadmap-27-march.pdf. (March 2020). Last accessed: Apr. 30, 2020.
[32] Xiaoran Fan, Han Ding, Sugang Li, Michael Sanzari, Yanyong Zhang, Wade Trappe, Zhu Han, and Richard E. Howard. 2018. Energy-Ball: Wireless Power Transfer for Batteryless Internet of Things through Distributed Beamforming. *ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 2 (June 2018), 65:1–65:22.
[33] Fluke. 2020. Fluke 87V Industrial Multimeter. https://www.fluke.com/en-us/product/electrical-testing/digital-multimeters/fluke-87v. (March 2020). Last accessed: Jun. 22, 2020.
[34] Fujitsu Semiconductor Ltd. 2018. MB85RS4MT 512 KB SPI FRAM. https://www.fujitsu.com/uk/Images/MB85RS4MT.pdf. (2018). Last accessed: Apr. 25, 2020.
[35] Karthik Ganesan, Joshua San Miguel, and Natalie Enright Jerger. 2019. The What's Next Intermittent Computing Architecture. In *Proc. HPCA*. IEEE, Washington, DC, USA, 211–223.
[36] German Federal Minister for the Environment, Nature Conservation, and Nuclear Safety. 2020. Umweltpolitische Digitalagenda (in German). https://www.bmu.de/fileadmin/Daten_BMU/Pools/Broschueren/broschuere_digitalagenda_bf.pdf. (Feb. 2020). Last accessed: Apr. 28, 2020.
[37] Graham Gobieski, Brandon Lucia, and Nathan Beckmann. 2019. Intelligence Beyond the Edge: Inference on Intermittent Embedded Systems. In *Proc. ASPLOS*. ACM, Providence, RI, USA, 199–213.
[38] Lewis Gordon. 2019. The Environmental Impact of a PlayStation 4. https://www.theverge.com/2019/12/5/20985330/ps4-sony-playstation-environmental-impact-carbon-footprint-manufacturing-25-anniversary. (Dec. 2019). Last accessed: Apr. 28, 2020.
[39] Green Electronics Council. 2020. Electronic Product Environmental Assessment Tool. https://epeat.net. (April 2020). Last accessed: Apr. 30, 2020.
[40] Manoj Gulati, Farshid Salemi Parizi, Eric Whitmire, Sidhant Gupta, Shobha Sundar Ram, Amarjeet Singh, and Shwetak N. Patel. 2018. CapHarvester: A Stick-on Capacitive Energy Harvester Using Stray Electric Field from AC Power Lines. *ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 3 (Sept. 2018), 110:1–110:20.
[41] Mehrdad Hessar, Ali Najafi, and Shyamnath Gollakota. 2019. NetScatter: Enabling Large-Scale Backscatter Networks. In *Proc. NSDI*. USENIX, Boston, MA, USA, 271–283.
[42] Josiah Hester, Lanny Sitanayah, and Jacob Sorber. 2015. Tragedy of the Coulombs: Federating Energy Storage for Tiny, Intermittently-Powered Sensors. In *Proc. SenSys* (Nov. 1–4). ACM, Seoul, South Korea, 5–16.
[43] Josiah Hester and Jacob Sorber. 2017a. Flicker: Rapid Prototyping for the Batteryless Internet-of-Things. In *Proc. SenSys* (Nov. 6–8). ACM, Delft, The Netherlands, 19:1–19:13.
[44] Josiah Hester and Jacob Sorber. 2017b. The Future of Sensing is Batteryless, Intermittent, and Awesome. In *Proc. SenSys* (Nov. 6–8). ACM, Delft, The Netherlands, 21:1–21:6.
[45] Josiah Hester, Kevin Storer, and Jacob Sorber. 2017. Timely Execution on Intermittently Powered Batteryless Sensors. In *Proc. SenSys* (Nov. 6–8). ACM, Delft, The Netherlands, 17:1–17:13.
[46] Matthew Hicks. 2017. Clank: Architectural Support for Intermittent Computation. In *Proc. ISCA* (June 24–28). ACM, Toronto, ON, Canada, 228–240.
[47] Meng-Ju Hsieh, Jr-Ling Guo, Chin-Yuan Lu, Han-Wei Hsieh, Rong-Hao Liang, and Bing-Yu Chen. 2019. RFTouchPads: Batteryless and Wireless Modular Touch Sensor Pads Based on RFID. In *Proc. UIST*. ACM, New Orleans, LA, US, 999–1011.
