CapDYN: Adaptive Self-Scaling Energy Storage for Powering Batteryless IoT

MARIA DOGLIONI, Industrial Engineering, University of Trento, Trento, Italy
EREN YILDIZ, School of Interactive Computing, Georgia Institute of Technology, Atlanta, United States
MATTEO NARDELLO, Industrial Engineering, University of Trento, Trento, Italy
KHAKIM AKHUNOV, Department of Information Engineering and Computer Science (DISI), University of Trento, Trento, Italy
KASIM SINAN YILDIRIM, Department of Information Engineering and Computer Science (DISI), University of Trento, Trento, Italy
DAVIDE BRUNELLI, Industrial Engineering, University of Trento, Trento, Italy

Battery-free devices collect the harvested ambient energy in their energy storage capacitors. The size of the storage capacitor is one of the main factors affecting the device's active time and power failure rate. In fact, a larger capacitor ensures energy autonomy for longer operations, while a smaller capacitor charges faster and shrinks inefficient cold starts. This paper presents *CapDYN*, a new energy storage architecture that can self-adapt its capacity based on incoming ambient energy. *CapDYN* automatically reconfigures the size of its capacitor bank to both speed up charging and improve execution rate. *CapDYN* reduces the startup time by up to 98% and can schedule tasks up to 38% faster, compared to a fixed-size capacitor. *CapDYN* operates in a fully autonomous manner consuming down to 11.6 µW in its simplest implementation and replaces the power-hungry microcontroller governing the switching operation with a dedicated ultra-low-power circuit built with COTS components. Its power consumption improves the previous state of the art by 73%, all the while featuring uncompromising reactivity. *CapDYN* can instantly react to sudden power transients without incurring extra power draw by foregoing MCU-driven reconfiguration used in state-of-the-art dynamic energy storages.

CCS Concepts: • **Hardware → Reusable energy storage**; **Sensor devices and platforms**; **Renewable energy;**

This work was supported by the national research project GEMINI Green Machine Learning for the IoT), funded by the European Union under Next Generation EU, Missione 4 Componente 2, CUP 20223M4HZ4.
The testbed activities, which are distinct from those funded by GEMINI, were partially supported by the ECOSENTINEL (Ecological Sentinel) project, funded by the European Innovation Council (EIC) under grant agreement No 101186925.
No overlap or double funding occurred between the two projects.
Authors' Contact Information: Maria Doglioni, Industrial Engineering, University of Trento, Trento, Trentino-Alto Adige, Italy; e-mail: maria.doglioni@unitn.it; Eren Yildiz, School of Interactive Computing, Georgia Institute of Technology, Atlanta, Georgia, United States; e-mail: eyildiz8@gatech.edu; Matteo Nardello, Industrial Engineering, University of Trento, Trento, Trentino-Alto Adige, Italy; e-mail: matteo.nardello@unitn.it; Khakim Akhunov, Department of Information Engineering and Computer Science (DISI), University of Trento, Trento, Trentino-Alto Adige, Italy; e-mail: khakim.akhunov@unitn.it; Kasim Sinan Yildirim, Department of Information Engineering and Computer Science (DISI), University of Trento, Trento, Trentino-Alto Adige, Italy; e-mail: kasimsinan.yildirim@unitn.it; Davide Brunelli, Industrial Engineering, University of Trento, Trento, Trentino-Alto Adige, Italy; e-mail: davide.brunelli@unitn.it.

ACM 1539-9087/2025/09-ART77
https://doi.org/10.1145/3737288

Additional Key Words and Phrases: Energy harvesting, microbial fuel cells, intermittent computing, ultra-low power electronics

ACM Reference Format:
Maria Doglioni, Eren Yildiz, Matteo Nardello, Khakim Akhunov, Kasim Sinan Yildirim, and Davide Brunelli. 2025. *CapDYN*: Adaptive Self-Scaling Energy Storage for Powering Batteryless IoT. *ACM Trans. Embedd. Comput. Syst.* 24, 5, Article 77 (September 2025), 32 pages. https://doi.org/10.1145/3737288

1 Introduction

Powering embedded electronics away from mains is a critical issue for system designers: batteries might have been a convenient solution, but they offer a limited lifetime, have a high environmental impact, and potentially incur high maintenance costs. Recent technological advancements have enabled battery-free, ultra-low-power embedded systems to operate exclusively with the energy collected from their surroundings [2, 23, 29]. A typical battery-free device includes an energy harvesting circuit that harvests ambient power, e.g., from solar [1] or radio waves [20], and stores it in a capacitor, which then sources power to the device components, such as the microcontroller and sensors.

Energy harvesting (EH) sources might be transient and sporadic, which can prevent continuous EH-supplied operation. Indeed, battery-free nodes typically operate *intermittently* [10, 17], as they charge and discharge their storage capacitor in between their upper and lower operating voltage threshold. Selecting the right size for the energy storage capacitor is one of the most crucial design issues for intermittent systems [26]. A large capacitor extends the device on-time upon discharge, but it charges slower, leading to long but infrequent power failures. Conversely, a small capacitor grants a short on-time, but it charges faster, resulting in short but frequent power failures. Although a fixed capacitor can be tailored to a single task, it cannot sustain sets of tasks with different energy consumption levels without incurring in either long charging times for shorter tasks or inadequate energy levels to sustain longer tasks.

Variable capacitors have been proposed as a solution for sustaining sets of tasks with different energy draws, while also minimizing charging times. Two opposing capacitance control approaches emerge from the literature. The first approach is to leverage *autonomous* control. Gogolou et al. [13] and El Mahboubi et al. [12] autonomously combine different sized capacitors in series or parallel to offer two capacitance levels, either a small capacitance and fast charging or a large capacitance and long on-time, respectively. The second approach is to rely on *software* control of capacitor size[7]: Yang et al. proposed Morphy [30], which is the state-of-the-art energy storage management system for intermittent systems. Morphy is a software-defined configurable capacitor bank whose capacitors are connected via switches controlled by a *dedicated* microcontroller (MCU). Although software control enables the use of advanced control policies and grants a higher number of capacitance levels compared to the available autonomous solutions, it comes at the cost of an increased energy overhead. Capacitance steps are crucial for perfectly matching each task to its minimum capacity requirement, to minimize charging times. Energy efficiency is also paramount in battery-free ecosystems: the variable capacitance infrastructure must consume as little as possible to maximize the amount of harvested energy which is actually supplied to the load.

Instead of having a variable capacitor bank, other works solve the limitations of a fixed capacitor when supplying different tasks by *federating* the energy storage [3, 15, 16, 34]. This consists in dimensioning a single fixed capacitor for each peripheral, providing control as to which capacitor to charge. Although federation can be an interesting solution, it still faces the same limitations of fixed capacitors at the peripheral level, if multiple tasks use the same peripheral. Moreover, federation introduces the so-called *energy fragmentation* problem [34], as federation dilutes available energy

across capacitors, so that even if the total stored energy would be sufficient for executing a task, operative voltage is not reached at the required peripherals. Solving this requires sub-optimal charge transfers among capacitors [34]. Because of this reasons, we focus on developing a centralized variable storage solution, rather than a federated one.

Problem statement. An ideal energy storage system should perfectly match the task at hand by providing just enough capacitance to sustain it, so that charging times are minimized. The capacitance control should minimally impact the energy efficiency of the variable storage. While the software control of state-of-the-art reconfigurable storages offers greater flexibility, it reduces the energy efficiency, as active MCUs are power-hungry components. On the other hand, the autonomous control can provide capacitance scaling with less power draw, especially if integrated. Besides, software intervention ties *efficiency* and *reactivity*: the more the system needs to be reactive to changes, the more the MCU needs to be active and the less energy efficient the variable storage is. Autonomous (i.e., not MCU-driven) capacitance scaling solves the reactivity-efficiency trade-off, as reactivity does not lead to an increased power consumption. This is essential to allow intermittent systems operating on µW-level transient power inputs to collect even the shortest bursts of incoming energy.

Challenges. For the purpose of explaining the challenges of building a variable capacitance system, consider a simple illustrative design - which we term D0 - similar to the one proposed in [7] that consists of a variable number of parallel capacitors. While in [7] parallel capacitors are set by a software policy, in the D0 autonomous design we start charging a single capacitor and *progressively scale up* the bank capacitance by adding a capacitor in parallel once a threshold charging voltage is reached, in order to make room for more energy. As soon as we store sufficient energy for a certain task, we start discharging the bank of parallel capacitors. This design introduces **two** important design challenges. **I)** When scale up is triggered, adding an initially discharged capacitor in parallel to a charged capacitor gives way to *energy waste* due to dissipative voltage equalization transients and potential *power failures*, as the overall voltage might stabilize below the minimum supply level of the system. This can lead to undefined behavior of the load or even trigger a Brown Out Reset (BOR). **II)** Once the bank supplies the task and discharges to the minimum load supply voltage level, the energy below the voltage threshold can no longer be used to provide an operational voltage. This "leftover" energy cannot be consumed by the load, yet it must be harvested upon cold start to begin supplying the load, thus reducing the energy efficiency of the storage. With the aforementioned D0 design, we cannot tap into the leftover energy, which cannot be used to do useful work and unnecessarily burdens cold start.

Requirements. An ideal design for a reconfigurable capacitor bank should meet the following requirements: ⓪ It should provide granular capacitance steps in order to offer the perfect amount of storage for the next task in queue, to minimize charging times. ① It should minimize the leftover energy in the capacitor bank, once task execution is done. ② It must maintain the operating voltage during capacitance reconfigurations, without incurring in energy waste and/or BOR. ③ It should perform the storage scaling at a low-energy cost to maximize the storage's *energy efficiency* (i.e. the ratio between output and input energy). ④ It must be *instantly reactive* to ambient power dynamics and load demands.

Paper contributions. This paper presents *CapDYN*, an ultra-low power energy storage design that *autonomously* adapts its capacitance to the declared energy consumption of the next task in queue. *CapDYN* overcomes the limitations of a fixed energy storage, which can only be tailored to the most energy consuming task within an application (typically, transmission), imposing long recharge times for smaller tasks (e.g., sensing, computing). *CapDYN* is the first *fully-autonomous*, *ultra-low*

power reconfigurable capacitor bank that offers different capacitance steps tailored to sustain a set of tasks with varying energy consumption. *CapDYN* begins charging at its minimum capacitance, to speed up cold start and allow conversion electronics to enter their most efficient conversion range as soon as possible. Each time the bank charges up to the upper voltage threshold, *CapDYN* then *scales* its capacitance up to the requested, task-specific level by seamlessly switching its bank of capacitors without the need of a dedicated MCU. Once the requested energy level is available, the task is launched and *CapDYN* gradually (i.e. each time the bank discharges to a lower threshold voltage) scales down its capacitance back to its minimum level, maintaining the load's voltage range without output regulation. The load's range is maintained as voltage is boosted at each capacitance scale down step. Once the task is finished executing at the minimum capacitance level, leftover energy below the load's minimum operative voltage is minimized, resulting in significant improvements of the cold start times.

Its design is fully scalable and compatible with any energy-harvesting source that can be conditioned to operational-level DC voltage, as well as adaptable to any capacitor bank design. Additionally, we have developed a lightweight intermittent-computing runtime that works in conjunction with *CapDYN*. The runtime receives energy-aware signals from *CapDYN* and maps the energy consumption of tasks to the available energy at each capacitance level of the bank. In short, the main contributions of this paper are as follows:

- We present *CapDYN*, a dynamic energy storage solution which enables *autonomous* scale up/down of its overall capacitance level based on its state of charge, without needing any external MCU.
- We provide a *CapDYN* lightweight runtime that exploits the bank's energy state interface to fully remove sleep mode power consumption and to efficiently and intermittently execute energy-aware, task-based programs.
- We release *CapDYN* hardware design, LTspice model and software runtime as open-source for researchers and the widespread adoption of batteryless energy harvesting systems.

