Simba: A Unified Framework to Explore and Facilitate the Design of Battery-Free Systems

Hannah Brunner
hannah.brunner@tugraz.at
Graz University of Technology
Graz, Austria

Jasper de Winkel
jasper.dewinkel@jdew.nl
Delft University of Technology
Delft, Netherlands

Carlo Alberto Boano
cboano@tugraz.at
Graz University of Technology
Graz, Austria

Przemysław Pawełczak
p.pawelczak@tudelft.nl
Delft University of Technology
Delft, Netherlands

Kay Römer
roemer@tugraz.at
Graz University of Technology
Graz, Austria

ABSTRACT

Battery-free sensing devices have gained growing popularity as they can operate relying solely on harvested energy and environmentally friendly capacitors. However, despite the increasing number of battery-free solutions, their design remains a difficult task. In fact, the limited energy storage capacity and the resulting coupling between energy supply and demand introduce new design trade-offs that cannot be explored using conventional tools that consider a constant power supply. To enable fast design space exploration and facilitate the development of battery-free systems, we introduce Simba, an open-source simulation framework that allows to investigate in detail the complex interplay between various device components. We demonstrate the benefits of Simba in two case studies, evaluated experimentally, targeting real-world, state-of-the-art battery-free devices. First, we illustrate how Simba can explore the dependencies between different component configurations and assess their impact on the overall system performance. Among others, we show that changing the storage capacity or slightly modifying the load behavior can improve data throughput by a factor of up to 5.1x and 9.7x, respectively. Second, we present how Simba allows to automatically select key parameters that optimize the operations of a battery-free system (e.g., its checkpointing mechanism), and showcase how Simba enables performance evaluations based on real-world energy harvesting traces.

CCS CONCEPTS

• **Computer systems organization → Embedded systems**.

KEYWORDS

Battery-Free Systems, Sensor Nodes, Simulator

1 INTRODUCTION

The development of battery-free sensor nodes is an attempt to cope with the extensive scale of the Internet of Things (IoT). A projected number of tens of billions of connected devices [52] requires long-lasting, maintenance-free operation, as replacing and disposing batteries at this volume is impractical, expensive, and not sustainable. Battery-free devices are equipped with energy harvesters to extract energy from ambient sources (such as light, temperature, or vibrations) and employ capacitors as energy storage in order to provide energy autonomy and independence from bulky batteries. This freedom enables the development of devices with tiny form factors that can be deployed in harsh environments and operate without maintenance for years. Given these advantages, a large number of battery-free IoT systems have been developed, including radio frequency identification tags [45], Bluetooth Low Energy beacons [18, 22, 33, 46], sensing platforms [1, 14, 16, 48, 58], a batteryless handheld gaming console [13], and a wireless camera [23].

Designing battery-free systems is complex. Despite the success of these platforms and the insights gained from their development, the design of battery-free devices remains a rather complex task.

Large design space. All the mentioned systems differ largely in terms of harvestable energy, energy storage capacity, and power demand of the sensor element. For example, depending on the harvesting source, the incoming power can be in the order of hundreds of mW [58] down to tens of µW [49], with large differences in temporal availability. Consequently, the employed storage capacitors in battery-free devices span from tiny ceramic capacitors to large supercapacitors, while the devices' operations range from sporadic, intermittent sensing [1] to powerful radio transmissions [23] and resource-intensive machine learning [6]. In light of this heterogeneity and the diversity across IoT applications in general, developers are confronted with a large design space. Thus, finding a solution that best meets the application requirements can be challenging.

Energy dynamics increase complexity. Additionally, unique challenges emerge from the dynamics of harvested energy that are not present when dealing with a constant power supply [50]. The performance and power consumption of a device are tightly coupled with the energy income and vice versa [27]. Furthermore, if energy is scarce, devices might have to deal with power failures and operate intermittently, i.e., they turn on and off as energy is available and apply state-retention mechanisms – so-called checkpointing – to ensure forward progress [50]. These circumstances make the design and evaluation of battery-free systems a complex and tedious task, as common development tools typically assume a constant power supply and thus cannot capture energy-driven operations.

Existing tools and models. In order to facilitate development, a number of tools that specifically target battery-free systems have been introduced, including testbeds [21], debuggers [12, 15], and energy emulators [27]. While these are valuable to validate and debug specific designs, they require access to real hardware and have limited capabilities w.r.t. design space exploration. To better understand

979-8-3503-6201-5/24/$31.00 ©2024 IEEE
DOI 10.1109/IPSN61024.2024.00016

battery-free system designs, other works have focused on modeling and simulating different aspects of battery-free systems [20, 51]. The community has proposed models of specific device components (e.g., the energy storage [3, 7, 9], the energy source [35, 36, 53, 55], or the processing unit [2]), as well as models of the state-retention process [43]. Although insightful, the modelling of individual components does not allow to assess the non-trivial, energy-driven dependencies within a battery-free system. For example, it has been shown that the choice of the optimal state-retention mechanism depends on the energy harvesting source [4] or that small changes in the energy storage capacity can significantly improve the device performance [59]. While previous works point out the necessity to model the *entire system*, these works consider only certain architectures [4, 59] or target specific platforms [20].

The need for a unified simulation framework. We thus argue that there is a need for a *unified simulation framework* that combines a variety of device component models and energy harvesting datasets within a single tool. Such a unified framework would allow to investigate the complex interactions between different hardware components, gain a better understanding of dependencies in battery-free systems, and inform future design decisions. It would further accelerate development, enable evaluation and comparison of designs in a fair and repeatable manner, as well as foster research on battery-free systems without the need for specific hardware.

Our contributions. In this work, we present `Simba`, a simulation framework for battery-free devices. `Simba` is built in a modular and extensible manner, which allows the integration of various component models and datasets in a single simulation core. We demonstrate its abilities in two case studies, targeting two different state-of-the-art battery-free devices. Specifically, we use `Simba` to *explore* the dependencies between the different parts of Botoks [14], a simple, battery-free sensing device, and show that small changes in each component can have a significant impact on the overall performance. For example, changing the storage capacity or slightly modifying the load behavior can improve data throughput by a factor of up to $5.2x$ and $9.7x$, respectively, while adjustment in the device's voltage conversion circuit largely affects availability in challenging energy harvesting scenarios. Furthermore, we use a (comparably powerful) handheld console (the Battery-free Game Boy [13]) to show how `Simba` can *facilitate* the design process by *enabling an automatic parameter selection* to optimize the device's checkpoint mechanism. To support our findings, we evaluate `Simba` experimentally and complement the case studies' simulation results with measurement data. We further make `Simba` available open-source[1] for the community to encourage and facilitate research on battery-free systems without the need for specific hardware.

2 COMPONENTS OF A BATTERY-FREE SENSOR NODE AND THEIR DEPENDENCIES

A typical architecture of a battery-free device is depicted in Fig. 1 and consists of an energy harvester, an energy storage (i.e., capacitor), a load, and (optional) voltage converters.

[1]The simulation framework, its tools, examples, and documentation, as well as artifacts data are available at: https://github.com/LENS-TUGraz/simba.

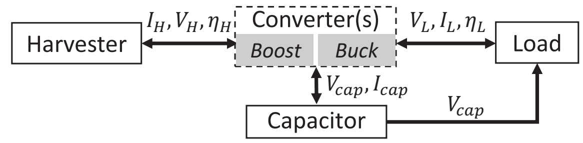

Figure 1: Architecture of a typical battery-free sensor node.

The *energy harvester* converts ambient energy of different sources (e.g., light, movement, or temperature gradients) into electrical energy and typically provides power in the range of hundreds of µW to tens of mW, depending on size, energy source, and environmental conditions. Note that the level of the harvester's voltage V_H can vary vastly and might necessitate voltage conversion. Furthermore, many harvesters exhibit distinct, non-linear IV characteristics, which means that the harvesting current I_H (along with the output power) depends on the applied voltage V_H. Since the instantaneous harvested power is often not sufficient to directly drive a typical sensor node, a capacitor is employed as energy storage between the harvester and the load.

The *capacitor* buffers incoming energy and thus allows the load to operate even if its power demand exceeds the momentarily harvested power. Capacitors in existing battery-free systems range from tiny ceramic capacitors (e.g., $10\,\mu F$) to large supercapacitors (e.g., $1-10\,F$). The choice of storage capacitance is mostly dictated by the application requirements, the harvesting potential, as well as space constraints. Given their size and the amount of incoming energy, capacitors are either used to buffer enough energy to constantly supply the load, even in periods of energy absence, or to collect small amounts of energy to operate the load in bursts (i.e., intermittent operation). In the latter case, the energy storage capacity highly affects the behavior of the device: larger storage allows longer sustained operation, but yields increased charging times; devices with smaller storage are more reactive, but can only operate for short periods. For capacitors, the energy storage capacity depends on its capacitance and the used voltage range (i.e., $E_{\text{cap}} = \frac{C}{2}(V^2_{\text{cap, max}} - V^2_{\text{cap, min}})$), where the voltage range is largely depending on the employed converter architecture.

Battery-free devices employ either a *converter-less* or *converter-based* architecture. In converter-less systems, the harvester, capacitor, and load are *directly coupled*. While these systems avoid converter losses, are very cheap, and can be built in tiny form factors, they are inflexible and exhibit many dependencies that can cause inefficiencies. More specifically, all components share the same operating point ($V_H = V_{\text{cap}} = V_L$). Thus, the harvester's output voltage must be compatible with the load's voltage specification, as the load can only operate within certain voltage limits ($V_L \approx 1.8\ldots3.3\,V$ for typical MCUs). Furthermore, the energy harvesting performance (driven by V_H) is directly dependent on the capacitor's state-of-charge ($\propto V_{\text{cap}}$) and consequently on the load's power consumption (I_L). In a converter-based system, instead, one or more voltage converters are used to decouple the operating points of the harvester, capacitor, and load. For example, a boost converter can be placed between the harvester and the capacitor such that the harvester can be operated at its optimal voltage (i.e., the maximum power point [21]) to maximize the power output. As a result, the harvester must not match the load's specifications, as the capacitor can be charged to any voltage level ($V_{\text{cap}} > V_H$). Additionally, a buck converter can be used to separate the capacitor from the load. This way, the capacitor can be charged higher than the load's maximum operating voltage to increase the storage capacity utilization. Furthermore, the buck converter supplies the load with a constant (minimal) operating voltage, which is typically more efficient. Note that converter-based architectures are often more expensive and also complex to model, as (non-linear) converter efficiencies (η_H, η_L) and quiescent currents have to be considered.