[48] Meng-Ju Hsieh, Rong-Hao Liang, Da-Yuan Huang, Jheng-You Ke, and Bing-Yu Chen. 2018. RFIBricks: Interactive Building Blocks Based on RFID. In *Proc. CHI* (April 21–26). ACM, Montréal, QC, Canada, 1–10.
[49] Nick Huber. 2020. Internet of Things: Smart Cities Pick up the Pace. https://www.ft.com/content/140ae3f0-1b6f-11ea-81f0-0c253907d3e0. (Jan. 2020). Last accessed: May 7, 2020.
[50] Chanyou Hwang, Saumay Pushp, Changyoung Koh, Jungpil Yoon, Yunxin Liu, Seungpyo Choi, and Junehwa Song. 2017. RAVEN: Perception-aware Optimization of Power Consumption for Mobile Games. In *Proc. MobiCom*. ACM, Snowbird, UT, USA, 422–434.
[51] Kaori Ikematsu, Masaaki Fukumoto, and Itiro Siio. 2019. Ohmic-Sticker: Force-to-Motion Type Input Device for Capacitive Touch Surface. In *Proc. CHI* (May 4–9). ACM, Glasgow, Scotland, UK, LBW0223:1–LBW0223:6.

[52] Jittrapol Intarasirisawat, Cheesiang Ang, Christos Efstratiou, Luke William Feidhlim Dickens, and Rupert Page. 2019. Exploring the Touch and Motion Features in Game-Based Cognitive Assessments. *ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 3 (Sept. 2019), 87:1–87:25.
[53] Christos Ioannou, Patrick Archard, Eamonn O'Neill, and Christof Lutteroth. 2019. Virtual Performance Augmentation in an Immersive Jump & Run Exergame. In *Proc. CHI* (May 4–9). ACM, Glasgow, Scotland, UK, 158:1–158:15.
[54] Vikram Iyer, Elyas Bayati, Rajalakshmi Nandakumar, Arka Majumdar, and Shyam Gollakota. 2017. Charging a Smartphone Across a Room Using Lasers. *ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 4 (Dec. 2017), 143:1–143:21.
[55] Ravi Jain and John Wullert II. 2002. Challenges: Environmental Design for Pervasive Computing Systems. In *Proc. MobiCom* (Sept. 23–28). ACM, Atlanta, GA, USA, 263–270.
[56] Japan Display Inc. 2016. LPM013M126A 1.28" MIP Reflective Color LTPS TFT LCD. https://www.j-display.com/product/pdf/Datasheet/4LPM013M126A_specification_Ver02.pdf. (2016). Last accessed: Apr. 25, 2020.
[57] Hrishikesh Jayakumar, Arnab Raha, Woo Suk Lee, and Vijay Raghunathan. 2015. Quickrecall: A HW/SW Approach for Computing Across Power Cycles in Transiently Powered Computers. *J. Emerg. Technol. Comput. Syst.* 12, 1 (July 2015), 8:1–8:19.
[58] Asangi Jayatilaka, Quoc Hung Dang, Shengjian Jammy Chen, Renuka Visvanathan, Christophe Fumeaux, and Damith C. Ranasinghe. 2019. Designing Batteryless Wearables for Hospitalized Older People. In *Proc. ISWC*. ACM, London, UK, 91–95.
[59] Haojian Jin, Jingxian Wang, Zhijian Yang, Swarun Kumar, and Jason Hong. 2018. WiSh: Towards a Wireless Shape-aware World using Passive RFIDs. In *Proc. MobiSys*. ACM, Munich, Germany, 428–441.
[60] Kumara Kahatapitiya, Chamod Weerasinghe, Jinal Jayawardhana, Hiranya Kuruppu, Kanchana Thilakarathna, and Dileeka Dias. 2018. Low-power Step Counting Paired with Electromagnetic Energy Harvesting for Wearables. In *Proc. ISWC* (Oct. 8–12). ACM, Singapore, 218–219.
[61] Mustafa Emre Karagozler, Ivan Poupyrev, Gary K. Fedder, and Yuri Suzuki. 2013. Paper Generators: Harvesting Energy from Touching, Rubbing and Sliding. In *Proc. UIST*. ACM, St. Andrews, UK, 23–30.
[62] Keiko Katsuragawa, Ju Wang, Ziyang Shan, Ningshan Ouyang, Omid Abari, and Daniel Vogel. 2019. Tip-Tap: Battery-free Discrete 2D Fingertip Input. In *Proc. UIST*. ACM, New Orleans, LA, US, 1045–1057.