We evaluate *CapDYN*'s *efficiency, scalability* and *reactivity* by the use of a complete LTspice model, using two relevant capacitor bank designs. We simulate *CapDYN*'s deployment in a real energy harvesting application and build *CapDYN*'s first PCB prototype, presented in Figure 1. *CapDYN* improves cold start of up to 97% and execution rates of up to 38% over a fixed capacitor storage. Its power consumption ranges from 11.7 μW (2 levels, D1 bank) to 20.6 μW (6 levels, D2 bank), outperforming the prior state-of-the-art implementations thanks to low-power components and high-efficiency design. Thanks to its fully autonomous control circuitry, *CapDYN* also solves the reactivity-efficiency trade-off that afflicts software-defined reconfigurable storages.

The paper is organized as follows: Section 2 presents the comprehensive analysis of the prior state of the art about intermittent systems and *CapDYN* design requirements. In Section 3, we introduce *CapDYN* and the developed architecture to meet the design requirements. Simulations and HW implementation are discussed in Section 4, while the assessment of the proposed architecture is presented in Section 5. Finally, an outlook of future works and conclusions are presented in Sections 6 and 7.

2 Background and Motivation

Most battery-free energy harvesting devices feature a single, fixed-size centralized capacitor, which is generally as large as the most energy-consuming operation that must be executed without power failure [4, 9]. This introduces a delicate balance if the storage must support operations with varied levels of energy consumption: while a larger energy reservoir can support more substantial tasks, it concurrently prolongs the recharge time for shorter tasks, thereby diminishing the system's overall responsiveness [7].

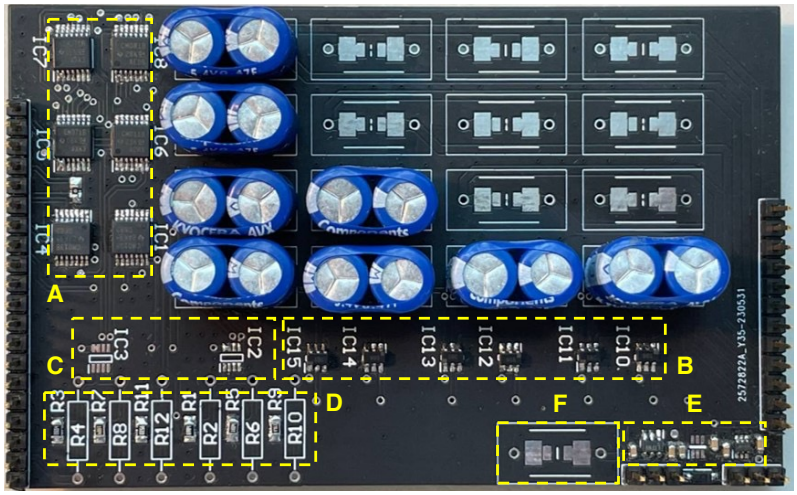


Fig. 1. *CapDYN* prototype board featuring the D1 bank. The PCB can fit up to 16 supercapacitors configured into 4 independent banks, as shown in the center of the layout. The control logic (A) drives the bank's analog switches (B) according to the threshold voltages of the voltage monitors (C), set through resistive dividers (D). A DC/DC converter (E) stabilizes the backup capacitor voltage (F) and provides a stable power source to *CapDYN*'s components.

Table 1. Comparison of the Proposed CapDYN Architecture to Related Works

	Dynamic Charging Time	Reduced leftover energy	Energy Aware Task Execution and scheduling	Auto Reconfigurability
Single Capacitor	✗	✗	✗	✗
Capybara [7]	✓	✗	✓	✗
Reconf. SC [13, 27],Variable Storage [12]	✓	✗	✗	✓
Catnap [22]	✓	✓	✓	✗
UFoP [15], Flicker [16]	✓	✗	✓	✗
GRANT [34]	✓	✓	✓	✗
Stash [3]	✓	✗	✓	✗
Morphy [30]	✓	✓	✓	✗
CapDYN	✓	✓	✓	✓

As highlighted in prior articles [7, 22, 24, 30], a batteryless system faces multiple power supply-related challenges. These include achieving operational voltage from an empty storage (cold start), sustaining tasks with different energy requirements, using stored energy efficiently, and managing intermittent execution. Recent works have focused on addressing these challenges by providing variable storage upon software request [7, 30] or autonomously [13, 27] switching between a small and a large capacitor to achieve faster startup [12], or by considering per-peripheral capacitors [15]. Among these works, reconfigurable capacitor architectures [7, 30] provide a clever way to solve the fast recharge/sustained operation tradeoff. However, connecting multiple capacitors and enabling their software-based reconfiguration introduces additional overhead, increasing complexity and power requirements.

2.1 Differences from the State-of-the-Art

Table 1 provides a brief comparison of *CapDYN* and existing reconfigurable storage work. *CapDYN* advances the prior art by combining all features presented in the literature with the ability of

auto-reconfigurability, meaning that it does not need an external microcontroller or software control to manage the capacitor bank.

① Dynamic Charging Time and reduced leftover energy.

A simple solution to provide dynamic charging time is to provide two capacitance steps [12, 13, 27], one for fast startup, one for storing extra energy if the input power is greater than the end-node's active power consumption. While this approach is an important example of autonomous capacitance scaling, the purpose of providing a larger capacitance step is to store excess energy without burdening cold start, not to support larger tasks. Indeed, larger tasks relying on the larger capacitance steps might never execute, unless the power input is higher than the node's active power consumption.

This shortcoming is improved with Capybara [7], where storage size is explicitly matched to task energy request, be it large or small, and the charging of higher capacitance levels is decoupled from active load power consumption, as the load is only activated when sufficient energy for task execution has been harvested. When scheduling a task, Capybara connects a certain number of parallel capacitors sufficient to match the task's energy requirements (termed *energy mode*). The storage is only modified when scheduling a task, then it acts as a fixed-size capacitor (i.e. it does not reconfigure throughout charging and task execution). Given this reconfiguration scheme, to minimize leftover energy, Capybara must employ an output voltage boost to generate an operative voltage as the bank dips below the load's lower operative voltage. This approach surely favours simplicity, but it comes at the cost of some extra infrastructural power consumption and conversion losses due to the output boost converter.

Yang et al. further expanded the software-defined capacitance of Capybara through *progressive scaling* in Morphy [30]. The same idea, with an improved implementation, is also employed in *CapDYN*. The progressive scaling mechanism consists in gradually reconfiguring the capacitor bank to scale up/down the bank capacitance level throughout charging/discharging. Charging begins at a minimum capacitance level, so that operative voltage is achieved as soon as possible. Then, capacitance is scaled up either when an upper charging voltage is reached (*CapDYN*) or when the bank is deemed optimally charged (Morphy), to make room for more energy. When energy is deemed sufficient for task execution, bank discharge begins, and capacitance is gradually scaled back down to the minimum level. This provides three advantages over Capybara, which sets up the storage size before charging/discharging and does not reconfigure the bank until a new task is scheduled. First off, operative voltage is achieved earlier, and then it is maintained through progressive scaling. Conditioning electronics circuits have a much higher efficiency (5% vs 90% for commercial models such as Texas Instruments' BQ25505) when their output voltage - which is where the dynamic capacitor is placed - grows above a cold start voltage threshold. Progressive scaling allows *CapDYN* and Morphy to overcome cold start much sooner than a fixed capacitor tailored for the same task (Figure 10), meaning that conversion electronics work more efficiently sooner, resulting in more stored energy. Secondly, at each scale down, capacitors within the bank are reconfigured to boost the bank's voltage, until the minimum capacitance is reached: leftover energy - energy below operative voltage - is thus effectively reduced with respect to the fixed capacitor needed for an equivalent task execution. As mentioned in Section 1, a lower leftover energy also improves cold start speed. Thirdly, voltage boosting is achieved without using an output boost component as in Capybara, which introduces additional consumption and inefficiencies.

CapDYN further improves the efficiency of the progressive scaling mechanism with respect to Morphy. Morphy implements progressive scaling by charging its entire bank, and then separating a subset of task-tailored capacitors during execution. Charge-pump boosting is then employed between the core array and the task capacitors, to maintain the task capacitors in the load's voltage

range. While this improves the flexibility of Morphy in terms of load voltage range and scaling policies, it also causes large dissipative transients due to frequent charge transfers among differently charged capacitors, which largely limit the efficiency of the boosting process. *CapDYN* does not feature a core/task capacitor separation: instead, *CapDYN* avoids the inefficiencies they carry by tailoring its whole bank's capacitance levels to match the different task energy requests. This further speeds up the charging process with respect to Morphy, where overall bank capacitance is not explicitly matched to a single task, as the task-matching job is left to the core/task array separation. As a bonus, by matching capacitance steps with task requests, we obtain a hardware *energy state interface*: by autonomously monitoring the digital lines used to pilot the bank switches, *CapDYN* can fully power-gate its load until it reaches the stage needed for the next task's execution. This energy state interface is thoroughly explained in Section 3.3.

② **Auto Reconfigurability.** Software-controlled dynamic bank solutions allow greater scheduling flexibility, but do not solve the reactivity-efficiency trade-off. Indeed, if reconfiguration is tied to the activation of a power-hungry MCU [7] or periodic state of charge measurements [30], the more the system will require reconfiguration, the more energy will be wasted. Autonomous control reduces power consumption, but also solves the reactivity-efficiency tradeoff: active monitoring and reconfiguration do not come at an added cost, unlike with software-controlled solutions, as evaluated in Section 5.1.1. For these reasons, *CapDYN* was designed to operate autonomously, without an MCU. Although the current prototype is implemented using discrete components, such as voltage comparators and logic gates, *CapDYN* could be integrated into a single chip, further reducing power consumption and improving its energy efficiency. To the best of our knowledge, *CapDYN* is the first dynamic capacitance implementation that combines auto-reconfigurability, scheduling support, and easy scalability. Other autonomous solutions presented in literature are focused on harvesting excess energy, rather than providing dynamic capacitance. In STASH [3], the front-end, federated capacitors are tailored to sustain peripherals throughout execution, while a back-end capacitor is available to autonomously collect energy when the energetic needs of the load are satisfied (i.e. the front-end storage is full and task execution needs to wait). This mechanism, although inefficient due to the charge transfer through a Shottky diode from the back-end capacitors to the front-end capacitor, manages to store extra energy without slowing down cold start, as the back-end capacitor is connected to the harvester output only in times of energy abundance. A similar result to Stash's is obtained in [12, 13, 27], but extra energy is only collected if the power input is higher than the load's active power consumption. *CapDYN*'s autonomous control, coupled with its *progressive scaling*, manages to reactively expand, even if the next task energy level has been reached and task execution needs to wait, essentially providing Stash's functionality without extra hardware.