Finally, the *load* is typically a sensor node consisting of a processing unit (MCU), sensors, and a wireless radio. The load applies energy-driven computing [50], i.e., it acts according to energy availability, and typically operates either energy-neutral or intermittently. In energy-neutral operation, the load adapts its power consumption (e.g., duty cycle) to balance energy demand and supply, avoiding power failures and ensuring continuous operation. If energy is very scarce, the load must accept power failures and operate intermittently but might employ state-retention mechanisms (e.g., checkpointing) to guarantee the application's correct forward progress. In either configuration, the load's performance highly depends on the energy budget. For example, the load might experience power failures (if $V_L < V_{L,min}$) or it may dynamically adjust its current consumption I_L depending on the capacitor voltage V_{cap}.

3 THE SIMBA FRAMEWORK

In this section, we introduce the Simba simulation framework and explain how it can capture all dependencies while keeping a generic and modular structure.

3.1 Architecture and Design Rationale

Simba's overall architecture is highly modular and hardware-agnostic (as depicted in Fig. 2) and allows to study the performance of a battery-free sensor node as well as the interactions between its device components in a repeatable and easy way.

Since we target a typical battery-free sensor node as described in Sec. 2, the sensor node model consists of four modules: a `capacitor`, a `harvester`, a `converter`, and a `load`. For each module, the user can choose from a set of *module implementations*. The module implementations describe the (real-world) hardware components and can be extremely versatile (see Sec. 3.4). For example, they might contain simple representations of ideal components, complex analytical models, or experimental data (e.g., power traces from energy harvesting devices or microcontrollers). For each component, *module factories* create a single *module instance* based on a *module configuration*, which selects the desired module implementation and configures its parameters accordingly. The module instances use pre-defined interfaces to interact with the simulation core (see Sec. 3.2) and store logging information (e.g., component state, voltage level, etc.).

In order to explore design trade-offs or to compare the performance of different components, a *trade-off exploration tool* can readily replace or re-configure the individual module instances and retrieve performance metrics from their logs. For example, in Sec. 5 we use this tool to obtain checkpointing-related performance metrics as a function of various capacitor and converter configurations and derive optimal device settings from these results.

3.2 Simulation Principle

Simba's simulation core implements a carefully designed discrete-time simulation of the sensor node's behavior that considers the complete set of dependencies described in Sec. 2 (and in Fig. 1).

Basic system model. In battery-free systems, the capacitor's state of charge ($\propto V_{\text{cap}}$) is often extremely dynamic, as the incoming current and power consumption can change rapidly and the energy storage capacity is comparably small. Since V_{cap} affects both the harvester's and load's efficiency, the system must be simulated in a closed-loop and with small time granularity.

The simulation procedure is thus centered around the capacitor's energy state, where its current-voltage relation is defined as

$$V_{\text{cap}}(t_0 + T) = \frac{1}{C}\int_{t_0}^{t_0+T} (I_{\text{cap}}(t) - I_{\text{leak}}(t))\, dt + V_{\text{cap}}(t_0), \quad (1)$$

where $I_{\text{cap}}(t)$ is the instantaneous current flowing into or out of the capacitor and is assumed to be constant within T, while $I_{\text{leak}}(t)$ describes the internal capacitor losses (e.g., due to self-discharge). The current flow $I_{\text{cap}}(t)$ is the difference between the incoming current $I_{\text{cap,in}}$ and the outgoing current $I_{\text{cap,out}}$ as well as the converter's quiescent currents $I_{\text{quiescent}}$ and is given by

$$I_{\text{cap}}(t) = I_{\text{cap,in}}(t) - I_{\text{cap,out}}(t) - I_{\text{quiescent}}(t). \quad (2)$$

Note that the harvester and the capacitor might be decoupled and operate at different voltage levels. Hence, the current flowing into the capacitor $I_{cap,in}$ is not equal to the harvesting current I_H. Accounting for the different operating points and the converter's boost efficiency η_H, $I_{cap,in}$ can be calculated as

$$I_{cap,in}(t) = I_H(t)\eta_H(t)\frac{V_H(t)}{V_{cap}(t)}. \quad (3)$$

The same principle applies to the buck converter stage between capacitor and load, yielding $I_{cap,out}$ as

$$I_{cap,out}(t) = I_L(t)\frac{1}{\eta_L(t)}\frac{V_L(t)}{V_{cap}(t)}. \quad (4)$$

$I_{\text{cap,in}} = f(I_H, V_H, V_{\text{cap}}, \eta_H)$ and $I_{\text{cap,out}} = f(I_L, V_L, V_{\text{cap}}, \eta_L)$ are strongly affected by the choice of device components and must be calculated carefully while considering all dependencies described in Sec. 2. For example, the harvesting current I_H does not only depend on the harvester (and its environmental conditions), but is strongly affected by the applied voltage V_H, which, in turn, is set by the converter and depending on the capacitor's voltage V_{cap}.

Module interfaces and simulation core. Simba's extendable plug-and-play architecture defines consistent interfaces between different modules of the same type. The simulation core of Simba uses these interfaces to interact with each module and performs a discrete-time simulation as sketched in Alg. 1. After each time step T, the simulation core retrieves the modules' states to compute I_{cap} (according to (2), (3), and (4)) and instructs the `capacitor`, `harvester`, `converter`, and `load` modules to update their status accordingly. Note that the `capacitor` module itself is responsible for updating its state of charge according to (1), as this allows

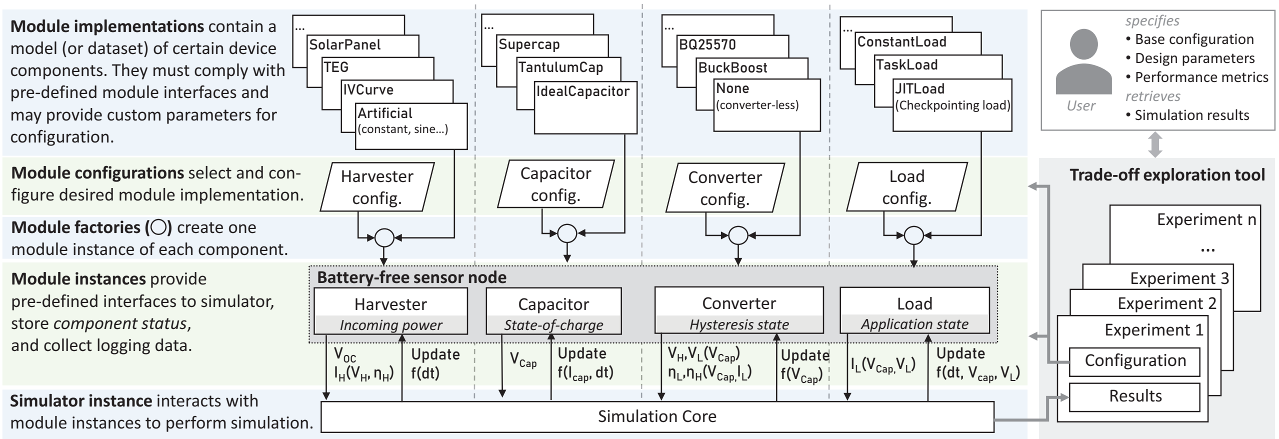

Figure 2: The modular and extensible architecture of `Simba` allows to simulate any combination of device components (i.e., `capacitor`, `harvester`, `converter`, and `load`). A trade-off exploration tool sets up simulator instances according to the user-defined design space, runs the simulations concurrently, and retrieves the requested performance metrics for each configuration.

a simulator-agnostic implementation of the capacitor's leakage behavior (i.e., $I_{\text{leak}}(V_{\text{cap}}, t)$).

Algorithm 1 `Simba`'s simulation principle.

Reset modules to start conditions, $t = 0$
while $t < t_{\text{sim,max}}$ **do**
  // Retrieve capacitor's state-of charge
  $V_{\text{cap}} \leftarrow f(\texttt{Capacitor})$

  // Retrieve harvesting-related values
  $V_H \leftarrow f(\texttt{Converter}, V_{\text{cap}})$
  $I_H \leftarrow f(\texttt{Harvester}, t, V_H)$
  $\eta_H \leftarrow f(\texttt{Converter}, V_H, I_H)$

  // Retrieve load-related values
  $V_L \leftarrow f(\texttt{Converter}, V_{\text{cap}})$
  $I_L \leftarrow f(\texttt{Load}, V_L, V_{\text{cap}})$
  $\eta_L \leftarrow f(\texttt{Converter}, V_{\text{cap}}, I_L)$
  $I_{\text{quiescent}} \leftarrow f(\texttt{Converter}, V_{\text{cap}})$

  // Compute current flow from/into capacitor
  $I_{\text{cap}} = I_H \cdot \frac{V_H}{V_{\text{cap}}} \cdot \eta_H - I_L \cdot \frac{V_L}{V_{\text{cap}}} \cdot \frac{1}{\eta_L} - I_{\text{quiescent}}$

  // Update modules' states
  $\texttt{Capacitor} \leftarrow f(\texttt{Capacitor}, I_{\text{cap}}, T)$
  $\texttt{Converter} \leftarrow f(\texttt{Converter}, V_{\text{cap}})$
  $\texttt{Load} \leftarrow f(\texttt{Load}, V_{\text{cap}}, V_L, T)$
  $\texttt{Harvester} \leftarrow f(\texttt{Harvester}, T)$

  // Compute timestep until next update
  $T \leftarrow f(\texttt{Capacitor}, \texttt{Converter}, \texttt{Load}, \texttt{Harvester})$
  $t = t + \max(T_{\min}, \min(T, T_{\max}))$
end while

3.3 Implementation Details: Simulator

`Simba` is implemented using *Python*, which provides flexibility, enables quick development of new modules, and offers established support for data processing and visualization.