[63] Bryce Kellogg, Aaron Parks, Shyamnath Gollakota, Joshua R. Smith, and David Wetherall. 2014. Wi-Fi Backscatter: Internet Connectivity for RF-Powered Devices. In *Proc. SIGCOMM*. ACM, Chicago, IL, USA, 607–618.
[64] Ben Kenwright. 2012. Fast Efficient Fixed-Size Memory Pool: No Loops and No Overhead. In *Proc. Computation Tools*. IARIA, Nice, France.
[65] Azam Khan. 2011. Swimming Upstream in Sustainable Design. *Interactions* 18, 5 (Sept. 2011), 12—-14.
[66] Yoojung Kim, Arpita Bhattacharya, Julie A. Kientz, and Jin Ha Lee. 2020. "It Should Be a Game for Fun, Not Exercise": Tensions in Designing Health-Related Features for Pokémon GO. In *Proc. CHI* (April 25–30). ACM, Honolulu, HI, USA.
[67] Bran Knowles, Lynne Blair, Mike Hazas, and Stuart Walker. 2013. Exploring Sustainability Research in Computing: Where we Are and Where we go Next. In *Proc. UbiComp* (Sept. 8–12). ACM, Zurich, Switzerland, 305–314.
[68] Vito Kortbeek, Kasım Sinan Yıldırım, Abu Bakar, Jacob Sorber, Josiah Hester, and Przemysław Pawełczak. 2020. Time-sensitive Intermittent Computing Meets Legacy Software. In *Proc. ASPLOS* (March 16–20). ACM, Lausanne, Switzerland, 85––99.
[69] Andre Lam, Ivan Talev, and Jennifer Kim. 2020. Design life-Cycle: University of California, Davis, CA, USA, Department of Design's Undergraduate Students Project Designed by Christina Cogdell. http://www.designlife-cycle.com/nintendo-switch. (2020). Last accessed: Apr. 30, 2020.
[70] Seulki Lee, Bashima Islam, Yubo Luo, and Shahriar Nirjon. 2019. Intermittent Learning: On-Device Machine Learning on Intermittently Powered System. *ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 4 (Dec. 2019), 141:1–141:30.
[71] Dong Li, Feng Ding, Qian Zhang, Run Zhao, Jinshi Zhang, and Dong Wang. 2017. TagController: A Universal Wireless and Battery-free Remote Controller using Passive RFID Tags. In *Proc. MobiQuitous*. ACM, Melbourne, VIC, Australia, 166–175.
[72] Hanchuan Li, Eric Brockmeyer, Elizabeth J. Carter, Josh Fromm, Scott E. Hudson, Shwetak N. Patel, and Alanson Sample. 2016. PaperID: A Technique for Drawing Functional Battery-Free Wireless Interfaces on Paper. In *Proc. CHI* (May 7–12). ACM, San Jose, CA, USA, 5885–5896.
[73] Tianxing Li and Xia Zhou. 2018. Battery-Free Eye Tracker on Glasses. In *Proc. MobiCom* (October 29 — November 2). ACM, New Delhi, India, 67–82.
[74] Yichen Li, Tianxing Li, Xing-Dong Yang Ruchir A. Patel, and Xia Zhou. 2018. Self-Powered Gesture Recognition with Ambient Light. In *Proc. UIST*. ACM, Berlin, Germany, 595–608.
[75] Zhuying Li, Yan Wang, Wei Wang, Weikang Chen, Ti Hoang, Stefan Greuter, and Florian 'Floyd' Mueller. 2019. HeatCraft: Designing Playful Experiences with Ingestible Sensors via Localized Thermal Stimuli. In *Proc. CHI* (May 4–9). ACM, Glasgow, Scotland, UK, 576:1–576:12.
[76] Rong-Hao Liang, Meng-Ju Hsieh, Jheng-You Ke, Jr-Ling Guo, and Bing-Yu Chen. 2018. RFIMatch: Distributed Batteryless Near-Field Identification Using RFID-Tagged Magnet-Biased Reed Switches. In *Proc. UIST*. ACM, Berlin, Germany, 473–483.

[77] Brandon Lucia, Vignesh Balaji, Alexei Colin, Kiwan Maeng, and Emily Ruppel. 2017. Intermittent Computing: Challenges and Opportunities. In *Proc. SNAPL* (May 7–10). Alisomar, CA, USA, 8:1–8:14.