③ **Energy-Aware Task Execution and Scheduling Support.**

Energy aware execution requires knowledge of both the tasks' energy consumption and the storage's state of charge. While the former is easily achieved through offline profiling, the latter requires either active measurements, threshold monitoring or some other form of *energy awareness*. UFoP [15] and Morphy [30] actively poll their storage voltage, incurring in a considerable energy overhead as this requires MCU intervention, while Flicker [16] avoids this energy waste through interrupt-based monitoring. Flicker proposes an interesting solution, which couples energy awareness and charging prioritization through two programmable voltage monitors (charge and interrupt controllers) for each peripheral. The charge controller enables peripheral capacitor charging when the first-stage capacitor (which powers the core) grows above a first voltage threshold. A lower first threshold corresponds to a higher peripheral priority. The interrupt controller then issues an interrupt when the peripheral capacitor is charged above a second voltage threshold, which

indicates that the task running on said peripheral can execute. Thanks to the reconfigurability of the controller thresholds, prioritization and task energy can be modified, ensuring energy-aware scheduling freedom. This comes at the cost of extra power consumption, as the monitoring structure is replicated at each peripheral. Unfortunately, we can only speculate as to how charge is transferred among capacitors, and whether this mechanism is efficient, as details of the charge controller are not provided. We would need to know how the capacitors are connected to the harvester output and to each other, to understand if the charging voltages of each peripheral must be in some relative relationship with respect to one another: this is what is hinted in [3], Section 4, meaning that this prioritization mechanism faces constraints posed by the ranges of each peripheral. *CapDYN* features a more efficient approach to energy awareness. Its digital lines have a twofold purpose: the first is to pilot bank reconfiguration, the second is to inform the load about the current capacitance step, which also represents the bank's state of charge, through the *energy state interface* (Section 3.3). As we associate each capacitance step with the tasks it can sustain, through hardware-software co-design, by knowing at which capacitance step the system is charged, we have instant feedback on which task can be executed. Thanks to *CapDYN*'s *energy state interface*, the load running *CapDYN*'s runtime is only activated when the main bank holds sufficient energy to execute the next task. This feature greatly improves energy efficiency, as it removes the overheads of load sleep mode power consumption and/or periodic polling of the bank's state of charge, which burden software-controlled dynamic capacitors (Section 3.4). In addition, charge prioritization is not required, as *CapDYN* centralizes its charge, meaning it does not need to be multiplexed over federated capacitors. This removes the complexity and energy overhead of hardware prioritization approaches such as Flicker's.

Batteryless devices need to execute tasks which pose constraints not only on *how much* energy they need to successfully execute, but also on *when* they need the requested energy. Such *time-constrained* tasks need to execute *reactively*, as an immediate response to an event. *Recharge time* determines whether a time-constrained task will execute reactively, as the system is unresponsive during recharge [32, 33]. As previously explained in Section 1, a fixed buffer cannot reactively respond to different energy consumption requests. While the end-node must only know the storage's state of charge to schedule energy-constrained tasks, scheduling time-constrained tasks requires either *reducing recharge times*, by having a storage that achieves operating voltage as soon as the requested energy is available, or *pre-charging* the amount of energy that must be sourced reactively. Catnap [22] logically (i.e. not in hardware) partitions its fixed-size energy buffer, reserving charge for reactive response to events and executing an energy-constrained task queue when extra energy is available. This allows fine-grained modification of the pre-charging energy level, but it requires constant energy level monitoring, which is energy-inefficient. Similarly, Capybara [7] features a hardware pre-charge feature for time-critical responses: a section of Capybara's bank is charged with the requested energy and then isolated, as it awaits for the time-critical task. This simple mechanism can suffer from self-discharge, as the isolated bank loses charge due to leakage. On the other hand, *CapDYN* enables both pre-charging and recharge time minimization. Pre-charging is achieved by not depleting the lower capacitance steps, which are reserved for reactive responses. Energy reservation is enforced through *CapDYN*'s digital output lines. Additionally, the reserved energy is not isolated as in Capybara, but it is stored in the main bank, which keeps charging normally, thus counteracting the leakage effect on the reserved pre-charge. Recharge times are also minimized thanks to *CapDYN*'s progressive scaling mechanism, which not only matches capacitance step to reactive task request, but also improves conversion efficiency by allowing improved cold-start overcoming (Section 5.2). Additionally, if *persistent timekeepers* such as CHRT [8] are integrated into *CapDYN*, we can infer average power input levels by measuring charging time, which opens the possibility to adapt the schedule to the harvested power level.

2.2 Centralized and federated solutions

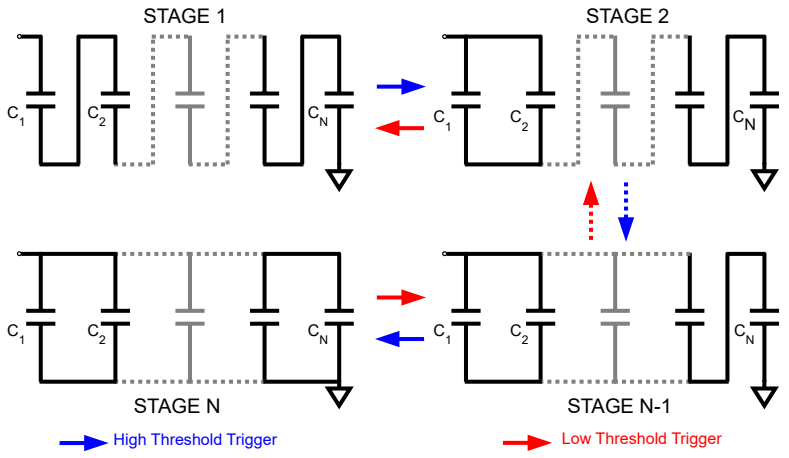
A radically different approach to centralized, variable capacitance is to *federate* the energy storage across peripherals, by dedicating a fixed-size capacitor to each peripheral. By only activating the charging of the required peripheral capacitors for a given task execution, overall storage capacitance can be reduced with respect to a fixed-size centralized capacitor, especially in the case of applications with diverse energy requirements. This approach has three major shortcomings with respect to centralized solutions. Firstly, even though the time to operative voltage is minimized across all required peripherals, task execution needs to wait for the charging of the slowest peripheral. This means that the peripherals which already have achieved operating voltage sit idle and draw power from their storages. With a centralized solution, by tailoring the capacitance to a task, we minimize not only the time to operative voltage, but also the time to task execution, so when useful work is done, without having to consider peripheral charging synchronization delays and energy waste. Secondly, in federated approaches energy *is fragmented* across peripheral capacitors, as debated in [34], which means that even though the overall stored energy would be sufficient to launch a task, each peripheral is below operating voltage and cannot operate. Solving this problem requires sub-optimal charge transfers among different peripheral capacitors, which reduces the energy efficiency of the system [34]. Centralized storages such as *CapDYN* avoid fragmentation by design and do not require lossy defragmentation techniques. Besides for the purpose of defragmentation, the routing of charge from one capacitor to a differently charged one regularly takes place in federated storages (e.g., stash capacitor to peripheral capacitors [3], MCU capacitor to peripheral capacitor [15, 16]), and gives way to large energy transfer losses. Thirdly, federated storages dimension the peripheral capacitor according to the consumption of a single task using that peripheral. If a peripheral requires different energy budgets for different tasks, this approach inherits the shortcomings of fixed capacitors when scheduling tasks with varied energy draws. In the case of centralized systems such as *CapDYN*, Capybara and Morphy, we can remap different tasks using the same peripheral to different capacitance levels, overcoming the mentioned limitation of per-peripheral capacitors. Hester et al. [15] argue that, unlike federated storages, centralized solutions are vulnerable to the so-called "tragedy of the coulombs", which is what happens if a faulty task or component can access and wrongly deplete a shared storage. Centralized systems can defend themselves from this problem by implementing *energy watchdogs* [30], which power-gate peripherals through switches if the energy draw goes above the expected level. *CapDYN* is immune to this problem, as tasks are triggered as soon as the bank has sufficient energy to complete them. So, a bugged task will, at worst, not complete. Blacklisting can also be implemented to avoid re-execution. If pre-charging is activated, we can safeguard the pre-charged energy through an energy watchdog which leverages *CapDYN*'s capacitance step knowledge.

2.3 Other relevant works

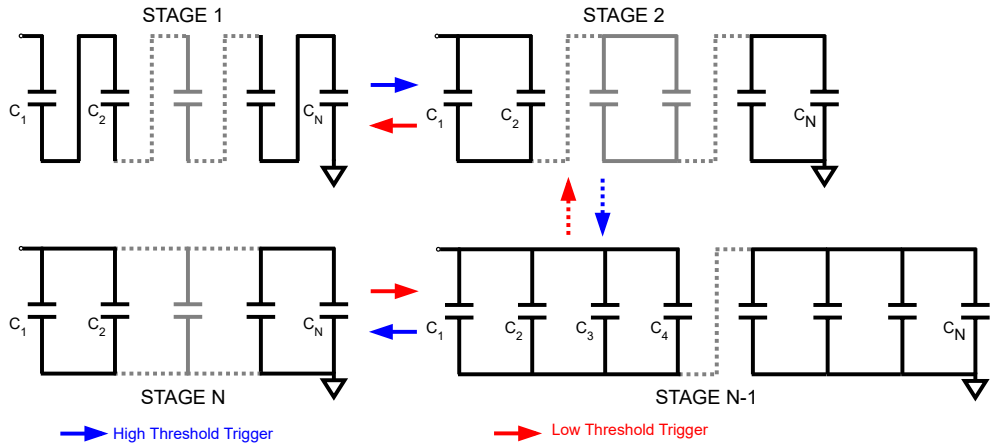
Two recent works, Culpeo [25] and CapOS [5], propose interesting solutions which could be integrated with *CapDYN*. Culpeo [25] considers the voltage drop caused by the equivalent series resistance in capacitors. This voltage drop might lead to an unexpected power failure even though the stored energy remains in the capacitor. CapOS [5] considers the capacitor degradation due to repeated charging and discharging operations. The benefits of the proposed solutions in these works can be applied to *CapDYN* to further improve its performance. We leave these aspects out of the scope of this paper.

3 *CapDYN* Design

With *CapDYN*, we introduce a novel dynamic energy storage which *automatically* adjusts its overall capacitance based on its current state of charge. *CapDYN* executes these capacitance adjustment in



(a) Design 1.



(b) Design 2.

Fig. 2. The *progressive scaling* capacitor reconfiguration pattern for *CapDYN*, in two bank design variants. The initial configuration of both banks is *stage 1*, where all the capacitors are connected in series. As the stored energy increases, each design differently reconfigures the connections between the capacitors in successive steps to connect them in parallel. At *stage N*, all the capacitors of both banks are connected in parallel.

steps though what we term *progressive scaling*, as shown in Figure 2: upon system cold start, *CapDYN* begins charging at its minimum capacitance level, then it *progressively* increases its capacitance level each time an upper voltage threshold is reached. When the capacitance level required to execute the next task *without power failure* is fully charged (Section 3.3), *CapDYN* enables task execution: as the bank discharges, *CapDYN* decreases its capacitance level each time a lower voltage threshold is crossed, until the minimum capacitance level is discharged down to the lower voltage threshold. At this point, *CapDYN* has extracted all viable energy from its bank, having minimized leftover energy through the aforementioned progressive scaling mechanism. Through the scaling process, the output voltage is limited, as the output voltage is maintained between an upper and a

lower voltage threshold. Besides this, the progressive scaling approach gives *CapDYN* the following advantages with respect to a fixed capacitor solution:

- By tailoring *CapDYN*'s capacitance steps to each task's consumption, we minimize charging times across tasks with varied energy requirements, all the while sustaining their execution without incurring power failures.
- *CapDYN* reaches operative voltage sooner and maintains it throughout charging, thanks to its *progressive scaling* approach: this means conditioning electronics operate in their most efficient range (that is, out of cold start) for more time.
- *CapDYN* can discharge the bank down to its minimum capacitance level, effectively reducing the leftover energy - that is, energy that cannot be used for powering the load and thus unnecessarily burdens cold start.