Timestep computation. The simulation is based on discrete (integer) time intervals, where the minimum timestep T_{min} is configurable ($T_{\text{min,default}} = 1\ \mu s$). Furthermore, the timestep T is variable and computed during run-time to increase simulation speed. More specifically, it is set to the time of the next update within any of the submodules. For example, the load might change its power consumption if its application schedules a different task, while the harvester might change its current output due to environmental changes. Additionally, users can specify a maximum timestep $T_{\max}$ to trade simulation speed for accuracy.

Logging. `Simba` adopts a two-layered logging mechanism. The simulation core can be instructed to log generic information (i.e., any values available in Alg. 1), allowing quick comparison of systems despite different module implementations. Furthermore, each module can log fine-grained, component-specific data and derive statistics that are made available to the trade-off-exploration tool.

Trade-off exploration tool. `Simba` provides a trade-off exploration tool that allows to automatically run simulations in a user-defined design space and retrieves the desired performance metrics from the modules' logs. More specifically, the users can specify a base configuration (of each module) and an arbitrary number of design parameters as well as performance metrics they want to explore. Note that the parameters and metrics are not restricted to a single module: iterating over any component/parameter combination is possible. The trade-off exploration tool permutes the parameters, sets up the simulation core accordingly, and runs the simulations concurrently (i.e., using multiprocessing) to speed up the exploration process.

The obtained results can then be used to optimize the system w.r.t. a figure of merit. In battery-free systems, optimization objectives (e.g., overall efficiency, availability, data throughput etc.) and constraints (e.g., physical size, maximum energy income etc.) are extremely application-specific and often require multi-objective decision-making. This is outside the scope of this work, but we give a simple example of how to use the trade-off exploration tool to optimize a system's parameters in Sec. 5.

3.4 Implementation Details: Modules

`Simba` provides several module implementations for each device component. These include simplistic models and generic representations of typical components used in battery-free systems (i.e., to foster research and design space exploration), but also datasets and models from real-world components (i.e., to facilitate the design and validation of battery-free sensor nodes without access to real hardware). The provided modules (see Fig. 2) are not exhaustive but should serve as a starting point and will be extended with additional modules of state-of-the-art components in the future.

Harvesters. The simplest `harvester` module implementation provides an arbitrary, artificial signal (e.g., constant current, sine wave, etc.) with adjustable amplitude and frequency: it is intended for short-term investigations, benchmarking, or to derive principal trends. For long-term evaluation, we implement models of different energy harvesters based on real-world harvesting traces. These include a datasheet-based photovoltaic (PV) cell model [55] integrated with year-long datasets of solar irradiation for both indoor [25] and outdoor [47] use cases, as well as current traces of thermoelectric generators (TEG) in residential settings [40]. Finally, `Simba` includes a module for pre-recorded IV curves/surfaces that can be obtained using existing tools (e.g., Ekho [27] or Shepherd [21]) or common source measurement units (SMU). For the latter, we provide scripts for two different SMUs to automatically acquire and process the IV curves of energy harvesters.

Capacitors. We implement `capacitor` modules for ideal capacitors ($I_{\text{leak}} = 0$) and tantalum capacitors. In the latter, we model the leakage as a function of the capacitance, rated voltage, and instantaneous voltage according to the manufacturer's description of the AVX TAJ series [5, 59].[2]

Converters. `Simba` provides a `converter` module that resembles a converter-less architecture and modules containing generic models of a buck-, boost-, and buck-boost configuration, where input/output voltages and the corresponding conversion efficiencies can be configured arbitrarily. For our case studies and experimental evaluation, we further build models of more complex, real-world converter structures (including a load switch with hysteresis and the popular BQ25570 power management IC [31]). They are described in more detail in Sec. 4 and Sec. 5, respectively.

Loads. We develop two `load` modules that can be configured to reflect the power consumption of many (battery-free) IoT applications.

The `TaskLoad` implements a simple round-robin scheduler that executes user-specified tasks of a certain length and current consumption. To enable quick modeling of heterogeneous hardware platforms, the current consumption of each task accounts for the (average) power demand of both MCU and peripherals. However, developers may also implement more fine-grained modules to investigate other load-specific properties. To simplify the load modeling process, we provide a tool based on the popular nRF PPK2 power profiler that directly derives a load's task configuration from correlating current measurements and GPIO traces (i.e., given that the load under test indicates each task by toggling a GPIO pin). We show how to use the `TaskLoad` to represent a typical sensor node application in our first case study in Sec. 4.

The `JITLoad` models an intermittent load supporting *just-in-time (JIT) checkpointing*. This means, that the load enters a state-saving mode (i.e., it stores a checkpoint) just before power failure, i.e., once the capacitor voltage reaches a certain lower threshold. Upon restart, the checkpoint is restored and the normal operation resumes. We use this load in our second case study and discuss important design decisions targeting JIT-loads in Sec. 5.

[2] `Simba`'s interfaces also allow the integration of supercapacitor models, which are typically more complex [3, 7] but very valuable for battery-free operation. We aim to tackle the integration of such models in future work.

4 CASE STUDY 1: USING SIMBA FOR DESIGN SPACE EXPLORATION

In this section, we use `Simba` to explore and discuss the design space of an existing state-of-the-art battery-free sensor node and validate `Simba` experimentally. We show that changes in each *single device component* can have a significant impact on the *overall system's performance* and that `Simba` accurately captures these dependencies.

4.1 Used Platform and Experimental Setup

Platform. In this case study we focus on Botoks, an open-source, batteryless sensor [14] that contains a TI MSP430FR5994 MCU, a Microsemi ZL70550 ultra-low power radio, a small solar cell, and a 100 μF storage capacitor. Botoks is intended to operate intermittently, i.e., it turns on opportunistically once harvested energy is available and operates until its capacitor is depleted. We consider Botoks to be a good example of simple, tiny, low-cost, batteryless 'fire-and-forget' IoT platforms (such as beacons).

Experimental setup. We use a Botoks platform for the harvesting circuit (i.e., solar panel and converter), but replicate its load and capacitor on a bread-board (using an MSP430FR5994 Launchpad, a custom ZL70550 radio breakout board, and various capacitors). This allows us to access the MCU's GPIOs and to quickly try out different capacitor configurations. To supply Botoks, we either use a Keysight 2450 source meter (bypassing the solar panel) or create a repeatable and controlled light source by illuminating Botoks with a wirelessly-controlled LED bulb [37] placed in a closed lightbox. For each measurement, we monitor and record the MCU status (i.e., indicated by its GPIO pins), the load voltage, and the capacitor voltage for 10 seconds using a Saleae Pro 8 logic analyzer [44].

4.2 Modelling Botoks

Modelling the load. Botoks provides an example application, in which the device wakes up, performs a measurement, and immediately sends the result to an (always-on) receiver. It then remains in reception mode to deplete (i.e., 'burn') the remaining energy in the capacitor as fast as possible to eventually turn off, recharge, and wake up again. To model this behavior, we use the `TaskLoad` module and define four separate tasks (i.e., `INIT`, `SENSE`, `SEND`, `BURN`) with their corresponding lengths and current consumptions.

Modelling the converter. On Botoks, the harvester is directly connected to the storage capacitor and thus the input stage modeling is trivial (i.e., $V_H = V_{cap}, \eta_H = 1$). Between capacitor and load, Botoks uses a MIC841 comparator with hysteresis in combination with a low-drop-out regulator (LDO). This way, the MCU is turned on and off at fixed voltage thresholds and always operates

Component	Botoks hardware	Simba module impl.	Module parameters	Chosen (default) parameters
Capacitor	100 µF MLCC	`IdealCapacitor`	Capacitance Initial voltage	$100 \mu F$ $3.1\,V$
Converter	MIC841 comp.+ S1313A22 LDO	`LDO+Hysteresis`	V_L V_{Low} V_{High} $i_{\text{quiescent}}$	$2V$ $2.4\,V$ $3.1\,V$ $3\,\mu A$
Load	MSP430FR + ZL70550 radio	`TaskLoad`	List of tasks with length and current consumption (Task: t_{Task}, i_{Task})	INIT: $2.3\,ms$, $600\,\mu A$ SENSE: $0.75\,ms$, $700\,\mu A$ SEND: $7.25\,ms$, $1960\,\mu A$ BURN: ∞, $3140\,\mu A$
Harvester	KXOB25-02X8F solar panel	`Artificial`	Waveform Amplitude I_H	`constant` $600\,\mu A$

Table 1: Module configuration representing Botoks. The parameters reflect the device's non-optimal default configuration.

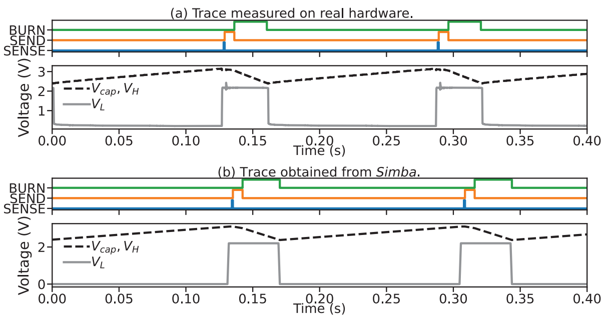

Figure 3: Voltages and load states of Botoks measured experimentally (a) and simulated by `Simba` (b). As the harvester and capacitor are directly coupled, they share the same voltage level.

at a constant voltage level. We model this converter architecture in a dedicated `LDO+Hysteresis` module considering the LDO's efficiency $\eta_L = \frac{V_L}{V_{\text{cap}}}$ [54], the hysteresis thresholds (V_{High}, V_{Low}), and a constant, configurable quiescent current.