[78] Brandon Lucia and Benjamin Ransford. 2015. A simpler, Safer Programming and Execution Model for Intermittent Systems. In *Proc. PLDI* (Aug. 13–17). ACM, Portland, OR, USA, 575–585.
[79] Dong Ma, Guohao Lan, Mahbub Hassan, Wen Hu, Mushfika Baishakhi Upama, Ashraf Uddin, and Moustafa Youssef. 2019. SolarGest: Ubiquitous and Battery-free Gesture Recognition using Solar Cells. In *Proc. MobiCom* (April 21–25). ACM, Los Cabos, Mexico, 12:1–12:15.
[80] Yunfei Ma, Nicholas Selby, and Fadel Adib. 2017. Drone Relays for Battery-Free Networks. In *Proc. SIGCOMM* (Aug. 21–25). ACM, Los Angeles, CA, USA, 335—-347.
[81] Kiwan Maeng, Alexei Colin, and Brandon Lucia. 2017. Alpaca: Intermittent Execution without Checkpoints. In *Proc. OOPSLA* (Oct. 22–27). ACM, Vancouver, BC, Canada, 96:1–96:30.
[82] Kiwan Maeng, Alexei Colin, and Brandon Lucia. 2018. Adaptive Dynamic Checkpointing for Safe Efficient Intermittent Computing. In *Proc. OSDI* (Oct. 8–10). USENIX, Carlsbad, CA, USA, 129–144.
[83] Kiwan Maeng and Brandon Lucia. 2019. Supporting Peripherals in Intermittent Systems with Just-in-time Checkpoints. In *Proc. PLDI*. ACM, Phoenix, AZ, USA, 1101–1116.
[84] Amjad Yousef Majid, Carlo Delle Donne, Kiwan Maeng, Alexei Colin, Kasım Sinan Yıldırım, Brandon Lucia, and Przemysław Pawełczak. 2020. Dynamic Task-based Intermittent Execution for Energy-harvesting Devices. *ACM Transactions on Sensor Networks* 16, 1 (Feb. 2020), 5:1–5:24.
[85] Jennifer C. Mankoff, Eli Blevis, Alan Borning, Batya Friedman, Susan R. Fussell, Jay Hasbrouck, Allison Woodruff, and Phoebe Sengers. 2007. Environmental Sustainability and Interaction. In *Proc. CHI* (April 28 – May 3). ACM, San Jose, CA, USA, 2121–2124.
[86] Gaia Maselli, Mauro Piva, Giorgia Ramponi, and Deepak Ganesan. 2016. Demo: JoyTag: a battery-less videogame controller exploiting RFID backscattering. In *Proc. MobiCom*. ACM, New York City, NY, USA, 515–516.
[87] Yogesh Kumar Meena, Krishna Seunarine, Deepak Ranjan Sahoo, Simon Robinson, Jennifer Pearson, Chi Zhang, Matt Carnie, Adam Pockett, Andrew Prescott, Suzanne K. Thomas, Harrison Ka Hin Lee, and Matt Jones. 2020. PV-Tiles: Towards Closely-Coupled Photovoltaic and Digital Materials for Useful, Beautiful and Sustainable Interactive Surfaces. In *Proc. CHI* (April 25–30). ACM, Honolulu, HI, USA.
[88] Hashan Roshantha Mendis and Pi-Cheng Hsiu. 2019. Accumulative Display Updating for Intermittent Systems. *ACM Transactions on Embedded Computing Systems* 18, 5s (Oct. 2019), 72:1–72:22.
[89] Geoff V. Merrett and Bashir M. Al-Hashimi. 2017. Energy-Driven Computing: Rethinking the Design of Energy Harvesting Systems. In *Proc. DATE*. IEEE, Lausanne, Switzerland, 960–965.
[90] Evan Mills, Norman Bourassa, Leo Rainer, Jimmy Mai, Arman Shehabi, and Nathaniel Mills. 2019. Toward Greener Gaming: Estimating National Energy Use and Energy Efficiency Potential. *The Computer Games Journal* 8 (Oct. 2019), 157–178.
[91] Florian 'Floyd' Mueller, Richard Byrne, Josh Andres, and Rakesh Patibanda. 2018. Experiencing the Body as Play. In *Proc. CHI* (April 21–26). ACM, Montréal, QC, Canada, 210:1–210:13.
[92] Saman Naderiparizi, Mehrdad Hessar, Vamsi Talla, Shyamnath Gollakota, and Joshua R. Smith. 2018. Towards Battery-Free HD Video Streaming. In *Proc. NSDI* (April 9–11). USENIX, Renton, WA, USA, 233–247.