CapDYN system's architecture is composed of four parts, which are discussed in the following section and represented in Figure 3. **(1)** The energy storage *scaling logic*, which enforces *progressive capacitance scaling*; **(2)** the capacitor bank (or *main bank*) with its reconfigurable switches, piloted by the scaling logic; **(3)** the *I/O flow control*, which orchestrates the power flow into and out of the main bank and **(4)** the *energy state interface*, which allows the load to indicate at which energy level it wants to be powered, in order to execute its next task. The scaling logic is in charge of monitoring the main bank voltage and of accordingly piloting the progressive scaling of the bank's capacitance through the bank's switches. The proposed scaling logic can be matched with infinite capacitor bank designs. In this paper, as a proof of concept, we focus on two convenient capacitor bank designs. The I/O flow control ensures that the main bank is only ever charged/discharged when the backup capacitor can sustain the scaling operations. The energy state interface allows the bank's power output to be activated by the I/O flow control only once the load's next task energy level has been achieved, in order to avoid wasting energy in load sleep mode power consumption. The power output is then disabled when the main bank is fully discharged, in order to avoid successive cold starts. Both the scaling logic and the I/O flow control are fully hardware-based, as they forego active control (i.e., an MCU) to carry out their tasks. As we will see in Section 5.1.1, this solves the reactivity-efficiency tradeoff mentioned in Section 1. Besides being flexible in terms of capacitor bank design, *CapDYN* is also compatible with any harvesting source which can be conditioned to output DC current/voltage, through the required source-specific energy conversion circuitry.

3.1 The progressive capacitance scaling logic

CapDYN's scaling logic relies on monitoring the main bank's voltage to estimate the stored energy within the bank and to autonomously trigger scale up/down of the bank capacitance. Capacitance scale-up/down is achieved by modifying the connections between the bank's capacitors through a series of switches, which are controlled by the scaling logic. Capacitance scale-up triggers when the main bank's voltage grows above an upper threshold V_{SH}, while scale-down triggers when the voltage falls below a lower threshold V_{SL}. Note that $V_{SH} > V_{SL}$. In this way, the main bank is limited in the $[V_{SL}, V_{SH}]$ range at all times during normal operation. In its most basic form, the scaling logic is a shift register which shifts its bits at each scaling event. Each bit pilots a subset of the bank's switches to implement different overall capacitance steps. By using different bit combinations to control the bank's digital lines, *CapDYN*'s scaling logic can be adapted to infinite reconfigurable capacitor bank designs.

3.1.1 Avoiding Brown Out, Overcharging and Loops in Progressive Scaling. Switching thresholds V_{SL}, V_{SH} require careful design in relation to the load's voltage range, V_L, V_H and to the biggest voltage drops/spikes associated to capacitance variations. Voltage spikes/drops upon transitions

depend on the selected bank design, specifically on the reconfiguration strategy and on how the capacitors within the bank are dimensioned, as discussed in Section 3.2.

In order to directly (i.e., without direct output regulation) supply an operating voltage to the load throughout the progressive scaling operations, the main bank's output voltage must be limited to the load's operating range. Indeed, if capacitance scale-up/down gives way to a bank voltage spike/drop that exits the load's range, the load could be either damaged or suffer from a power failure. At the same time, if these voltage spikes are larger than the $[V_{SL}, V_{SH}]$ range, a *loop condition* occurs. In fact, if upon reaching V_{SH}, capacitance scale-up generates a voltage drop that reaches V_{SL}, scale-down is triggered right after scale-up, and scale-down comes with a voltage spike which newly exceeds V_{SH}, resulting in a new scale-up. In order to respect the load's voltage range throughout scaling and make as much room as possible to accommodate scaling-associated voltage spikes/drops, we select $V_{SL} = V_L$, $V_{SH} = V_H$. In the next paragraph, we highlight two capacitor dimensioning solutions which can be matched to different load voltage ranges.

3.2 Main bank design: switching strategy and capacitor dimensioning solution

CapDYN's capacitor bank is composed of multiple capacitors, whose connections are reconfigured through switches piloted by the scaling logic. A capacitor bank is fully defined by its *switching strategy* and by its *capacitor dimensioning solution*. The switching strategy defines how the capacitors of the bank are arranged in order to achieve varying bank capacitance steps, while the capacitor dimensioning solution expresses how each capacitor within the bank is dimensioned with respect to each other.

CapDYN is compatible with multiple bank designs. Figure 2 shows the switching strategies of the two banks we evaluate in this paper. Design 1 (D1) ensures a large capacitance multiplication range with limited components, and because the component's number increases linearly with stage number, it is easily scalable. Design 2 (D2) is less scalable, as the components' number increases twofold for each additional stage, but features more energy efficient transitions. Section 5.1 evaluates their differences.

D1's *switching strategy* (Figure 2(a)) features all bank capacitors connected in series in the initial stage/level, in order to set the minimum capacitance for quick cold-start overcoming. The bank then progressively scales up its capacitance each time it reaches V_{SH}, by connecting a capacitor in parallel with C_1. The maximum capacitance level is reached when all capacitors are connected in parallel to C_1. Bank discharge/task execution is enabled when a desired energy level, represented by a certain capacitance level, is achieved (Section 3.3). When discharging, the bank progressively scales down its capacitance each time it reaches V_{SL}, by removing a capacitor from the parallel to C_1 and connecting it in series, providing the benefits outlined in Section 2.1.

For an N-stage D1 *CapDYN*, we dimension the capacitors as multiples of C_1, which we term *unit capacitance*, according to Equation (1):

$$\begin{aligned} C_2 &= C_1 \\ C_i &= 2^{i-2} \cdot C_1, i = [3, N] \end{aligned} \tag{1}$$

For the D1 switching strategy, there exists more than one *capacitor dimensioning solution* that constrains the voltage spikes/drops within the load voltage range. We focus solely on one viable solution, as the main contribution of this paper lies in the autonomous bank reconfiguration, rather than the switching strategy or the capacitor bank itself. With the dimensioning solution here outlied, the main bank's capacitance spans from $C_{min} = \frac{2^N}{3 \cdot 2^{N-2^2}} C_1$, up to $C_{max} = 2^{N-1} C_1$, in $N-1$ steps. In the D2 bank (Figure 2(b)), all the capacitors have equal size and they begin charging in series and at each scale up half of the bank is parallelized, until all capacitors are in parallel. We refer the reader to [30] for its full presentation, for the sake of space.

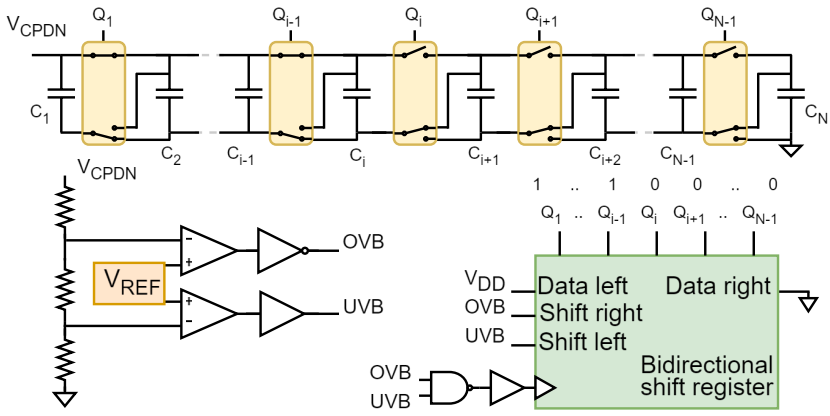
Once the bank is defined, it must be coupled to the desired load voltage range. In fact, switching thresholds V_{SH}, V_{SL} must respect the load's voltage range as well as make room for the voltage drops/spikes due to reconfiguration. From LTspice simulations of the D1 bank, the capacitance scale-up/down voltage drops are larger when switching between larger capacitance stages. Ultimately, switching to/from the C_{max} stage halves/doubles the main bank voltage, respectively. For the D2 bank, this happens at each transition. It follows that loads with $V_{SH} \geq 2V_{SL}$ will fully exploit both capacitor bank designs, managing charge until the last stage, while loads with $V_{SH} < 2V_{SL}$ will only be able to use D1 and will stop charging to an i-th stage, with $i < N$, to avoid brownout. Most low power integrated circuits fit the former case, as they operate in the range of [1.8, 3.6] V. If the load falls in the latter case, *CapDYN* achieves an equivalent C_{min}, at the cost of a slightly smaller C_{max}. With the two designs here outlined, we fit *CapDYN* with any load voltage range, without the need for an additional output voltage smoothing/regulating stage.

The last point of the main bank design is concerned with matching the switching logic to the switches' control signals needed to achieve the switching strategies presented in Figure 2. As shown in Figure 3(a), we can implement both of the desired switching strategy by managing the connection of each capacitor with two switches, driven by the same control signal $\texttt{Q}_\texttt{i}$. We design the progressive scaling logic as an N-bit *bidirectional shift register* piloted by a window-type voltage monitor with two programmable thresholds coinciding with the capacitance scaling voltages, V_{SH}, V_{SL}. The voltage monitor provides two signals, OVB and UVB: OVB is asserted when the main bank voltage grows up to the upper scaling voltage, while UVB is asserted when monitored voltage falls to the lower scaling threshold. As shown in Figure 3, OVB and UVB trigger the *right shift* of a high bit and *left shift* of a low bit of the bidirectional register bits, respectively, which will then control the main bank switches, responsible for the capacitance scale up/down. We refer to the N-bit bank control signal as Q or bitmask, alternatively. For D1, Q directly controls the capacitor switches, while D2 requires an additional AND logic to produce the desired switching strategy. By implementing different bitmasks, we can extend the progressive scaling logic to other designs.

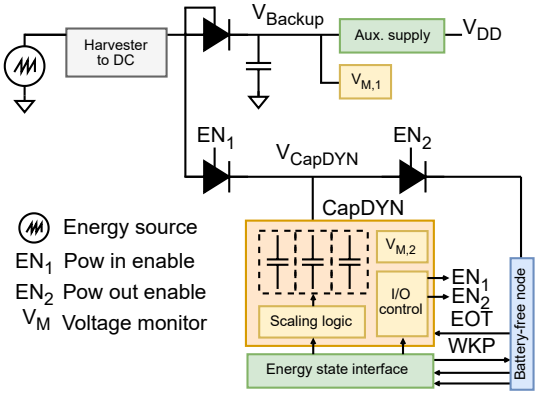
3.3 Power supply design and input/output power flow management

CapDYN is designed around ultra-low power energy harvesters at the edge of remote energy scavenging, such as small photovoltaic panels, microbial fuel cells, thermoelectric, or even piezoelectric generators. Any node operating directly from such energy sources must rely on a *harvester-to-DC* conditioning circuit, as these sources produce voltage outputs that need to be either boosted, bucked or rectified to operational-level DC voltages. We designed this circuit to be *harvester-agnostic* by considering conditioned DC voltage as *CapDYN*'s power input. This assumption separates the energy harvester conditioning circuit design from that of our system, virtually making it compatible with any harvester, as long as DC-voltage can be generated from it.

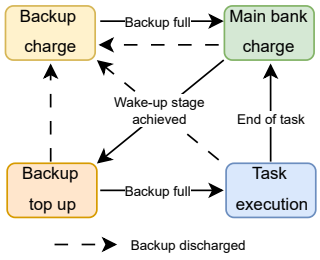
The output of the selected harvester, which we consider to be already conditioned for the previously mentioned reasons, is the input of two separate but interdependent energy banks within *CapDYN* (Figure 3(b)). One is the *main* supply (V_{CapDYN}), tapping into *CapDYN*'s main capacitor bank to power the intermittent node, and the other is the *backup* supply (V_{Backup}) hinging on a backup capacitor to power *CapDYN*'s components. The power flow into and out of these two banks is controlled thanks to reverse-current blocking ideal diodes, piloted by the I/O control's two output signals, $\texttt{EN}_1$ and $\texttt{EN}_2$, as shown in Figure 3(b). The I/O control implements a finite state machine (FSM), shown in Figure 3(c). This FSM makes sure that power can flow into/out of the main bank solely when the backup capacitor can correctly supply *CapDYN*'s components and thus grant *CapDYN*'s self-scaling capabilities. The FSM also enables the main bank's output solely when *CapDYN* fully charges the capacitance level needed to execute the next task without power failures, which we term *wake-up* stage. By fully deactivating the load until useful work can be done, we avoid



(a)



(b)



(c)

Fig. 3. The top image (a) shows the D1 bank's schematic for the i-th stage. The window-type voltage monitor triggers a left/right shift of the bidirectional shift register's bits. On the top, we can see the switch connections driven by the shift register's outputs, Q_i. The middle figure (b) shows *CapDYN*'s block diagram. The bottom image (c) highlights the I/O control's state machine which controls the input and output flow of both *CapDYN* and its backup capacitor.

sleep mode power consumption on the load side and improve energy efficiency (Section 3.4). The FSM also safeguards *CapDYN*'s and the load's operation in the case of malfunctions: $\texttt{EN}_2$ interrupts the output in the case that either the backup or the main bank are too discharged to provide operative voltage to either the *CapDYN* infrastructure or the load. In its basic functioning, the FSM ensures that the main bank begins charging when the backup is full. Then, when the wake-up stage is achieved, the backup is briefly topped up to ensure it can sustain *CapDYN* during execution. Once that happens, the load is activated and the task can execute. The load then power-gates itself by indicating to the I/O control when it has concluded its task.