Modelling the capacitor and harvester. We choose the `Ideal-Capacitor` module to represent Botoks' capacitor and set its capacitance accordingly. As harvesters, we either use the `Artificial-Source` module (i.e., when supplying the device from the source meter with constant current), or the `IVCurve` module (i.e., to model the solar cell's behavior). For the latter, we obtain the solar panel's IV curve for different brightness levels using the source meter.

Verification. We model Botoks according to the module selection and configurations in Tab. 1. As shown in Fig. 3, the voltage traces and state information obtained from `Simba` (a) closely resemble the actual behavior of the Botoks platform measured experimentally (b). For more details on the simulation error (such as differences in charging time), we give a quantitative comparison in the following experiments (Sec. 4.3), where we consecutively replace each device component and reconfigure each module, respectively.

4.3 Design Space Exploration and Evaluation

We now use `Simba` to discuss a number of possible design decisions for Botoks. To support our findings and evaluate `Simba`, we also provide experimental data complementing the simulation results.

4.3.1 Exploring the impact of capacitor size. Adequate sizing of the energy storage (i.e., the capacitor size) is a critical aspect when designing intermittently-powered systems. A common approach is to select the *minimum required energy storage* that allows the device to complete its largest atomic operation [11]. This minimizes the

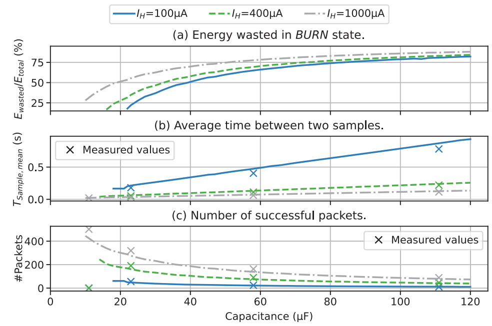

Figure 4: Impact of capacitance on Botoks' energy usage, charging time, and number of transmitted packets. Larger capacitances have a negative impact on device efficiency and latency.

device's charging time (i.e., maximizing reactivity), the capacitor's physical size, and the amount of energy that is wasted. The latter refers to the remaining energy in the capacitor that is insufficient to achieve any progress but needs to be consumed before recharging. The optimal capacitance depends on many factors, such as the load's power consumption (including start-up and actual processing), the converter structure, and the incoming harvested energy.

Using `Simba`, it is possible to quickly explore the impact of the capacitor size on Botoks' system performance. To this end, we simulate Botoks for a time period of 10 seconds using different capacitances (ranging from 10 to 120 μF) and energy harvesting conditions ($I_H = \{100, 400, 1000\}\,\mu A$) and obtain the amount of wasted energy, the average time between two successful packet transmissions ($T_{\text{Sample,mean}}$), and the number of successful packets (*#Packets*) from its logs.

Fig. 4 shows that small capacitances are highly beneficial, as they minimize the time between two successive samples, thus increasing the total number of packets. For example, at an input current of $I_H = 100\,\mu A$, decreasing the capacitance from the original $100\,\mu F$ to $22\,\mu F$ allows to transmit *3.8x more packets* within the same time window. On Botoks, large capacitances are a major source of inefficiencies, since they force the load to spend most of its active time in `BURN` state to deplete the capacitor and eventually recharge (as also shown in Fig. 3). We explore this design choice in more detail in Sec. 4.3.2.

Fig. 4 further shows that the capacitance cannot be minimized arbitrarily, as a minimum capacitance C_{min} is required to achieve any meaningful operation. Note that C_{min} depends on the harvesting current (as it affects the net current flow out of the capacitor), and thus allows designers to tune the capacitance according to the application's requirements. For example, if Botoks is intended to be used in outdoor scenarios (e.g., $I_H \approx 1000\,\mu A$), a capacitance of $10\,\mu F$ is sufficient and increases the number of successful packets by a factor of 1.5 and 5.1 compared to $22\,\mu F$ and $100\,\mu F$, respectively. This configuration, however, trades generality for system performance, as Botoks could not operate in low-light conditions.

Evaluation. Using the setup described in Sec. 4.1, we obtain the number of packets and mean time between samples for different capacitances ($C_{\text{nom}} = \{10, 22, 47, 100\}\,\mu F$) and input currents ($I_H = \{100, 400, 1000\}\,\mu A$) experimentally and plot them along with the simulation results in as markers in Fig. 4. The absolute simulation

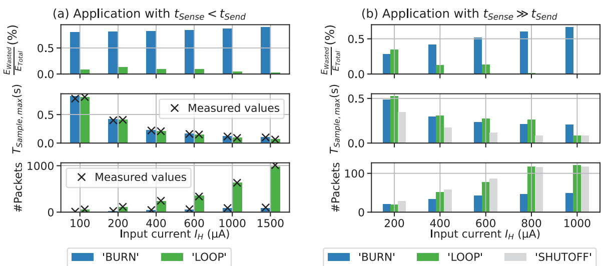

Figure 5: Impact of different load implementations for an application with a short (a) and long (b) sensing task. Compared to repeated sensing (LOOP), burning the residual energy (BURN) is only beneficial in certain application scenarios.

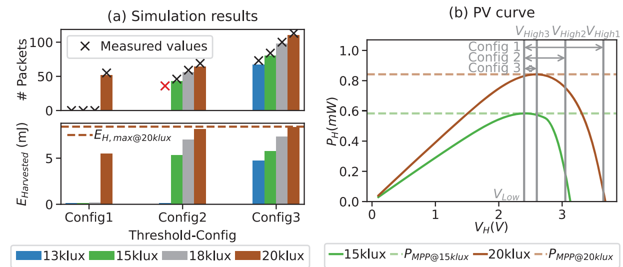

Figure 6: Impact of different capacitor and voltage threshold configurations on application and harvesting performance. Config3 (featuring a low V_{High} and large C) is superior compared to Config1 and Config2 (a). This is due to the solar panel's PV characteristics, delivering more power in certain voltage ranges (b).

error we observe is on average 5.2% and 5% for $T_{\text{Sample,mean}}$ and $\#Packets$ and never exceeds 10.1% and 10%, respectively.

Takeaway. The employed capacitance significantly affects the device's efficiency and throughput and should be selected as small as possible according to the expected harvesting current. Note that Simba is able to show this effect in less than a minute with sufficient accuracy, while the manual measurement of even a small amount of configurations is a labor-intensive task.

4.3.2 Exploring different load implementations. In intermittent systems with scarce incoming energy and small storage buffers, devices are often able to perform only a single task/useful operation per power cycle. Any successive task that runs afterward might not be able to complete such that any progress to this point is lost and the time between two successful tasks is unnecessarily prolonged [1]. For sensing applications (and event detection), this is especially problematic, as the latency of sensor data can affect the measurement quality significantly and might be more important than the actual data throughput. To minimize latency (i.e., maximize reactivity), devices like Botoks employ a technique where the residual energy is depleted as quickly as possible, e.g., by entering a high-power consumption state, so to quickly force a recharge. As hinted in Sec. 4.3.1, we are interested in exploring the effectiveness of this approach for the given hardware configuration.

To do so, we compare the existing load implementation (BURN) with a version that tries to sense and transmit as many samples as possible. In this implementation (LOOP) the load repeats the SENSE and SEND task from Tab. 1 until the device turns off. Fig. 5 (a) shows the amount of wasted energy, the maximum time between two samples (i.e., to assess the latency), as well as the number of transmitted packets for a 10-second simulation using Botok's default configuration. It can be seen that BURN is not beneficial, as LOOP achieves 3.5 – 9.7 times more packet transmissions while offering a similar reactivity and minimizing the wasted energy. This is due to the large amount of residual energy in the (oversized) capacitor and a comparable low power consumption of the BURN state, resulting in a large amount of time/energy that has to be spent to deplete the capacitor before rebooting.

However, the BURN approach can be feasible for other application characteristics, i.e., if there is less residual energy in the storage or if the power consumption difference between active operation (i.e., sensing and sending) and BURN state is higher. We show this behavior in Fig 5 (b), where we again compare BURN and LOOP but let the device sense 100 times before transmission (i.e., $t_{SENSE} = 75\,ms$), effectively decreasing the residual energy and average power consumption in active state. In this configuration, BURN's fast capacitor depletion after a successful transmission indeed decreases latency between 3.9 and 23% for $I_H \leq 800\,\mu A$. Once the harvesting current exceeds the load's average active current, LOOP again outperforms BURN, as it does not require the load to reboot between two samples. Finally, in Fig 5 (b) we also explore the performance of a third approach (SHUTOFF), where the load can voluntarily shut itself off (i.e., disable the converter output) after completion of its tasks and only restarts, once capacitor recharges to its turn-on threshold. As shown in Fig. 5 (b), SHUTOFF yields the best performance as the capacitor does not need to be depleted fully, and thus no energy is wasted. This allows to decrease $T_{Sample,max}$ by a factor of 0.32-0.65 and 0.4-0.71 compared to LOOP and BURN, respectively.

Evaluation. We modify Botok's firmware to support the LOOP approach and evaluate the simulation results experimentally for different input currents (see markers in Fig. 5 (a)). Across all datapoints, the absolute mean and maximum simulation errors amount to 3.9% and 8.4% for the maximum sample time, as well as 5.7% and 9.1% for the number of transmitted packets. Note that SHUTOFF is not evaluated experimentally, as this approach requires substantial hardware modifications. For more details, we refer the reader to previous works [1, 11, 24] that propose concepts similar to SHUTOFF.