[93] Saman Naderiparizi, Aaron N. Parks, Zerina Kapetanovic, Benjamin Ransford, and Joshua R. Smith. 2015a. WISPCam: A Battery-Free RFID Camera. In *Proc. IEEE RFID*. IEEE, San Diego, CA, USA, 166–173.
[94] Saman Naderiparizi, Yi Zhao, James Youngquist, Alanson P. Sample, and Joshua R. Smith. 2015b. Self-Localizing Battery-Free Cameras. In *Proc. UbiComp*. ACM, Osaka, Japan, 445–449.
[95] Yuji Nakamura. 2019. Peak Video Game? Top Analyst Sees Industry Slumping in 2019. https://www.bloomberg.com/news/articles/2019-01-23/peak-video-game-top-analyst-sees-industry-slumping-in-2019. (Jan. 2019). Last accessed: Jan. 7, 2020.
[96] Matteo Nardello, Harsh Desai, Davide Brunelli, and Brandon Lucia. 2019. Camaroptera: a Batteryless Long-Range Remote Visual Sensing System. In *Proc. ENSsys*. ACM, New York, NY, USA, 8–14.
[97] Phuc Nguyen, Ufuk Muncuk, Ashwin Ashok, Kaushik Roy Chowdhury, Marco Gruteser, and Tam Vu. 2016. Battery-Free Identification Token for Touch Sensing Devices. In *Proc. SenSys*. ACM, Stanford, CA, USA, 109–122.
[98] Nintendo Co., Ltd. 2019. Dedicated Video Game Sales Units. https://www.nintendo.co.jp/ir/en/finance/hard_soft/index.html. (Sept. 2019). Last accessed: Apr. 30, 2020.
[99] Nintendo Co., Ltd. 2020. Nintendo Game Boy. https://en.wikipedia.org/wiki/Game_Boy. (July 2020). Last accessed: Jul. 23, 2020.
[100] Sam Nussey and Christopher Cushing. 2020. Nintendo Seen Extending Profit Streak as Housebound Consumers Switch On. https://www.reuters.com/article/us-nintendo-results-preview/nintendo-seen-extending-profit-streak-as-housebound-consumers-switch-on-idUSKBN22C0B4. (April 2020). Last accessed: May 15, 2020.
[101] Netta Ofer, Idan David, Hadas Erel, and Oren Zuckerman. 2019. Coding for Outdoor Play: A Coding Platform for Children to Invent and Enhance Outdoor Play Experiences. In *Proc. CHI* (May 4–9). ACM, Glasgow, Scotland, UK, 1–12.

[102] Kazuya Oharada, Buntarou Shizuki, and Shin Takahashi. 2017. AccelTag: A Passive Smart ID Tag With an Acceleration Sensor for Interactive Applications. In *Proc. UIST Adjunct*. ACM, Québec City, Canada, 63–64.
[103] Panasonic Electric Works Europe AG. 2019. AM-1417CA Amorphous Silicon Solar Cell. https://www.panasonic-electric-works.com/cps/rde/xbcr/pew_eu_en/ca_amorton_solar_cells_en.pdf. (2019). Last accessed: Apr. 25, 2020.
[104] Panic Inc. 2020. Playdate Console Home Page. https://play.date. (March 2020). Last accessed: May 2, 2020.
[105] Aaron N. Parks, Angli Liu, Shyamnath Gollakota, and Joshua R. Smith. 2014. Turbocharging Ambient Backscatter Communication. In *Proc. SIGCOMM*. ACM, Chicago, IL, USA, 619–630.
[106] Matthai Philipose, Joshua R. Smith, Bing Jiang, Alexander Mamishev, Sumit Roy, and Kishor Sundara-Rajan. 2005. Battery-Free Wireless Identification and Sensing. *IEEE Pervasive Comput.* 4, 1 (Jan.–Mar. 2005), 37–45.
[107] R. Venkatesha Prasad, Shruti Devasenapathy, Vijay S. Rao, and Javad Vazifehdan. 2014. Reincarnation in the Ambiance: Devices and Networks with Energy Harvesting. *IEEE Commun. Surveys Tuts.* 11, 1 (First Quarter 2014), 195–213.
[108] United Nations Environment Programme. 2020. Playing for the Planet Consortium. https://playing4theplanet.org. (April 2020). Last accessed: Apr. 28, 2020.