The infrastructure which takes care of monitoring whether the wake-up stage has been achieved or not is the *energy state interface*, which comprises the *wake-up register* and the *task match* circuitry. The wake-up register is connected to the load via a four-wire interface for the N-stages *CapDYN* design. The first two wires, data and clock lines, allow the MCU to set up the wake-up register for the next task execution. The third wire, the `WKP` signal, is the output of the task match circuitry, which asserts whether the wake-up stage and the current stage of the *CapDYN* capacitor bank is matched. The final wire is dedicated to the `EOT` - end of task - this signal is asserted by the MCU when it completes a task and the next task's wake-up stage has not been achieved yet, resulting in the load's power-gating. The wake-up register is first initialized during *CapDYN*'s cold start through a power-on signal: this allows *CapDYN* to configure a default stage at which the load is first supplied with power, and can thus initialize its runtime, outlined in Section 3.4. At the end of a successful task execution within the runtime, the load configures *CapDYN*'s wake-up register with a bitmask representative of the wake-up stage. The task match circuitry uses this bitmask to understand from the switches' control signals, produced by the progressive scaling logic, if the wake-up stage has been achieved or not. Once the task match circuitry signals that the wake-up stage has been achieved (through signal `WKP`), the load's supply is enabled through $\texttt{EN}_2$ and the next task can be executed. $\texttt{EN}_2$ then turns off the output when the load has concluded its current task and it has set up the next one, through signal `EOT`.

We power *CapDYN*'s load directly from the main bank's unregulated voltage, which is limited to the $[V_{SL}, V_{SH}]$ range, while we power *CapDYN*'s components by regulating the backup supply's voltage. Voltage regulation is deemed necessary to ensure the combinational logic's correct operation: the regulated voltage level is set to the minimum possible value to minimize power consumption. We dimension the backup capacitor to withstand the longest atomic operation of the application to which we are fitting the hardware, considering the worst-case scenario of no input power. Backup capacitor size must be kept as small as possible: as its size entirely depends on task execution time and *CapDYN*'s idle consumption, *CapDYN*'s components must be kept as low-power as possible, and time overhead due to backup capacitor charging must be considered when straying away from applications which do not entail short, intermittent tasks. A good rule of thumb is to keep the backup capacitor dimension below the capacitance of the first stage so that tasks with stage-1-level energy consumption are not excessively slowed down.

3.4 *CapDYN* Task-based, Energy-Aware Runtime

CapDYN utilizes an energy-aware, task-based intermittent runtime model. The *CapDYN* runtime executes each task by considering if the task's energy requirement is satisfied by *CapDYN*' state of charge, to avoid power failures. Therefore, *CapDYN* requires an offline task energy profiling process, which is explained in Section 4.4, to obtain a mapping of each task to a certain *CapDYN* stage (termed *task-to-stage* mapping). This runtime takes full advantage of *CapDYN*'s wake-up stage configuration and I/O control. As explained in Section 3.3, *CapDYN*'s I/O control activates the load only once the stage held in the wake-up register has been achieved. Figure 4 presents a brief overview of the runtime's flow chart. Once the *CapDYN* hardware activates its load for

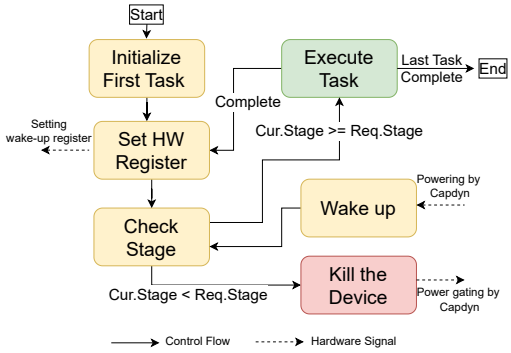


Fig. 4. Overview of the runtime's flow chart.

the first time, the *CapDYN* runtime initializes the first task by configuring the wake-up register with the first task's required stage. The current stage of the capacitor bank is then checked. If the required stage has not been achieved, the runtime triggers the *CapDYN* hardware to cut the load's power. The hardware will newly power the load when *CapDYN* achieves the configured wake-up stage. While waiting, the load is fully off and does not draw power. Instead, if the required stage has been achieved, the runtime executes the current task and configures the next task's required stage into the wake-up register. This triggers a new bank stage check, which leads to the previously explained energy-aware execution choice and allows the *CapDYN* runtime to progress through its tasks without power failures, until the final task is completed.

With *CapDYN*, the load does not need to directly monitor *CapDYN*'s state of charge to achieve energy awareness - the awareness that the load needs, which is the one telling the load if sufficient energy is available to execute the next task - is built into the hardware. This hardware approach allows us to fully solve the reactivity-efficiency trade-off mentioned in Section 1. In fact, *CapDYN* is instantly reactive to input power bursts, without impacting its efficiency with software-based state of charge measurements. With *CapDYN*, we overcome both polling and interrupt-driven reconfiguration approaches. The first approach wastes a significant amount of energy to activate the load for periodic monitoring, and it has to draw assumptions about the minimum monitoring period. Besides, as the system is not reactive/reconfigurable outside of its monitoring periods, it potentially misses bursts of incoming energy [15, 30]. The second approach requires the load to be responsive to state-of-charge-driven interrupts [16], so it forces the load to be in sleep mode, wasting energy with respect to *CapDYN*'s approach, which is fully power-gating the load until enough energy has been harvested.

4 Implementation

In this section, we discuss *CapDYN*'s hardware design and the implementation of its runtime. We then provide a complete LTspice simulation of *CapDYN*, which is an accurate dimensioning tool for integrating *CapDYN* into batteryless applications. Additionally, the simulator is useful for adapting *CapDYN* to different capacitor banks. We take advantage of this feature to simulate the two bank designs presented in Section 3.2. We also implement the D1 capacitor bank and the progressive scaling logic in hardware, to validate the model and evaluate the effects discovered during the LTspice evaluation (Section 5.1). Ultimately, we present the implementation of a lightweight C runtime which allows sensor nodes powered by *CapDYN* to run task-based applications in an ultra-efficient manner, fully exploiting *CapDYN*'s energy state interface.

4.1 HW Implementation

In designing *CapDYN*'s schematic, we sought the lowest power-consuming solutions to minimize *CapDYN*'s energy overhead and extend its autonomy upon power failure. The switches chosen for the main capacitor bank are Analog Devices' ADG801 (SPST) and ADG819 (SPDT). Their 1 nA current consumption ensures minimum burden on the backup supply, while their low on-resistance (0.4 and 0.8 Ω max, for the ADG801 and ADG819, respectively) minimizes the charging/discharging energy loss and capacitor bank imbalances. These capacitor bank switches conduct equally well in both directions, allowing both charging and discharging operations. Unlike the capacitor bank switches, the power input/output's ideal diodes (Figure 3) need to provide reverse current protection: we thus use Vishay's reverse current blocking load switches SIP32431, which consume only 10 pA and introduce 130 nA of leakage when reverse biased.

Two Analog Devices' LTC2934-2 voltage monitors are then used for piloting the progressive scaling logic and the I/O power flow management described in Section 3.3. Each voltage monitor has a quiescent current of 500 nA and features a resistive divider at its input, through which we set two threshold levels on the tracked voltage. The selected voltage monitor features ultra-low leakage at the two control inputs (2 nA combined). The maximum resistor divider size is set to achieve a 1% voltage detection error due to the leakage of the control inputs. The divider is dimensioned so that the divider current is 100 times larger (so, 0.2 μA) than the inputs' leakage at the lowest monitored threshold. The first voltage monitor is configured to track the main capacitor bank voltage and detect when it crosses V_{SL} = 1.8 and V_{SH} = 3.65 V. Whenever the bank voltage grows up to 3.65 V (or falls to 1.8 V), the voltage monitor triggers capacitance scale-up (or down) thanks to the progressive scaling circuit. These two voltage values were chosen to match our load's maximum voltage supply range [1.8, 3.6] V, leaving a small 50 mV margin to ensure correct scaling even in the case of the largest voltage spikes/drops (Section 3.2). We then set *CapDYN*'s digital components' supply line to 3 V, obtained from the backup capacitor through Analog Devices' LT6656-3 precision voltage reference. Analog components such as the bank's switches are directly powered from the backup capacitor. The second voltage monitor is set to track the backup capacitor voltage, in order to prevent power input/output flow from the bank when the backup cannot grant correct powering of the *CapDYN* infrastructure (Section 3.3). The lower backup voltage threshold is selected to prevent the clamping diodes of the analog switches from entering conduction. The switches' supply, and so the backup voltage, should not drop below the switches' maximum input voltage (V_{SH}) plus the diode's forward drop (0.3 V), which amounts to 3.85 V. The upper backup voltage threshold is set to 5.5 V, corresponding to the our reference application's[1] DC/DC converter's (the BQ25502) maximum charging voltage. The I/O flow control first charges the backup capacitor up to 5.5 V, then allows power to flow into *CapDYN* until the backup capacitor is discharged below 3.85 V, where the main bank input is again interrupted, to favour backup recharge instead. Ultimately, for the progressive scaling logic and the energy state interface, both showcased in Figure 3, we relied on Texas Instruments' ultra-low power digital components: CD4000 logic gates (CD4011 NAND, CD4071 OR, CD4081 AND) and flip flops (CD4013).

4.1.1 CapDYN prototype. The first *CapDYN* D1 bank prototype can be seen in Figure 1. The board features the possibility of using a single capacitor size to obtain the 1-1-2-4 capacitance setting explained in Section 3.2, for the sake of reducing the bill of material. This board can be matched with the BQ25505EVM-218 DC/DC converter evaluation board to swiftly harvest energy from low DC voltage energy sources (Figure 5). The BQ25505 converter can also be matched with an AC/DC rectifier to harvest energy from AC voltage sources such as piezoelectric generators.