Takeaway. Simba allows to quickly compare different load implementations (even across different hardware capabilities). In this example, we show that the effectiveness of BURN is highly dependent on the application characteristics. This capability is especially helpful if the same hardware platform should be used in similar, yet different applications (e.g., to assess the performance for various sensors or sampling rates), but can also be used in early design stages (e.g., to assess if hardware modifications are worthwhile).

4.3.3 Exploring the effect of converter configurations. As shown in Sec. 4.3.1 and Sec. 4.3.2, the device's performance significantly depends on the amount of available (residual) energy in the capacitor. The latter is not only depending on the employed capacitance, but also on the voltage range in which the capacitor can be used (i.e., $E_{\text{Cap}} \approx \frac{1}{2}C(V_{\text{High}}^2 - V_{\text{Low}}^2)$. On Botoks, this range is determined by

	$C(\mu F)$	$V_{\text{High}}(V)$	$V_{\text{Low}}(V)$
`Config1`	22 (22*)	3.74 (3.65*)	2.4
`Config2`	47 (58*)	3.1 (3.05*)	
`Config3`	100 (110*)	2.75 (2.65*)	

*Actual capacitance/voltage level (due to component tolerances)

Table 2: Converter threshold and capacitance selection

the converter's hysteresis thresholds, which enable/disable the load supply at fixed voltage levels. The turn-on threshold V_{High} ensures that the device does not turn on too early (i.e., if not enough energy has been collected yet), while the turn-off threshold V_{Low} prevents an MCU brownout that can result in excessive power draw.

Consequently, by selecting V_{High} and V_{Low}, the same energy budget E_{Cap} can be provided using different capacitances. Intuitively, one would expect a similar performance for similar E_{Cap} configurations (given that V_{High} and V_{Low} are in a reasonable range). However, due to the *direct coupling* between the energy harvester and the capacitor, this assumption does not hold true, and the converter's threshold configuration significantly affects the device behavior.

We demonstrate these effects by selecting three converter and capacitor configurations that yield the same E_{Cap}[3], as summarized in Tab. 2, and simulate Botoks at different illuminance levels that roughly represent sunny outdoor field conditions [39]. The results, shown in Fig. 6 (a), reveal two main effects w.r.t. to the converter thresholds. First, if a small capacitor in combination with a high turn-on threshold is chosen (`Config1`), the device might not be able to turn on at all in low light conditions, as the solar panel can only supply a limited harvesting voltage V_H, which might lie well below its specified open circuit voltage and may never exceed V_{High}. Second, the converter's voltage thresholds directly control the operating point of the solar panel and thus *heavily affect the harvesting efficiency*. In comparison to the energy harvested at its optimal operating point at 20 klux (i.e., if $V_H = V_{\text{Solar,MPP}}$), the panel can extract only 65% in `Config1`, but more than 97% in `Config2` and `Config3`, respectively. The solar panel's PV curve in Fig. 6 (b) confirms this observation, as `Config2` and `Config3` allow the panel to operate in ranges close to P_{MPP}. It is worth noting that this applies also to other illuminance levels. More specifically, in `Config2` and in `Config3`, between 98.3-99.7% and 91.5-97% of $E_{\text{H,MPP}}$ are harvested. This highlights that well-configured directly coupled systems can be extremely efficient while minimizing complexity, size, and costs.

Evaluation. We evaluate the configurations from Tab. 2 experimentally by adjusting the resistors on Botoks' comparator and retrieving the number of transmitted packets as well as the average sample time $T_{\text{Sample,mean}}$. Fig. 6 (a) shows a measurement outlier at 13 klux with `Config2` (red) that is caused by our measurement setup. At this particular setting, the turn-on threshold V_{High2} is very close to the recorded solar panel's maximum voltage V_H. A very small change in the device's position causes the panel voltage to increase slightly and allows Botoks to turn on. This observation emphasizes that aggressive tuning toward optimal parameters can be problematic for battery-free devices, as the performance degradation is typically not gradual but their operation fails completely (cmp. capacitor selection in Sec. 4.3.1). Excluding this outlier, we obtain an absolute

[3]Note that due to hardware imperfections (see Tab. 2), the actual energy budget differs slightly for each configuration. Nevertheless, the described effects are still valid.

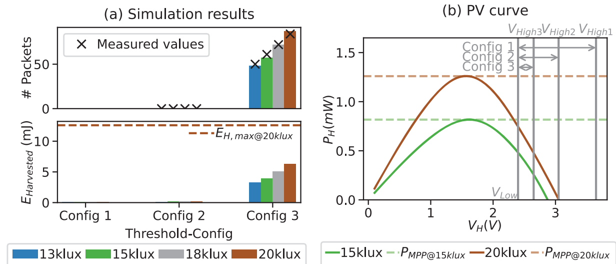

Figure 7: Impact of different capacitor and voltage threshold configurations on application and harvesting performance. Since the solar panel is operated outside of its optimal operating point (b), Botoks performs poorly (a).

simulation error of 4.2% (mean) and 7.5% (max) for $T_{\text{Sample,mean}}$, and 3.8% (mean) and 8.2% (max) for $\#Packets$, respectively.

Takeaway. Due to the non-linear PV characteristics of solar panels, the harvesting efficiency is strongly dependent on the converter's voltage thresholds, leading to performance differences despite the same energy budget. As a result, and in contrast to Sec. 4.3.1, we observe that larger capacitances (allowing the selection of a small V_{High}) are beneficial for the device's performance. This example emphasizes that the design choices for battery-free systems are non-trivial and that `Simba` can help to shed light on these dependencies.

4.3.4 Exploring different energy harvesters. Although well-configured directly coupled systems can be very efficient, they limit designers to solar panels that exactly match the load specification. We show this limitation in Fig. 7 (a), where we repeat the previous simulation, but supply Botoks with a Panasonic AM1417 solar panel (and supply `Simba` with the corresponding IV curves). This panel provides a lower open-circuit voltage and thus only `Config3` is feasible. Furthermore, considering the PV-curve in Fig. 7 (b), the efficiency constraints of directly-coupled systems are highlighted: although the AM1417 provides almost 50% more output power, only about 50% of it are actually harvested and thus the performance of Botoks is degraded. This is because the solar panel cannot be operated close to its maximum power point ($V_{\text{MPP,AM1417}} \approx 1.5\,V$), which lies outside of the load's operating range.

Evaluation. Again, we evaluate the simulation results in Fig.7 (a) experimentally and observe errors of 4.0 and 1.2% (mean) and 6.1% and 6.6% (max) for $T_{\text{Sample,mean}}$ and $\#Packets$, respectively.

Takeaway. `Simba` can accurately integrate harvesters with non-linear IV curves (e.g., solar panels) into the simulation, and is thus able to clearly show the shortcomings of a direct coupling between harvester and converter. If certain solar panels are a requirement, designers have to resort to boost converters that allow the operation of load and harvester at different voltage levels. We give an example of such an architecture in Sec. 5.

5 CASE STUDY 2: USING SIMBA TO FACILITATE DESIGN CHOICES

In the previous case study, we have used `Simba` to dive through the parameter space of a simple, directly coupled sensor node, exploring the effects of each device component. We next show how

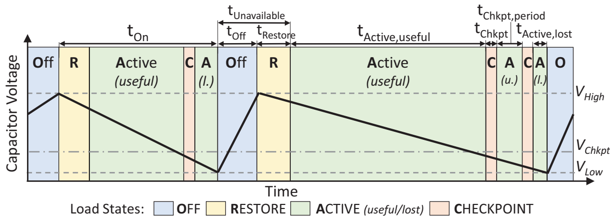

Figure 8: JIT-checkpointing mechanism of the battery-free Game Boy. Once the voltage reaches V_{Chkpt}, a checkpoint is saved periodically until shut-off. After recharging, at V_{High}, the checkpoint is restored and the device resumes its application.

Component	Game Boy Hardware	`Simba` module impl.	Module parameters	Chosen (default) parameters
Capacitor	3300 µF Alum. Elec.	`IdealCapacitor`	Capacitance Initial voltage	3300 μF 3.3 V
Converter	TI BQ25570	`BQ25570`	V_{Low} V_{High} V_L f_{MPP}	3.3 V 3.97 V 3 V 0.8
Load	Apollo 3 + LCD display + Fujitsu FRAM	`JITLoad`	$(t_{\text{Restore}}, i_{\text{Restore}})$ $(t_{\text{Chkpt}}, i_{\text{Chkpt}})$ V_{Chkpt} $t_{\text{Chkpt-Period}}$ i_{Active}	(255 ms, 2570 μA) (33 ms, 2780 μA) 3.4 V 270 ms 3180 μA
Harvester	8x Panasonic AM1417 solar panel	`IVCurve`	Solar panel Illuminance	AM1417x8 13 $klux$
		`SolarPanel` (for long-term eval.)	$V_{\text{open-circuit}}$ $I_{\text{short-circuit}}$ V_{MPP} I_{MPP}	3.56 V 14.8 mA 2.6 V 12.1 mA

Table 3: Module configuration of the battery-free Game Boy.

`Simba` can help to select appropriate parameters for a specific device and application. In particular, we use the trade-off exploration tool to optimize the checkpointing mechanism of the battery-free Game Boy [13] and perform a long-term simulation based on real harvesting traces to evaluate its performance.

5.1 Used Platform and Experimental Setup

Platform. The battery-free handheld concolse [13] is a power failure resilient Game Boy emulator powered solely by energy harvested from solar panels and button presses. It embeds a powerful Ambiq Apollo3 ARM Cortex-M4 MCU, an ultra-low power LCD display, and off-the-shelf energy harvesting switches that generate small amounts of energy while playing. The main power sources are solar panels, which are managed by a TI BQ25570 harvester/power management chip [31]. To ensure forward progress through power failures, the system state is (re)stored to and from external non-volatile memory (FRAM).

Experimental setup. For experimental evaluation we use the open-source battery-free Game Boy hardware running Tetris and only modify its storage capacitor and converter threshold settings. Similar to Sec. 4, we power it using artificial light in a closed box and collect voltage and GPIO traces using a logic analyzer.