[109] Benjamin Ransford, Jacob Sorber, and Kevin Fu. 2011. Mementos: System Support for Long-running Computation on RFID-scale Devices. In *Proc. ASPLOS* (March 5–11). ACM, Newport Beach, CA, USA, 159–170.
[110] Emily Ruppel and Brandon Lucia. 2019. Transactional Concurrency Control for Intermittent, Energy-Harvesting Computing Systems. In *Proc. PLDI*. ACM, Phoenix, AZ, USA, 1085–1100.
[111] Kimiko Ryokai, Peiqi Su, Eungchan Kim, and Bob Rollins. 2014. EnergyBugs: Energy Harvesting Wearables for Children. In *Proc. CHI* (Apr. 26 – May 1). ACM, Toronto, ON, Canada, 1039–1048.
[112] Ali Saffari, Mehrdad Hessar, Saman Naderiparizi, and Joshua R. Smith. 2019. Battery-Free Wireless Video Streaming Camera System. In *Proc. RFID*. IEEE, Phoenix, AZ, USA.
[113] Saleae. 2020. Saleae Logic Pro 8 Analyzer. http://downloads.saleae.com/specs/Logic+Pro+8+Data+Sheet.pdf. (2020). Last accessed: Jun. 22, 2020.
[114] Alanson P. Sample, Daniel J. Yeager, Pauline S. Powledge, Alexander V. Mamishev, and Joshua R. Smith. 2008. Design of an RFID-based battery-free programmable sensing platform. *IEEE Trans. Instrum. Meas.* 57, 11 (Nov. 2008), 2608–2615.
[115] Takuya Sasatani, Chouchang Jack Yang, Matthew J. Chabalko, Yoshihiro Kawahara, and Alanson P. Sample. 2018. Room-Wide Wireless Charging and Load-Modulation Communication via Quasistatic Cavity Resonance. *ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 4 (Dec. 2018), 188:1–188:23.
[116] Seth Schiesel. 2020. For the Uninitiated and Bored, an Introduction to the World of Gaming. https://www.nytimes.com/2020/04/01/arts/gaming-introduction-basics-quarantine-coronavirus.html. (April 2020). Last accessed: Apr. 28, 2020.
[117] Oliver Schneider, Jotaro Shigeyama, Robert Kovacs, Thijs Roumen, Sebastian Marwecki, Nico Boeckhoff, Daniel Amadeus Gloeckner, Jonas Bounama, and Patrick Baudisch. 2018. DualPanto: A Haptic Device that Enables Blind Users to Continuously Interact with Virtual Worlds. In *Proc. UIST*. ACM, Berlin, Germany, 877–887.
[118] Jason Schreier. 2020. Gaming Sales Are Up, but Production Is Down. https://www.nytimes.com/2020/04/21/technology/personaltech/coronavirus-video-game-production.html. (April 2020). Last accessed: Apr. 29, 2020.
[119] Semiconductor Components Industries LLC. 2017. NSR1030QMUTWG Schottky Full Diode Bridge. https://www.onsemi.com/pub/Collateral/NSR1030QMU-D.PDF. (2017). Last accessed: Apr. 25, 2020.
[120] Semtech. 2012. SX1503 4/8/16 Channel Low Voltage GPIO with NINT and NRESET. https://www.semtech.com/products/smart-sensing/io-expanders/sx1503. (2012). Last accessed: Apr. 25, 2020.
[121] Rishi Shukla, Neev Kiran, Rui Wang, Jeremy Gummeson, and Sunghoon Ivan Lee. 2019. SkinnyPower: Enabling Batteryless Wearable Sensors via Intra-Body Power Transfer. In *Proc. SenSys* (Nov. 10–13). ACM, New York City, NY, USA, 55–67.
[122] Philip Sparks. 2017. *The Route to a Trillion Devices: The Outlook for IoT investment to 2035.* Technical Report. ARM Limited. https://pages.arm.com/rs/312-SAX-488/images/Arm-The-route-to-trillion-devices_2018.pdf.
[123] STMicroelectronics. 2018. X-NUCLEO-LPM01A Nucleo Expansion Board for Power Consumption Measurement. https://www.st.com/en/evaluation-tools/x-nucleo-lpm01a.html. (March 2018). Last accessed: Jun. 22, 2020.
[124] Kazunobu Sumiya, Takuya Sasatani, Yuki Nishizawa, Kenji Tsushio, Yoshiaki Narusue, and Yoshihiro Kawahara. 2019. Alvus: A Reconfigurable 2-D Wireless Charging System. *ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 2 (June 2019), 68:1–68:29.