[1]The battery-free application that we reference in order to evaluate *CapDYN* is explained in Section 5.2

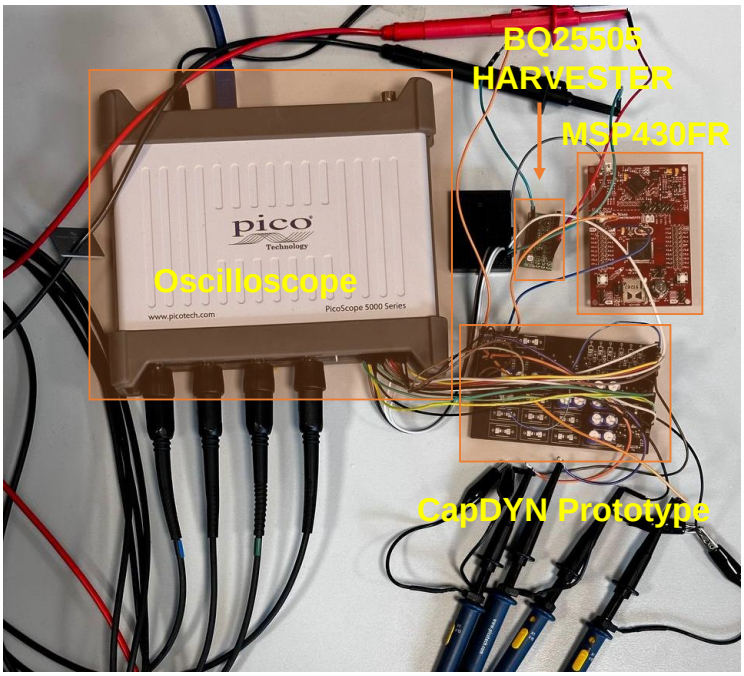


Fig. 5. Testbed for *CapDYN*'s evaluation. The D1 prototype bank shown here can be charged through a BQ25505 DC/DC converter and loaded by an MSP430FR evaluation board running the *CapDYN* runtime.

4.2 LTspice simulator

We fully simulate *CapDYN* in LTspice, with the intention of providing a hardware-software co-design tool that minimizes the number of prototype iterations to reach the best fit between a certain task set and *CapDYN*. With this simulator, we can easily include both the profiled average values and execution times for the task set we aim to supply with *CapDYN*, as well as inject real-world power input traces coming from a target energy harvesting source. This allows the user to fine-tune the *CapDYN* parameters for the task at hand, and test overall system behaviour under particular power input conditions, without having to build a prototype straight away. For maximum simulation accuracy, we use all the available supplier-provided LTspice models of the selected components. We favoured transistor-level models to get an accurate estimation of the component's power consumption, only considering behavioural models if the transistor-level models were not available. To integrate the behavioural models, we sourced their datasheet power consumption from the relevant supply line. Only in the case of the Vishay load switches, the SIP32431, no model was available: we developed a model that takes into account their reverse leakage and channel resistance effects. To showcase *CapDYN*'s adaptability to typical ultra-low power energy harvesting scenarios, we consider two different power traces taken from [18] as harvesting power inputs, one slow-varying and one with greater dynamics. We then compare the number of successful task executions with respect to a single capacitor system, to better show the advantages of dynamic capacitance over a static one.

4.3 SW Implementation

We implemented the *CapDYN* runtime as a lightweight C language library to enable task-based applications [6, 21, 31] and manage CapyDYN hardware through the FSM presented in Figure 4. Our

```c
#include "capdyn.h"
//Task declaration
Task task_t1() {
    ...
    transition_to(task_t2);
}
Task task_t2() {
    ...
    transition_to(task_t3);
}
Task task_t3() {
    ...
    transition_to(task_t4);
}
Task task_t4() {
    ...
    transition_to(task_t1);
}
...
//Defining Required Tasks' Stages
set_stage(task_t1, Stage2);
set_stage(task_t2, Stage3);
set_stage(task_t3, Stage1);
set_stage(task_t4, Stage4);
...
Entry_Task(task_t1);
```

(a) Example application code

```c
#include "capdyn.h"
...
int main(){
    ...
    if (!system_initialized){
        curr_task = initial_task;
        system_initialized = SET;
    }
    transition_to(curr_task);
}
void transition_to(task_t *next_task)
{
    uint8_t required_stage = stage_map[next_task->idx];
    next_task_setup(required_stage);
    int wake_up_stage = gpio_get_value(WKP_PIN);
    if (wake_up_stage == 1){
        exec_task(next_task);
    }
    else if (wake_up_stage == 0){
        curr_task = next_task;
        gpio_toggle(EOT_PIN);
    }


    ...
}
```

(b) Task initialization and transition of *CapDYN* runtime

Fig. 6. An example of application code using the *CapDYN* runtime is provided, along with a code snippet that illustrates how system initialization and task transitions are managed within the *CapDYN* runtime.

target platform is an MSP430FR [28] series microcontroller, which is widely used in intermittent systems and features an internal FRAM. Furthermore, the *CapDYN* runtime can be easily adapted to work on any platform that has non-volatile memory such as FRAM or MRAM. The *CapDYN* runtime executes each task by considering the task's stage requirement, in order to avoid power failures during the task execution. Offline profiling is needed to map each task to a *CapDYN* stage, as explained in detail in Section 4.4. Here, we discuss the implementation of flagship functions of the software library, for an N-stage *CapDYN*. Figure 6 shows the code snippet of an example application and the main entrance and task-to-task transition handling of the *CapDYN* runtime. Figure 6(a) presents an example application code that includes four tasks. The *CapDYN* runtime uses three main C macros to build an application: `Task`, `set_stage`, and `Entry_Task`. The `Task` macro is used to define tasks. Each task is a void function and has a corresponding data structure of type `task_t`, which holds non-volatile task control variables, such as the task's address and ID. The transition to the next task is managed by the `transition_to` macro at the end of each task. The `set_stage` macro sets the task's required stage within the `stage_map`. Finally, the `Entry_Task` macro declares the entry task (`initial_task` in Figure 6(b)) during system initialization.

Figure 6(b) presents the code snippet of the main entrance and the function responsible for the task-to-task transition (`transition_to`), which is invoked upon each wake up and after each task execution, to implement the set/check/execute loop of Figure 4. When the *CapDYN* hardware

triggers the load for the first time with the default wake-up stage, which is user configurable, the *CapDYN* runtime initializes the first task in the lines 5 to 8 and calls the `transition_to` function. The `transition_to` function is responsible for controlling the stage requirement of the task and the current stage of the *CapDYN* hardware: it either executes the task if the current stage is sufficient, or it cuts the node's power if the current stage is not sufficient. In more detail, this function obtains the required stage of the task utilizing the `stage_map` array and the task ID at line 13. Then it writes the required stage to the wake-up register in line 14. *CapDYN*'s `WKP_PIN` informs the runtime if the required wake-up stage has been achieved in the capacitor bank: thus, in line 15, `transition_to` checks the `WKP_PIN`. If the required stage has been achieved, the runtime executes the task in line 17: at the end of the task's execution, `transition_to` is newly called to check if the successive task can in run immediately or not. If the required stage has not been achieved, *CapDYN* saves the task information to the `curr_task` in line 20 and sends an `EOT` - end of task - signal to *CapDYN* hardware to cut the power of the load in line 21. Thus *CapDYN* prevents wasting power by turning off the MCU when waiting for the charging of the required stage. When the required stage is achieved, the *CapDYN* hardware newly powers the load and invokes `transition_to` to check again if the saved task in `curr_task` in line 20 can execute. Thus, *CapDYN* executes all tasks considering their energy requirements and preventing power failures during their execution.

4.4 Energy Profiling of Tasks

As explained in Section 2.1, each stage represents a certain state of charge of the bank. Each stage is also connected to the tasks it can sustain without power failure. This task-to-stage mapping is obtained by first profiling the power consumption of the task set, and then designing a *CapDYN* instance to best match each task's measured energy consumption. Offline profiling is a fundamental step, as we design *CapDYN* to perfectly sustain a certain task set. The more a stage matches the energy demands of a certain task, the more *CapDYN* will reduce charging times and improve reactivity. The profiling must be executed by considering both the run-to-run energy consumption variability and the variable voltage that *CapDYN* supplies to the load, in order to avoid power failures even in worst-case scenarios. To account for variable voltage during profiling, we fix the load voltage at the load's upper voltage threshold - which is where the loads consumes more power. The load's upper voltage threshold is set by *CapDYN*'s upper scaling threshold, V_{SH} (Section 3.1.1). Then, to account for run-to-run variability, we collect multiple (e.g., enough to have a statistical sample) energy consumption measurements, ultimately considering the largest consumption linked to a successful task execution as reference value for the dimensioning of *CapDYN*. This will slightly overprovision *CapDYN* with respect to the average run, but it will ensure faultless execution for higher-consuming runs. On the other hand, designers can analyze the energy consumption of each task by using software tools such as the Energy Trace Tool of Code Composer Studio. Additionally, they can employ energy profiling tools like Echo [14] for more accurate energy measurements. Once the energy consumption profile of each task is obtained, it can be matched to the energy level of the *CapDYN* capacitor bank's stages.

5 Evaluation

We separate the evaluation into three branches. Firstly, we analyze the efficiency and reactivity of two different capacitor bank designs that we can match with *CapDYN*, using LTspice simulations, and we compare the control of these two designs in literature (Section 5.1). Secondly, we simulate the deployment of a profiled task set powered by *CapDYN* using the two different bank designs in two real energy harvesting scenarios (Section 5.2), one characterized by quasi-static power availability and one characterized by a more dynamic power availability. These scenarios allow us to demonstrate *CapDYN*'s reactivity in comparison with a fixed capacitor, which executes its

Table 2. Switching efficiency of the two capacitor bank designs (ratio between the stored energy before and after each transition)

Switching efficiency [%]	Switching direction	stage 1/stage 2	stage 2/ stage 3	stage 3/ stage 4
Design 1	UP	99.99	96.20	94.02
	DOWN	99.99	99.99	99.99
Design 2	UP	99.97	99.98	99.99
	DOWN	99.98	99.99	99.99

All transitions are 99.9% efficient, except the stage 2 to 3 and stage 3 to 4 transitions, which respectively lose 3.80% and 5.98% of energy to small equalization transients.

tasks with no energy awareness. The paramount figure of merit is task execution number: the system that executes more tasks, given the same power input conditions and same task set, is the more successful at enabling battery-free execution. In these first two sections, both the evaluated banks feature 1 mF of unit capacitance: D1 follows Equation (1) to dimension its bank capacitors from the unit capacitance value, while D2's capacitors are all equal to 1 mF. Thirdly, we present *CapDYN* D1 bank's first hardware implementation. We used a 0.47 F supercapacitor (Kyocera AVX, SCMR14D474PRBB0) in parallel with a 220 µF tantalum capacitor (Vishay, TH3E227K010C0500) to obtain a unit capacitance of C_1=0.47022 F Equation (1). We evaluate that the progressive scaling logic operates as expected and features a switching efficiency close to the one obtained from the simulations. We then measure the prototype's power consumption and verify that it is closely estimated by the simulator. Lastly, we propose a methodology for using the simulator to optimize *CapDYN* prototype development, given certain power input and output profiles (Section 5.3).

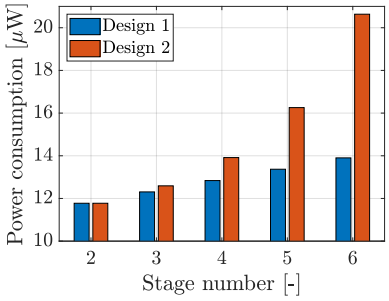
5.1 A comparison: two different capacitor bank designs and different ways to control them

Leveraging *CapDYN*'s adaptability to different capacitor banks, we have evaluated the two bank designs presented in Section 3.2 in terms of switching efficiency and power consumption. A lower infrastructural power consumption increases the system's overall efficiency and reduces the delays due to backup charging, while a high switching efficiency ensures that energy is not wasted in undesired charge transfers.