5.2 Modelling the Battery-Free Game Boy

Modelling the load. The battery-free Game console runs a Game Boy emulator software, which we consider to have a constant current consumption (I_{Active}). To operate correctly despite power failures, the Game Boy employs a JIT-checkpointing mechanism as sketched in Fig. 8. This means that once the capacitor voltage reaches a certain lower threshold V_{Chkpt} (i.e., shortly before power is lost), a `CHECKPOINT` operation is triggered periodically to copy the application state to non-volatile memory. After a power failure and at device restart, a `RESTORE` operation retrieves the application state such that the device can resume correctly. Note that any computation after the last checkpoint is lost. We model this behavior using the `JITLoad` module and specify the voltage threshold, checkpointing period, computing power consumption, as well as the length and power consumption of the `CHECKPOINT` and `RESTORE` operation according to experimentally-obtained current consumption traces.

Modelling the converter. The employed TI BQ25570 is a state-of-the-art energy-harvesting chip that offers both a boost- and buck converter as well as voltage management features. The *boost converter* allows charging the capacitor even if $V_H < V_{Cap}$ and is further used to operate the harvester at a certain voltage level maximizing the harvesting efficiency. The operating voltage (V_H) is derived from the harvester's open-circuit voltage and a constant, configurable factor (f_{MPP}). To model the boost converter's efficiency, we create a look-up table based on the experimental data from [42] and set it according to the harvesting current and voltage. The *buck converter* supplies the load with a constant, configurable voltage (V_L). Again, we use a look-up table to account for the converter's efficiencies.Since only limited data is available, we obtain the efficiencies experimentally as proposed in [30]. Finally, the BQ25570 integrates *voltage comparators* to enable/disable the buck-converter at certain voltage thresholds. On the Game Boy, the lower threshold is fixed to $V_{\text{Low}} = 3.3\,V$, while the upper turn-on threshold is configurable using on-board jumpers ($V_{\text{High}} = \{3.63, 3.97, 4.3, 4.61, 4.87\}\,V$). We model the chip's behavior and embed the look-up tables in a dedicated `BQ25570` converter module.

Modelling the capacitor and harvester. We use the `IdealCapacitor` module with its capacitance configured accordingly and the `IVCurve` module supplied with IV curves from the Game Boy's solar panels. Although `Simba` would also support the modelling of hybrid energy harvesters, we do not include the Game Boy's button harvesting functionality, as the provided energy is comparably small and difficult to reproduce reliably in the experiments.

Verification and evaluation. We summarize the Game Boy's module configuration in Tab. 3 and verify that the simulation traces closely match the real-world behavior in Fig. 9. To quantify the simulation accuracy, we obtain the average charging time (t_{Off}) and operating time (t_{On}) both experimentally and in simulation using five illuminance levels (10, 13, 15, 18, and 20 klux), three capacitors ($C = 1000, 3300, 6800\,\mu F$), and three converter threshold settings ($V_{\text{High}} = 3.63, 3.97, 4.3\,V$). Across all measurement points, the absolute simulation error amounts on average to 4.4% and 6.5% for t_{Off} and t_{On}, and never exceeds 16.4% and 13.5%, respectively.

5.3 Using `Simba` to Simplify Design Decisions

The Game Boy employs a JIT-checkpointing approach (see Fig. 8) and thus its performance depends, among others, on the selected voltage thresholds and the employed capacitance. In the following,

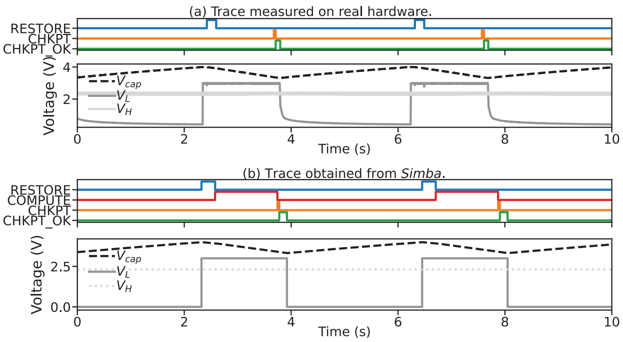

Figure 9: Voltages and load states of the battery-free Game Boy measured experimentally (a) and simulated by `Simba` (b). Due to the boost converter, the capacitor can charge despite a low harvesting voltage (i.e., even if $V_H < V_{\text{cap}}$).

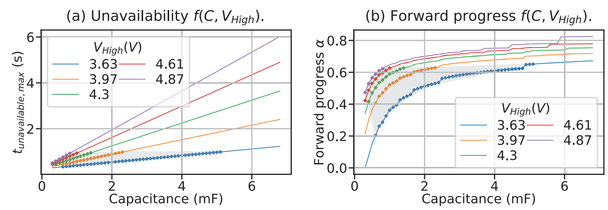

Figure 10: Trade-off between unavailability and forward progress for different configurations of capacitance C and voltage threshold V_{High} of the Game Boy. Feasible parameter pairs (i.e., where $t_{\text{Unavailable}} < 1\,s$) are highlighted in light grey.

we discuss important performance metrics in the context of JIT-checkpointing and show how to tackle their trade-offs using `Simba`.

Performance metrics and trade-offs in JIT-checkpointing. While checkpointing is necessary to ensure correct operation in the presence of power failures, it introduces state-saving and restoring overheads that can impede *forward progress*. Forward progress describes the relative time [59] or energy [43] that a device spends on useful work. Considering Fig. 8 and the definition in [59], we define the *forward progress* α for the Game Boy as

$$\alpha = \frac{\sum t_{\text{Active,useful}}}{\sum t_{\text{Active (useful+lost)}} + \sum t_{\text{Off}} + \sum t_{\text{Restore}} + \sum t_{\text{Chkpt}}} \tag{5}$$

Forward progress is, among others, affected by the amount of energy storage that the device provides. If the energy storage is very small, t_{On} is short, and thus the majority of time/energy is spent on restoring and state-saving, minimizing the time of actual useful work. Large energy storage, on the other hand, increases t_{On} and thus the checkpointing overheads are comparably small. The latter, however, comes at the cost of increased charging times, which makes the system less reactive. This can have a negative impact on data quality (i.e., for sensing applications) or user experience (i.e., for the Game Boy). On the Game Boy, the system remains unresponsive (i.e., the user cannot play) when the device is in `OFF` and `RESTORE` state, respectively. Consequently, following Fig. 8, we define the *unavailability* for the Game Boy as

$$t_{\text{Unavailable}} = \max_{i \in I}(t_{\text{Off,i}} + t_{\text{Restore,i}}) \tag{6}$$

where I corresponds to all power cycles during the operation period.

Maximizing forward progress by tuning V_{Chkpt}. To maximize forward progress, the selection of the checkpoint threshold V_{Chkpt} (i.e., when the device starts to save the application state) is crucial: If V_{Chkpt} is too low, the energy left in the capacitor might be insufficient to complete the checkpoint and no forward progress can be made. If V_{Chkpt} is too high, superfluous checkpoint operation(s) and an increasing amount of lost computation ($t_{\text{Active,lost}}$) limit the forward progress. The optimal checkpoint threshold $V_{\text{Chkpt,opt}}$ is thus the *minimum voltage* at which a checkpoint will succeed without any energy income.

In principle, $V_{\text{Chkpt,opt}}$ can be computed analytically but requires knowledge about the employed capacitor, the energy demand of the load's checkpointing process, as well as the losses in the converter and capacitor. `Simba` allows obtaining $V_{\text{Chkpt,opt}}$ in a device-agnostic and simple way. To do so, we supply the trade-off exploration tool with the base configuration (Tab. 3) without an energy harvester (i.e., $I_H = 0$) and let it iterate over V_{Chkpt} from V_{High} to V_{Low} in steps of $0.01\,V$ while collecting the $num_{\text{Chkpt,success}}$ information from the load. $V_{\text{Chkpt,opt}}$ is then given by the minimum V_{Chkpt} for which $num_{\text{Chkpt,success}} = 1$.

Following this approach and using the Game Boy's default configuration (from Tab. 3), we obtain $V_{\text{Chkpt,opt}} = 3.35\,V$. In comparison to the default value ($V_{\text{Chkpt,default}} = 3.4\,V$), when simulating at illuminance levels of 10, 13, 15, 18, and 20 klux, the forward progress α improves between 9.4% and 10.1%.

Trading forward progress and unavailability by selecting V_{High} and C. As discussed previously, forward progress and unavailability are strongly affected by the amount of energy storage. On the Game Boy, the energy storage depends on the capacitance C, the voltage thresholds, and the converter efficiency (i.e., $E \approx \frac{1}{2}C(V_{\text{High}}^2 - V_{\text{Low}}^2)$). Considering the fixed turn-off threshold (V_{Low}=3 V), forward progress and unavailability are thus a function of V_{High} and C. How to properly trade-off forward progress and unavailability is in general highly application-specific, but we give an example for the Game Boy using the following constraints (`C1`, `C2`). In an outdoor setting with moderate sun exposure (delivering a harvesting current of 3.4 mA [13]), the user should not experience an outage of more than one second (i.e., $t_{\text{Unavailable}} \leq 1\,s$) (`C1`). At the same time, the forward progress should be maximized (`C2`).

To find an appropriate parameter pair, we evaluate forward progress and unavailability for a large number of V_{High} and C using `Simba`. Specifically, we use the Game Boy's default configuration and let `Simba` iterate over capacitances from 1 to 6.8 mF (in steps of 0.1 mF) and the five possible V_{High} settings. Fig. 10 shows the discussed trade-off: with increasing V_{High} and C (corresponding to larger available energy storage), the forward progress rises but the unavailability is affected negatively. From the obtained results, we retrieve all *feasible parameter pairs*, as highlighted in Fig. 10 (a) and (b) in grey. Note that for each V_{High} there exists a $C_{\min}$ that ensures a successful checkpoint even in the absence of incoming energy and a $C_{\max}$, such that $t_{\text{Unavailable}} \leq 1s$ (`C1`) is still satisfied. Out of the feasible candidates, we select the parameter pair that yields the maximum α (`C2`) and obtain $\{C_{\text{Opt}} = 5.1\,mF, V_{\text{High,Opt}} = 3.63\,V\}$ as the optimal parameters for our system.