[125] Vamsi Talla, Mehrdad Hassar, Bryce Kellogg, Ali Najafi, Joshua R. Smith, and Shyam Gollakota. 2017a. LoRa Backscatter: Enabling the Vision of Ubiquitous Connectivity. *ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 3 (Sept. 2017), 105:1–105:24.
[126] Vamsi Talla, Bryce Kellogg, Shyamnath Gollakota, and Joshua R Smith. 2017b. Battery-free Cellphone. *ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 2 (June 2017), 25:1–25:19.
[127] Haydn Taylor. 2020. COVID-19: The State of the Games Industry. https://www.gamesindustry.biz/articles/2020-04-09-covid-19-the-state-of-the-games-industry. (April 2020). Last accessed: Apr. 28, 2020.
[128] Texas Instruments Inc. 2013. BQ25570 Ultra Low power Harvester power Management IC with Boost Charger, and Nanopower Buck Converter. https://www.ti.com/lit/ds/symlink/bq25570.pdf. (2013). Last accessed: Apr. 25, 2020.

[129] Texas Instruments Inc. 2016. TPS61099 0.7 V_{in} Synchronous Boost Converter with 800 nA Ultra-Low Quiescent Current. http://www.ti.com/lit/ds/symlink/tps61099.pdf. (2016). Last accessed: Apr. 25, 2020.
[130] Texas Instruments Inc. 2017. MSP430FR59xx Mixed-Signal Microcontrollers (Rev. F). http://www.ti.com/lit/ds/symlink/msp430fr5969.pdf. (March 2017). Last accessed: May 1, 2020.
[131] Bill Tomlinson, M. Six Silberman, Don Patterson, Yue Pan, and Eli Blevis. 2012. Collapse Informatics: Augmenting the Sustainability & ICT4D Discourse in HCI. In *Proc. CHI*. ACM, Austin, TX, USA, 655–664.
[132] Hoang Truong, Shuo Zhang, Ufuk Muncuk, Phuc Nguyen, Nam Bui, Anh Nguyen, Qin Lv, Kaushik Chowdhury, Thang Dinh, and Tam Vu. 2018. CapBand: Battery-free Successive Capacitance Sensing Wristband for Hand Gesture Recognition. In *Proc. SenSys* (Nov. 4–7). ACM, Shenzhen, China, 54–67.
[133] UNI-T. 2020. UT383 Mini Light Meter. https://www.uni-trend.com/html/product/Environmental/Environmental_Tester/Mini/UT383.html. (2020). Last accessed: Jun. 22, 2020.
[134] University of Michigan, MI, USA. 2011. UMich Moo GitHub Page. https://github.com/spqr/umichmoo. (Feb. 2011). Last accessed: Apr. 19, 2020.
[135] University of Washington, Seattle, WA, USA. 2010. Wireless Identification and Sensing Platform GitHub Page. https://github.com/wisp. (Nov. 2010). Last accessed: Apr. 19, 2020.
[136] Valve Inc. 2020. Source 3D Game Engine. https://developer.valvesoftware.com/wiki/source. (March 2020). Last accessed: May 2, 2020.
[137] Joel Van Der Woude and Matthew Hicks. 2016. Intermittent Computation Without Hardware Support or Programmer Intervention. In *Proc. OSDI* (Nov. 2–4). ACM, Savannah, GA, USA, 17–32.
[138] Virag Varga, Gergely Vakulya, Alanson Sample, and Thomas R. Gross. 2017a. Enabling Interactive Infrastructure with Body Channel Communication. *ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 4 (Dec. 2017), 169:1–169:.
[139] Virag Varga, Gergely Vakulya, Alanson Sample, and Thomas R. Gross. 2017b. Playful Interactions with Body Channel Communication: Conquer it!. In *Proc. UIST Adjunct*. ACM, Québec City, Canada, 81–82.
[140] Anandghan Waghmare, Qiuyue Xue, Dingtian Zhang, Yuhui Zhao, Shivan Mittal, Nivedita Arora, Ceara Byrne, Thad Starner, and Gregory D. Abowd. 2020. UbiquiTouch: Self Sustaining Ubiquitous Touch Interfaces. *ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 1 (March 2020), 27:1–27:22.
[141] Ju Wang, Liqiong Chang, Omid Abari, and Srinivasan Keshav. 2019a. Are RFID Sensing Systems Ready for the Real World?. In *Proc. MobiSys*. ACM, Seoul, Korea, 366–377.