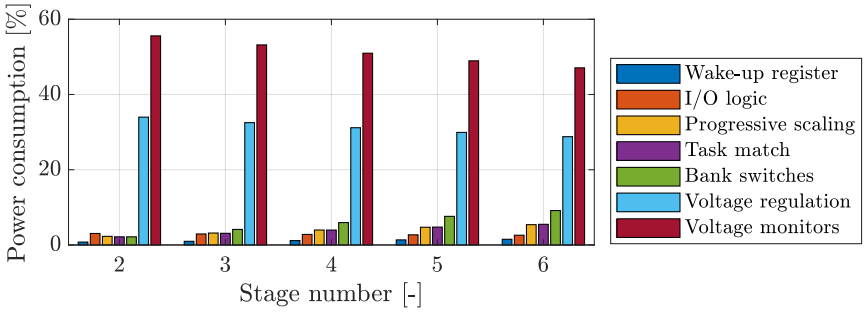
The switching efficiency results can be seen in Table 2. Efficiency is computed upon a full charge and discharge of both designs, in both switching directions (scale-up and down). It is computed as the ratio of the stored energy before and after the stage switch. The D2 bank proves to be the most efficient, as the worst measured switching efficiency is 99.97% (stage 1 to 2). On the other hand, all of D1's transitions retain 99.99% of energy, achieving figures similar to D2, except in two cases: the transitions from stage 2 to 3 and from stage 3 to 4 have a slightly lower switching efficiency, measured at 96.20% and 94.02% respectively. Hardware evaluation of the switching efficiency yielded comparable results (Section 5.3): the slightly lower efficiency of these two transitions is due to small equalization transients happening between parallelized capacitors, post switch (Figure 11). Other capacitor banks with an equal design suffer from the same shortcoming [27].

We evaluated the power consumption of the two designs when increasing their capacitance stages through the LTspice model introduced in Section 4.2. While D2 edges out D1 in terms of switching efficiency, D1 features a smaller, linearly growing power consumption, while D2's is always slightly larger and exponentially growing (Figure 7(a)). The culprit of the exponential growth are the bank's switches, which duplicate with each additional stage, as it can be seen from Figure 7(c).

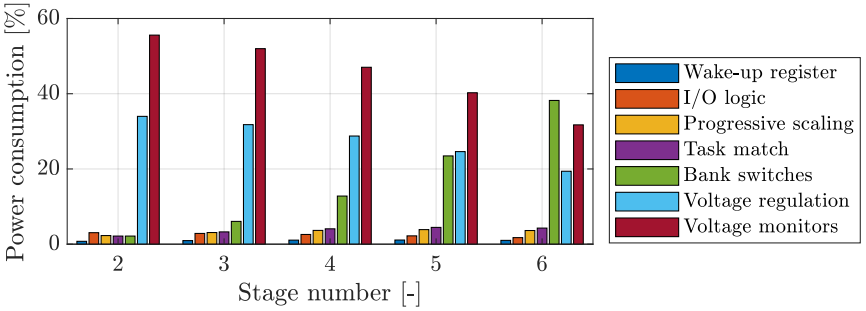
These two designs are in trade off and no clear winner emerges: D1's power consumption is lower and scales more gracefully than D2's with stage number increase. On the other hand, D1



(a) D1 and D2's overall infrastructural power consumption, as a function of *CapDYN*'s total stage number. It can be seen that D2's consumption grows exponentially, while D1's grows linearly with stage number, proving D1 is more easily up-scaled.



(b) D1 power consumption breakdown across *CapDYN* component groups.



(c) D2 power consumption breakdown across *CapDYN* component groups. The culprit of the overall exponential power increase of D2 can be identified in the bank's switches consumption.

Fig. 7. *CapDYN*'s overall (a) and per-component group (b), (c) infrastructural consumption for two different capacitor bank designs, as a function of total bank stage number.

features slight equalization transients in some of its transitions, while D2 does not. This prompted us to further evaluate the banks in a real world deployment simulation (Section 5.2) and to measure the switching efficiency of D1 in hardware (Section 5.3).

5.1.1 Software and hardware controlled switching. In order to estimate what strategy is more efficient at controlling the previously introduced capacitor banks, we evaluate the benefits of *CapDYN*'s autonomous capacitance scaling and I/O control with respect to a software-based *CapDYN* version. In the software-based version, we replace the I/O power flow (Section 3.3) and

capacitance scaling (Section 3.2) section with an MSP430FR5994. The autonomous control section introduces an overhead of 1.036 µW, while the equivalent software-based control adds 2.38 µW and 3.65 µW of additional overhead, considering a monitoring frequency of 1 Hz and 50 Hz, respectively. The overhead introduced by the microcontroller is computed considering the MCU's low power mode power consumption, ADC measurements cost, and active mode cost during the stage monitoring. The results indicate that the autonomous *CapDYN* consumes less power while providing a higher reactivity compared to the software-based *CapDYN*. Although the overhead of the MCU-based model could be reduced by relaxing the monitoring frequency, which can be a solution for larger storages/higher-power scenarios, this reduces the system's reactivity to its active periods, which is problematic for smaller storages/lower-power scenarios, where more frequent charge monitoring is needed. *CapDYN* solves this reactivity-efficiency tradeoff, as it is always instantly reactive, with a fixed ultra-low power overhead, extending the application of dynamic capacitance to ultra-low power energy sources.

A noteworthy software-control solution is presented by Morphy, which can be identified as *CapDYN*'s closest work, as it shares the progressive capacitance scaling approach with *CapDYN*. To further motivate our autonomous control choice, we compare *CapDYN*'s constant overhead to Morphy. Morphy's average power consumption increases as the capacitor size scales down, due to a faster monitoring frequency. Indeed, a 4-stage Morphy with C = 100 mF, 10 mF, and 1 mF capacitors consumes 2.44 µW, 6.44 µW, and 46,99 µW, with 5, 0.5, and 0.05 sec. monitoring periods, respectively. *CapDYN*'s 4-stage implementation consumes 12.83 µW, regardless of the capacitor size. Morphy achieves a lower power consumption for larger capacitances by reducing its monitoring period, assuming that changes are slower. This reduces its reactivity to voltage drops and input power bursts. Therefore, Morphy might not guarantee the prevention of power failures at lower stages if its duty cycle is not properly adjusted. On the contrary, *CapDYN* is always reactive, which is especially important in the case of smaller capacitors: for the 1 mF case, *CapDYN* consumes up to 73% less average power than Morphy, besides requiring no software overhead for bank control. If the 1 mF Morphy were to achieve instant reactivity as *CapDYN*, Morphy would need to keep its control MCU always-on, which would lead to a power consumption 1000 times larger than *CapDYN*.

Besides controlling its progressive scaling with an MCU, Morphy also implements a software-based pre-regulation of its output voltage. Morphy separates a set of its bank (task capacitors) to power a certain task. Said task capacitors are maintained within an user-programmed range by charge-pump boosting, where the task capacitors are briefly connected to the remainder of the bank, which sits at a higher voltage. Although this approach allows more flexibility in terms of load voltage range than *CapDYN*'s output voltage limitation to $[V_{SL}, V_{SH}]$, Morphy's core/task voltage boosting solution is entirely based on lossy charge transfers. For example, if a 1 mF bank is charged at 3.6 V and is briefly connected to a 1 mF task capacitor, which is nearing the load's lower voltage limit of 1.8 V, 10% of the total energy is lost due to resistive dissipations in the charge transfers. If we factor in the overhead of a software-based control, this effect highly reduces Morphy's efficiency. We believe charge transfers should be limited by design, so we revtdo not consider this as a viable regulation strategy.

5.2 Simulation of real world deployment in comparison to a single capacitor system

To evaluate *CapDYN*'s performance in a realistic scenario, using our LTspice model we simulated its charge through two different ultra-low power input traces, which is where we want *CapDYN* to make a difference. We then matched the D1 design to sustain a task set profiled from an in-house battery-free application and evaluated the execution number for each task, in comparison to a single capacitor storage.

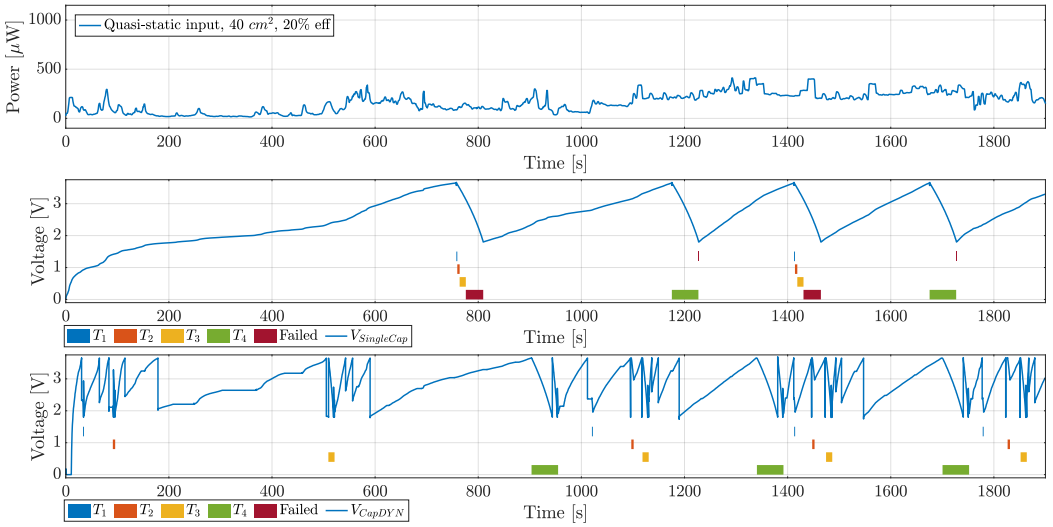


Fig. 8. Quasi-static power input (top): on average, the tasks are successfully executed 60% earlier and 28% faster when supplied with *CapDYN* fitted with the D1 bank (bottom), and not with a classical single capacitor storage (middle).

The power inputs are taken from a CRAWDAD dataset [18] containing irradiance values (incoming solar power over panel size) collected in two different scenarios. The first power input has been collected by a pedestrian walking in New York City, which we term "quasi-static" as it features a power input with lower variations. The second power input comes from a person walking indoors and outdoors, which we term "dynamic", as it features more transients. Assuming that the conversion efficiency from irradiance to electrical power is about 20% [19], and selecting a panel size of 40 cm^2 for the quasi-static case and 10 cm^2 for the dynamic case, we obtain electrical power traces which we can input to *CapDYN*'s simulator (Figure 8, top plot, Figure 9, top plot). In doing so, we implicitly avoid considering the load variation-induced effects on the harvested power trace, as they are source and converter-specific, and thus out of scope of this paper. As the behavior of *CapDYN* is connected to the power input it receives, shown in Figures 8 and 9, and not to the upstream conversion dynamics, this fact does not compromise the aforementioned experiments. We selected a lower panel size for the dynamic scenario, as otherwise the energy balance with respect to the selected application would have been too positive to really put *CapDYN* to test. Besides, to test the dimensioning of the system, we assume no incoming power as a task is executing.

The task set used in this evaluation comes from an in-house battery-free plant health monitoring node [11]. The application collects temperature, humidity, open circuit voltage (OCV) and electrochemical impedance spectroscopy (EIS) readings whenever energy is available, and transmits them through the LoRa protocol, for the purpose of correlating EIS/OCV readings with environmental data and external plant health monitoring. The application features four tasks: OCV measurement and EIS parameter processing (T1), temperature and humidity sensing (T2), LoRa data transmission (T3) and a full frequency EIS sweep (T4). The profiled energy from the tasks is shown in Table 4.