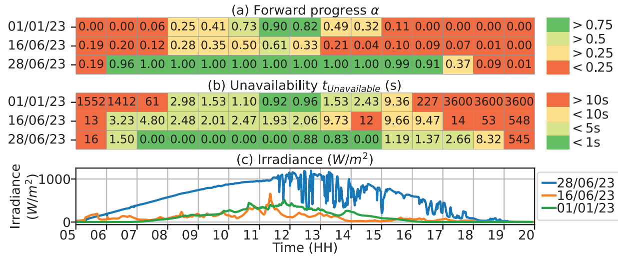

Figure 11: Hourly forward progress (a) and unavailability (b) based on real-world energy harvesting traces (c) obtained by Simba using C=5.1 mF and $V_{\textbf{High}}$=3.63 V. Adverse weather conditions degrade device performance significantly.

Evaluation with real-world harvesting traces. We now use Simba to evaluate this configuration in a realistic deployment scenario, i.e., using real-world energy harvesting traces. To this end, we make use of the SolarPanel harvester module that incorporates (i) a PV cell model [55] based on parameters that are commonly available in datasheets and (ii) real-world solar irradiance traces such as the NREL [47] or ENHANTs [25] data sets. We configure the module to use the Game Boy's PV cell parameters (see Tab. 3) as well as outdoor solar traces from Colorado [47], and let Simba simulate the Game Boy's operation for two entire months (January and June 2023). In Fig. 11, we show the hourly forward progress (a) and unavailability (b) for three days to demonstrate how solar irradiance affects device performance. Designers can use this feature to estimate whether the system meets the application requirements for a given deployment location (i.e., if irradiance traces are available) and to explore seasonal effects. For example, the hourly mean forward progress (between 5:00 and 20:00) is 0.61 in June, but only 0.34 in January, due to the limited amount of sun hours and lower irradiance in winter.

6 DISCUSSION AND FUTURE WORK

In this work, we have introduced Simba to simplify the design and evaluation of battery-free devices. Simba introduces an extendable foundation for the community to expand upon. In this section, we discuss the current limitations and potential future work.

Usability. Although the architecture of Simba is simple to understand, the initial configuration of platforms as well as the implementation and validation of new modules requires domain knowledge and careful modeling to provide reasonable results. To support users to get started, we complement Simba's source code with examples, tools to create new modules automatically, and a dedicated documentation page [8]. Additionally, together with the community we aim to collect feedback and improve Simba 's usability.

Improvement and extension of component modules. In the future, we plan to improve the existing module implementations and aim to add additional modules of state-of-the-art components. As Simba is intended to explore the interactions between device components in a quick and easy way, the provided modules so far are rather generic. For example, the load modules include the average power consumption of both MCU and peripherals, hence trading simplicity for accuracy. To investigate MCU-related effects (such as checkpointing approaches or energy-aware scheduling), Simba's interfaces and modularity would allow the integration of more fine-grained load modules, tailored to specific needs and hardware, where the load behavior is not modeled based on tasks, but on instruction level. This way, users could also integrate models that consider the voltage dependency of an MCU's power consumption [2] or effects of different memory accesses [34]. Furthermore, it is of interest to integrate code congruity, as used in existing simulators [17, 51], enabling direct debugging and reuse of the load's software on the target platform. Furthermore, the current BQ25570 module has only limited accuracy w.r.t. its cold-start mode. However, the cold-start (i.e., the initial charging to the converter's operating voltage) can be extremely inefficient and lead to long (non-negligible) activation times [56]. Thus, modeling this behavior and comparing it to other charging chips is an interesting avenue for future work. Finally, we also encourage the community to contribute new module implementations to the open-source framework, so to eventually create a database that facilitates early design-space exploration without access to real hardware.

Exploring energy-neutral operation. While both case studies in this paper both discuss intermittently powered battery-free devices, Simba can also be used to explore (and evaluate) energy-neutral operation due to its integration of energy harvesting datasets and the ability of long-term simulation.

Automating parameter and component selection. In Sec. 5 we show a simple, 'manual' parameter selection mechanism, which we aim to expand to a powerful optimization framework, including a user-friendly GUI. It should be able to consider different design parameters (e.g., harvesting potential, checkpointing frequency) and objectives (e.g., costs, physical size) and to optimize according to dedicated cost functions. For example, the user could specify a certain load and energy harvesting profile for which the optimization framework then provides the optimal capacitance and threshold settings. Note that such optimizations include multi-objective decision-making and are beyond the scope of this paper. Furthermore, in our case studies, we mainly focus on optimizing the parameters of certain (fixed) components, while such an optimization framework would also allow the selection of optimal components for a given problem (e.g., by choosing from a pool of different converters or capacitor types).

Federated energy storage and networking. Previous work has proposed the use of federated energy storage (i.e., multiple capacitors matching the load's energy demand) to increase the availability of battery-free devices [11, 28]. Due to its system model centering around a single storage capacitor, Simba is not compatible with such approaches. Furthermore, Simba's design is based on the architecture of a *single* battery-free sensor node and does not consider communication or networking functionalities. As we are not aware of any simulation environment targeting federated storage architectures or networking using battery-free devices, we deem this an interesting direction for future work.

7 RELATED WORK

In the recent years, a large number of battery-free devices have been developed [1, 6, 11, 13, 16, 18, 22, 33]. While the majority of them target specific applications (e.g., certain sensing or monitoring tasks [1, 23, 41, 48]), there is also an increasing trend towards battery-free general purpose and prototyping platforms [6, 10, 11, 29, 38]. To simplify and accelerate the development and to enable the evaluation of such devices, researchers have contributed different types of works, which can be broadly categorized into (i) models, (ii) development tools and testbeds, and (iii) simulators.

Models of battery-free systems and their components. Lots of efforts have been put into modelling single components of a battery-free device, including different energy harvesters (e.g., piezoelectric harvesters [36], PV panels [55], radio-frequency harvesters [53]) and converter architectures [24, 42]. Furthermore, there exist models for supercapacitors that consider the effect of intermittency (i.e., rapid charging and discharging) [3] or leakage currents [7, 60] on its state of charge. Others have focused on modeling and evaluating the impact of load-specific behavior, such as the checkpointing process [4, 43] or the MCU's power consumption in the presence of a varying operating voltage [2]. There exists also a limited number of works that model the interactions between the different components. DEBS [24] shows that the adjustment of the turn-on threshold based on the load's operations can improve the system's efficiency. In [59], the impact of capacitor size on checkpointing efficiency is explored. Note that these models are specific to certain device architectures, while `Simba` allows to unify and explore existing component models in a unified framework.

Development tools and testbeds. Besides models, researchers have introduced important tools that allow designers to debug and evaluate battery-free systems based on real hardware. DIPS [15] is a debugger specifically designed for intermittent systems, providing energy emulation and verification of memory states between power failures. Ekho [27] is an energy recorder/emulator device used to replicate energy harvesting environments in the laboratory setting. Shepherd [21] extends this idea and embeds energy recording/emulation capabilities in a testbed, which allows to account for the differences in energy harvesting performance of spatially-separated sensor nodes. While these tools are extremely valuable, they require access to real hardware and are limited to certain architectures and components. We thus consider `Simba` a complementary work that can be used in early development stages to rapidly explore the design space of battery-free systems.

Simulators for battery-free systems. Most existing sensor node simulators assume a constant power supply [17, 19] or do not account for the harvester's IV dependency [26, 32], and are thus infeasible to capture the energy-driven interactions between components and to examine the overall performance of battery-free systems. In contrast, Siren [20] extends the MSPSim simulator [17] with a capacitor model and the capability to replay IV surfaces recorded by Ekho. Siren is MSP430-specific and targets only converter-less systems. Fused [51] implements a closed-loop, MCU-centric simulation of battery-free devices that are either powered by a voltage-limited current source or a solar panel [57]. As it is implemented in SystemC, it offers high accuracy but the addition of new components is complex. Fused thus provides only a small number of hardware options and has restrictions w.r.t. long-term evaluations.

8 CONCLUSION

In this paper, we present `Simba`, a framework that allows to unify various component models within a single simulation core to investigate and simplify the design of battery-free systems. We use `Simba` to explore design trade-offs of existing state-of-the-art devices and provide experimental data to support our findings. We show that the dependencies between device components must not be neglected in system design, as small changes in single components can have a large impact on overall device performance. We demonstrate how `Simba` can support developers by providing means to trade off application-specific design parameters and to perform long-term simulations using real-world energy harvesting traces.

REFERENCES

[1] M. Afanasov et al. 2020. Battery-less Zero-maintenance Embedded Sensing at the Mithræum of Circus Maximus. *Proc. of the Conference on Embedded Networked Sensor Systems (SenSys '20).*

[2] S. Ahmed et al. 2020. Demystifying Energy Consumption Dynamics in Transiently Powered Computers. *Trans. on Embedded Computing Systems* (2020).

[3] J. Ahn et al. 2022. State-of-Charge Estimation of Supercapacitors in Transiently-Powered Sensor Nodes. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2022).

[4] Alberto Arreola et al. 2015. Approaches to Transient Computing for Energy Harvesting Systems: A Quantitative Evaluation. *Proc. of the International Workshop on Energy Harvesting and Energy Neutral Sensing Systems (ENSsys '15).*

[5] AVX. 2016. Low Leakage Current Aspect of Des. with Tantalum and Niobium Oxide Cap. [Online] https://tinyurl.com/4au4up35 – Last access: 2023-06-27.