[142] Jingxian Wang, Chengfeng Pan, Haojian Jin, Vaibhav Singh, Yash Jain, Jason Hong, Carmel Majidi, and Swarun Kumar. 2019b. RFID Tattoo: A Wireless Platform for Speech Recognition. *ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 4 (Dec. 2019), 155:1–155:24.
[143] Katelyn Wiley, Sarah Vedress, and Regan Mandryk. 2020. How Points and Theme Affect Performance and Experience in a Gamified Cognitive Task. In *Proc. CHI* (April 25–30). ACM, Honolulu, HI, USA.
[144] Harrison Williams, Xun Jian, and Matthew Hicks. 2020. Forget Failure: Exploiting SRAM Data Remanence for Low-overhead Intermittent Computation. In *Proc. ASPLOS* (March 16–20). ACM, Lausanne, Switzerland, 53—-67.
[145] Allison Woodruff, Jay Hasbrouck, and Sally Augustin. 2008. A Bright Green Perspective on Sustainable Choices. In *Proc. CHI*. ACM, Florence, Italy, 313–322.
[146] Chenren Xu, Lei Yang, and Pengyu Zhang. 2018a. Practical Backscatter Communication Systems for Battery-Free Internet of Things. *IEEE Signal Process. Mag.* 35, 5 (Sept. 2018), 16–27.
[147] Jian Xu, Suwen Zhu, Aruna Balasubramanian, Xiaojun Bi, and Roy Shilkrot. 2018b. Ultra-Low-Power Mode for Screenless Mobile Interaction. In *Proc. UIST*. ACM, Berlin, Germany, 557–568.
[148] Xieyang Xu, Yang Shen, Junrui Yang, Chenren Xu, Guobin Shen, Guojun Chen, and Yunzhe Ni. 2017. PassiveVLC: Enabling Practical Visible Light Backscatter Communication for Battery-free IoT Applications. In *Proc. MobiCom* (Oct. 16–20). ACM, Snowbird, UT, USA, 180–192.
[149] Wataru Yamada, Hiroyuki Manabe, and Daizo Ikeda. 2018. CamTrackPoint: Camera-Based Pointing Stick Using Transmitted Light through Finger. In *Proc. UIST*. ACM, Berlin, Germany, 313–320.
[150] Kasım Sinan Yıldırım, Amjad Yousef Majid, Dimitris Patoukas, Koen Schaper, Przemysław Pawełczak, and Josiah Hester. 2018. InK: Reactive Kernel for Tiny Batteryless Sensors. In *Proc. SenSys* (Nov. 4–7). ACM, Shenzhen, China, 41–53.
[151] Neng-Hao Yu, Sung-Sheng Tsai, I-Chun Hsiao, Dian-Je Tsai, Meng-Han Lee, Mike Y. Chen, and Yi-Ping Hung. 2011. Clip-on Gadgets: Expanding Multi-touch Interaction Area with Unpowered Tactile Controls. In *Proc. UIST* (Oct. 16–19). ACM, Santa Barbara, CA, USA, 367–371.
[152] Tom Zeller Jr. 2010. Green Guide to Electronics Is Disputed, but Influential. https://www.nytimes.com/2010/01/11/business/energy-environment/11green.html. (Jan. 2010). Last accessed: Apr. 30, 2020.
[153] ZF Friedrichshafen AG. 2015. AFIG-0007 Energy Harvesting Generator. https://switches-sensors.zf.com/us/wp-content/uploads/sites/7/2019/11/19_10_16-TS_AFIG-0007.pdf [specification], https://switches-sensors.zf.com/product/energy-harvesting-generators/ [summary]. (2015). Last accessed: Apr. 25, 2020.

[154] Chi Zhang, Sidharth Kumar, and Dinesh Bharadia. 2019b. Capterry: Scalable Battery-like Room-level Wireless Power. In *Proc. MobiSys*. ACM, Seoul, Korea, 1–13.

[155] Yang Zhang, Yasha Iravantchi, Haojian Jin, Swarun Kumar, and Chris Harrison. 2019a. Sozu: Self-Powered Radio Tags for Building-Scale Activity Sensing. In *Proc. UIST*. ACM, New Orleans, LA, USA, 973–985.

[156] Chen Zhao, Sam Yisrael, Joshua R. Smith, and Shwetak N. Patel. 2014. Powering Wireless Sensor Nodes with Ambient Temperature Changes. In *Proc. UbiComp* (Sept. 13–17). ACM, Seattle, WA, USA, 383––387.