[6] Abu Bakar et al. 2022. Protean: An Energy-Efficient and Heterogeneous Platform for Adaptive and Hardware-Accelerated Battery-Free Computing. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems (SenSys '22).*

[7] Hannah Brunner et al. 2022. Leakage-Aware Lifetime Estimation of Battery-Free Sensor Nodes Powered by Supercapacitors. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems (ENSsys '22).*

[8] Hannah Brunner et al. 2024. Simba simulation framework: documentation page. https://lens-tugraz.github.io/simba/.

[9] Ruizhi Chai and Ying Zhang. 2015. A Practical Supercapacitor Model for Power Management in Wireless Sensor Nodes. *Transactions on Power Electronics* (2015).

[10] Nessie Circuits. 2023. Riotee: An open-source platform for the battery-free IoT. [Online] https://tinyurl.com/38m5nhw6 – Last access: 2023-10-02.

[11] Alexei Colin et al. 2018. A Reconfigurable Energy Storage Architecture for Energy-Harvesting Devices. In *Proc. of the Int. Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '18).*

[12] Alexei Colin, Graham Harvey, Alanson P. Sample, and Brandon Lucia. 2017. An Energy-Aware Debugger for Intermittently Powered Systems. *IEEE Micro* (2017).

[13] Jasper de Winkel et al. 2020. Battery-Free Game Boy. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT '20).*

[14] Jasper de Winkel et al. 2020. Reliable timekeeping for intermittent computing. *Proc. of the 25th Int. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS '20).*

[15] Jasper de Winkel et al. 2022. DIPS: Debug Intermittently-Powered Systems Like Any Embedded System. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems (SenSys '22).*

[16] Jasper de Winkel et al. 2022. Intermittently-Powered Bluetooth that Works. *The 20th Int. Conf. on Mobile Systems, Applications and Services (MobiSys '22).*

[17] Joakim Eriksson et al. 2007. Poster Abstract: MSPsim–an Extensible Simulator for MSP430-equipped Sensor Boards. (2007).

[18] F. Fraternali et al. 2018. Pible: Battery-Free Mote for Perpetual Indoor BLE Applications. In *Proc. of the 5th Conf. on Systems for Built Env. (BuildSys '18).*

[19] Fredrik Österlind. 2006. A Sensor Network Simulator for the Contiki OS. https://api.semanticscholar.org/CorpusID:33079530

[20] Matthew Furlong et al. 2016. Realistic Simulation for Tiny Batteryless Sensors. In *Proc. Workshop on Energy Harvesting and Energy-Neutral Sens. Sys. (ENSsys'16).*

[21] Kai Geissdoerfer et al. 2019. Shepherd: A portable testbed for the batteryless IoT. *Proc. of the 17th Con. on Emb. Networked Sensor Systems (Sensys '19).*

[22] Kai Geissdoerfer and Marco Zimmerling. 2021. Bootstrapping Battery-free Wireless Networks: Efficient Neighbor Discovery and Synchronization in the Face of Intermittency. In *USENIX Symp. on Net. Sys. Design and Impl. (NSDI '21).*

[23] Marco Giordano et al. 2020. A Battery-Free Long-Range Wireless Smart Camera for Face Detection. *Proc. Int. Workshop on Energy Harvesting and Energy-Neutral Sensing Sys. (ENSsys '20)* (2020).

[24] Andres Gomez et al. 2016. Dynamic energy burst scaling for transiently powered systems. In *Design, Auto. & Test in Europe Conf. & Exhibition (DATE '16).*

[25] M. Gorlatova et al. 2011. Networking Low-Power Energy Harvesting Devices: Measurements and Algorithms. In *Proc. Conf. on Comp. Comm. (INFOCOM '11)*.

[26] Yizi Gu et al. 2016. NVPsim: A simulator for architecture explorations of nonvolatile processors. In *Asia and South Pacific Design Auto. Conf. (ASP-DAC '16)*.

[27] Josiah Hester et al. 2014. Ekho: Realistic and Repeatable Experimentation for Tiny Energy-Harvesting Sensors. In *Conf. on Emb. Net. Sensor Sys. (SenSys '14)*.

[28] Josiah Hester et al. 2015. Tragedy of the Coulombs: Federating Energy Storage for Tiny, Intermittently-Powered Sensors. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys '15)*.

[29] Josiah Hester and Jacob Sorber. 2017. Flicker: Rapid Prototyping for the Batteryless Internet-of-Things. In *Proc. Conf. on Emb. Net. Sensor Sys. (SenSys '17)*.

[30] Texas Instruments. 2014. *User's Guide for bq25570 Battery Charger Evaluation Module for Energy Harvesting*.

[31] Texas Instruments. 2019. BQ25570: Ultra Low power Harvester power Management IC with boost charger, and Nanopower Buck Converter.

[32] Neal Jackson et al. 2019. Capacity over Capacitance for Reliable Energy Harvesting Sensors. In *Proc. Conf. on Inf. Proc. in Sensor Networks (IPSN '19)*.

[33] Kang Eun Jeon et al. 2019. luXbeacon: A Batteryless Beacon for Green IoT: Design, Modeling, and Field Tests. *IEEE Internet of Things Journal* (2019).

[34] Dong Ji et al. 2022. Memory Layout Optimization for Task-Based Intermittent Computing Systems. In *Int. Conf. on Intelligent Technology and Embedded Systems (ICITES '22)*.

[35] Tom J. Kamierski and Steve Beeby. 2010. *Energy Harvesting Systems: Principles, Modeling and Applications.* Springer.

[36] Shashi Kiran et al. 2015. Modeling, simulation and analysis of piezoelectric energy harvester for wireless sensors. In *Int. Conf. on Control, Electronics, Renewable Energy and Communications (ICCEREC '15)*.

[37] Koninklijke Philips N.V. 2021. Hue Smart Light Bulb White Ambiance E27. [Online] https://tinyurl.com/b4426jum– Last access: 2021-08-19.

[38] Vito Kortbeek et al. 2020. BFree: Enabling Battery-Free Sensor Prototyping with Python. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 4 (2020).

[39] Carla Lança et al. 2019. The Effects of Different Outdoor Environments, Sunglasses and Hats on Light Levels: Implications for Myopia Prevention. *Translational Vision Science & Technology* (2019).

[40] Victor Ariel Leal Sobral et al. 2021. Thermal Energy Harvesting Profiles in Residential Settings. In *Proc. Conf. on Emb. Netw. Sensor Sys. (SenSys '21)*.

[41] Gael Loubet, Alexandru Takacs, and Daniela Dragomirescu. 2019. Implementation of a Battery-Free Wireless Sensor for Cyber-Physical Systems Dedicated to Structural Health Monitoring Applications. *IEEE Access* (2019).

[42] Mojtaba Masoudinejad et al. 2018. Average Modelling of State-of-the-Art Ultralow Power Energy Harvesting Converter IC. In *Int. Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM'18)*.

[43] Joshua San Miguel et al. 2018. The EH Model: Early design space exploration of intermittent processor architectures. *Proceedings of the Annual International Symposium on Microarchitecture (MICRO '18)*.

[44] Saleae Inc. 2021. Logic Pro 8 USB Logic Analyzer. [Online] https://tinyurl.com/yc29rbpz – Last access: 2020-06-12.

[45] Alanson P. Sample et al. 2008. Design of an RFID-Based Battery-Free Programmable Sensing Platform. *IEEE Trans. on Instrumentation and Meas.* (2008).

[46] Nurani Saoda and Bradford Campbell. 2019. No batteries needed: Providing physical context with Energy-harvesting beacons. *Proceedings of the 7th International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems (Enssys' 19).*

[47] M. Sengupta et al. 2018. The National Solar Radiation Data Base (NSRDB). *Renewable and Sustainable Energy Reviews* (2018).

[48] U. Senkans et al. 2017. Applications of Energy-Driven Computing: A Transiently-Powered Wireless Cycle Computer. *Proceedings of the 5th International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems (ENSsys '17)*.

[49] Ryo Shigeta et al. 2013. Ambient RF Energy Harvesting Sensor Device With Capacitor-Leakage-Aware Duty Cycle Control. *IEEE Sensors Journal* (2013).

[50] Sivert T. Sliper et al. 2019. Energy-driven computing. *Philosophical Transactions of the Royal Society A* (2019).

[51] Sivert T. Sliper et al. 2020. Fused: Closed-Loop Performance and Energy Simulation of Embedded Systems. In *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS '20)*.

[52] Statista. 2022. Number of Internet of Things connected devices worldwide from 2019 to 2021. [Online] https://tinyurl.com/msza2dtr – Last access: 2023-05-16.

[53] Tapashi Thakuria et al. 2017. Modelling and simulation of low cost RF energy harvesting system. In *Conf. Innovations in Elec., Signal Proc. & Comm. (IESC '17)*.

[54] Toshiba. 2022. Basics of Low-Dropout (LDO) Regulators.

[55] Silvano Vergura. 2016. A Complete and Simplified Datasheet-Based Model of PV Cells in Variable Env. Conditions for Circuit Simulation. *Energies* (2016).

[56] Chung-Hsiang Wang et al. 2023. Enh. charge circuitry for indoor PV energy harvesting with fast activation and high efficiency. *IET Power Electronics* (2023).

[57] Samuel C.B. Wong et al. 2020. Energy-Aware HW/SW Co-Modeling of Batteryless Wireless Sensor Nodes *(ENSsys '20)*.

[58] Fan Wu et al. 2018. WE-Safe: A Self-Powered Wearable IoT Sensor Network for Safety Applications Based on LoRa. *IEEE Access* (2018).

[59] Jie Zhan et al. 2022. Exploring the Effect of Energy Storage Sizing on Intermittent Computing System Performance. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* (2022).

[60] T. Zhu et al. 2009. Leakage-Aware Energy Synchronization for Wireless Sensor Networks. *Proc. Conf. on Mobile Systems, Appl., and Services (MobiSys '09)*.