We dimensioned 4-stage *CapDYN* D1 and D2 bank to accommodate T4 when fully charged, and used a 90 µF capacitor as backup. As it can be seen from Table 4, each of D1's stages can supply at least one of the application tasks, while that cannot be said for D2, as its first stage is too small to support even the smallest application task. We thus use the D1 bank for this evaluation, as it better fits the considered application. We compare the task execution rate to the one we would

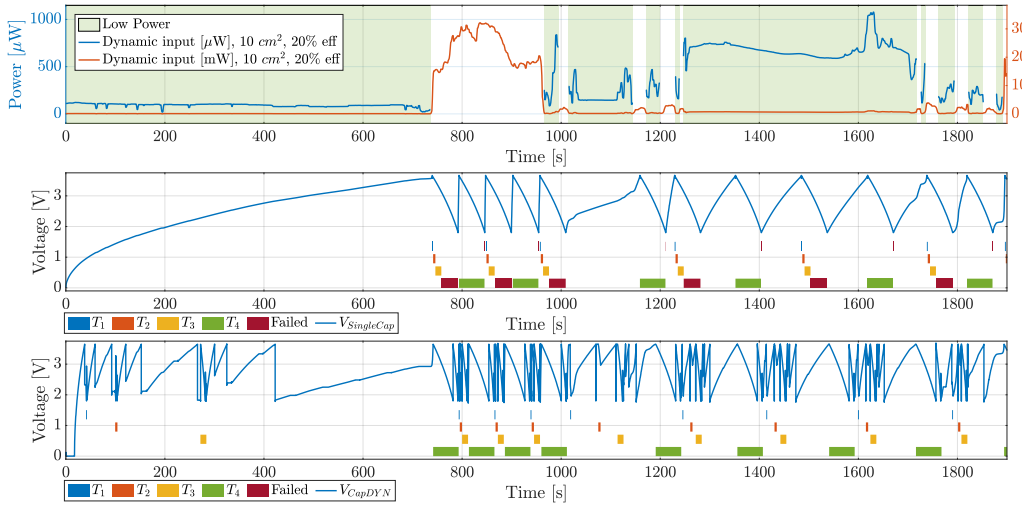


Fig. 9. Dynamic power input (top), plotted on two different scales, µW (left) and mW (right), to appreciate the lower power input sections (green background): on average, the tasks are successfully executed 63% earlier and 38% more frequently when supplied with *CapDYN* fitted with the D1 bank (bottom), and not with a classical single capacitor storage (middle).

Table 3. Number of Successful Task Executions when our in-House Application is Powered by Either a Single Capacitor or a D1 4-stage *CapDYN*

Successful exec #	**T1**	**T2**	**T3**	**T4**
***CapDYN* D1**	7	7	7	6
Single cap	5	5	5	5

(a) Quasi-static input evaluation results

Successful exec #	**T1**	**T2**	**T3**	**T4**
***CapDYN* D1**	18	18	18	18
Single cap	13	13	13	13

(b) Dynamic input evaluation results

We evaluate the two systems by supplying them as input two real-world power inputs (quasi-static and dynamic), under the worst-case assumption that as each task is executed, a power failure happens.

achieve by using a fixed capacitor dimensioned to supply T4 as storage. In the single capacitor simulation, tasks are executed with no energy awareness: we use a voltage monitor to control a flip flop and a switch, to activate the load when its upper voltage limit is reached, and deactivate it at the load's lower voltage limit. This infrastructure adds an overhead of 720 nA if we consider the same components as *CapDYN*. Besides, the single capacitor features a larger capacitance with respect to the fourth stage of D1 (10.35 mF vs 8 mF), as the single capacitor cannot benefit from the improved energy extraction of *CapDYN*.

The first 1900 out of 3600 seconds of the quasi-static and dynamic power inputs (top plot) and the relative *CapDYN* (bottom plot) and single capacitor (middle plot) voltages and execution rates are pictured in Figures 8 and 9, respectively. Successful task execution results are reported in Table 3. In the quasi static scenario (Table 3(a)), *CapDYN* executes all tasks seven times (aside T4, executed six times), while the single capacitor manages to execute them successfully only five times, and fails to execute T1 and T4 four and five times respectively, when they are launched when insufficient energy is available in the storage. In the dynamic scenario (Table 3(b)), *CapDYN* executes each task eighteen times, where the single cap only manages thirteen. *CapDYN* improves task execution rate of 28% and 38% with respect to a single capacitor system, in the static and dynamic energy harvesting scenarios respectively.

Table 4. Task-to-stage Mapping used to Evaluate the Task Execution Rate of an in-House Battery-Free Application when Supplied from *CapDYN* and from a Single Capacitor

Task/stage #	1	2	3	4
Task energy [mJ]	1.8	4.5	12.0	51.5
D1 stage energy [mJ]	1.96	5.12	13.12	53.08
D2 stage energy [mJ]	0.63	3.13	13.12	53.08

D1's stages are large enough to supply at least one of the tasks, while D2's stage 1 is too small to do so.

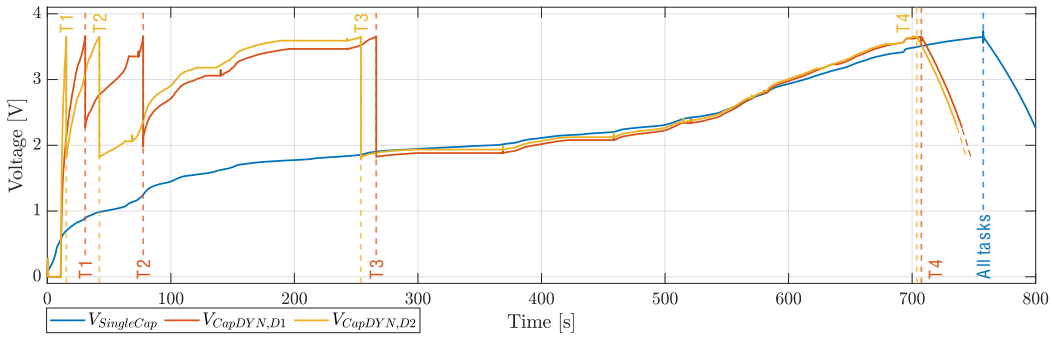


Fig. 10. Cold start of three storages charged with the quasi-static power trace: *CapDYN* with bank D1, *CapDYN* with bank D2 and a single capacitor. Vertical dashed lines indicate when enough energy has been stored to execute the task specified by the label. Both *CapDYN* D1 and D2 can schedule all tasks earlier with respect to the single capacitor.

To further demonstrate *CapDYN*'s importance in enabling batteryless execution, we rely on the quasi-static dataset to estimate the *reactivity* of D1 and D2 in comparison to that of a single capacitor, where all evaluated storages are dimensioned to supply T4. Reactivity is measured as the time required for the storage to launch a task, and so, to reach a certain stage, when initially empty. As shown in Figure 10, both the D1 and D2 *CapDYN* banks both improve reactivity with respect to the single capacitor, as they can schedule all tasks earlier. In fact, D1 can schedule its stage-matched tasks 95.94%, 89.76%, 64.86%, 6.61% earlier with respect to when the single capacitor achieves operative voltage, while D2 improves its task's execution times by 97.97%, 94.43%, 66.49%, 7.11% with respect to the single capacitor. The single capacitor is significantly slower as it requires to charge a capacitor which is larger than *CapDYN*'s fourth stage, in order to supply T4. Surprisingly, even though D2 has a slightly higher power consumption, it can schedule all tasks before D1: while this is to be expected for stage 1 and stage 2 of D2, as they have a lower overall capacitance than D1's stage 1 and stage 2, the fact that D2 achieves stage 3 and 4 earlier than D1 is due to D2's more efficient switching, which for this instance of *CapDYN* impacts reactivity more than D2's higher power consumption.

5.3 The first *CapDYN* prototype: hardware measurements and co-design guidelines

We built the first prototype of *CapDYN* D1 bank, shown in Figure 1 in order to observe the behaviours reported in Section 5.1. The prototype includes the bank switches and the progressive scaling logic.

We set the operating voltage range (which also correspond to the switching thresholds, as per Section 3.2) to $[1.8\text{V}, 3.6\text{V}]$ considering TI MSP430FR series, de facto microcontrollers for

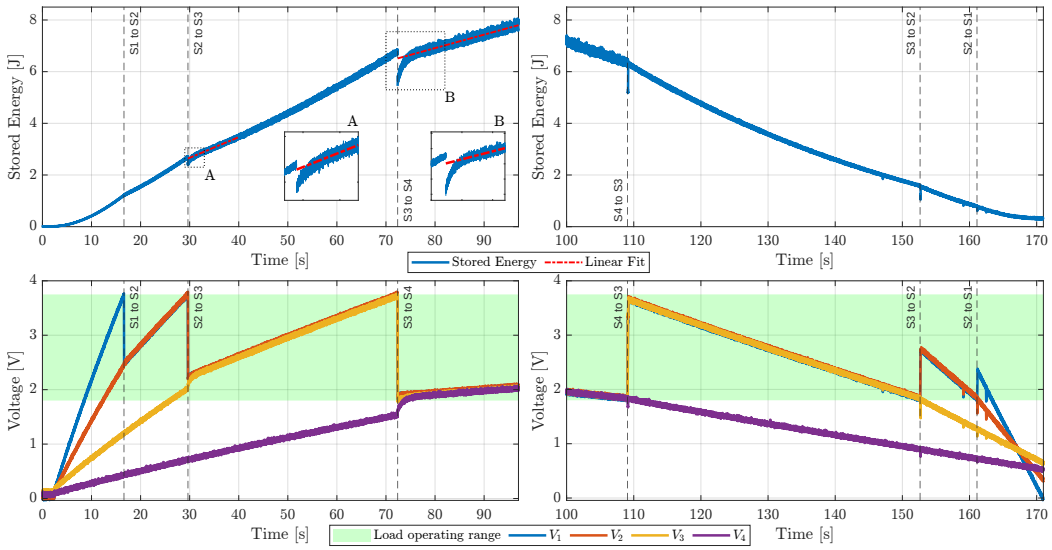


Fig. 11. Hardware measurements of the charging/discharging of 4-stage *CapDYN* D1 prototype. In the top plot, we can see the bank's stored energy: the stage transitions do not show energy drops, besides S2 to S3 and S3 to S4, which feature slight equalization transients (insets A and B). The red line is the linear fit used for computing post switch energy (Section 5.3). In the bottom plot, V_1 represents the bank's output voltage, while V_2, V_3, V_4 correspond to the voltages of the high side of the capacitors with respect to ground. We can note the slight equalization transients that characterize D1 in the transitions S2 to S3 and S3 to S4, whose effect on stored energy is highlighted by the insets A and B, top plot.

the intermittent systems. Since $V_H = 2V_L$, all D1 stages can be exploited without incurring in brownout (Section 3.2). A safety 0.05V margin was added at the upper threshold to allow the switching mechanism not to enter the *loop condition* due to the small equalization transients which characterize bank D1 after some stage transitions.

We first evaluate the charging and discharging behavior of *CapDYN*'s hardware implementation and verify the correct operation of its automatic switching mechanism. The components were powered by another SMU Keithley 2450, acting as a charged backup capacitor. We connect *CapDYN*'s main bank input to an SMU Keithley 2450 used as a 50 mA constant current source/load. This value was chosen as a reference, as most low-power loads draw similar currents. We charge the bank until it reaches full charge (stage 4, 3.65 V), then we discharge the bank until it is empty (stage 1, 0 V).

The bottom plot of Figure 11 shows the charging and discharging voltage waveforms: V_1 is the main bank's voltage, while V_2, V_3, V_4 correspond to the voltage at the top side of capacitors C_1, C_2, C_3 with respect to ground (Figure 2(a)). The *CapDYN* D1 bank starts the charging operation in stage 1, where all the capacitors are connected in series, at the bank's lowest capacitance level (Figure 2(a)). The upper voltage threshold is quickly achieved: this triggers the switch to stage 2, where C_1 and C_2 are connected in parallel. Upon switching to stage 2, the load node can be activated,and can potentially schedule tasks mapped to the first stage. As the charging process continues, capacitor bank voltage V_1 newly reaches the upper threshold, so it triggers *CapDYN*'s transition towards stage 3, where C_3 is connected in parallel to C_1, C_2. At this transition, we verify the presence of the equalization transient, confirming the LTspice evaluation results of Section 5.1: V_3 is slightly pulled up to $V_1, V2$. The circuit then proceeds in charging until stage 4 is reached. When transitioning from stage 3 to 4, connecting C_4 in parallel to C_1, C_2, C_3, we observe the second equalization transient: the fourth capacitor voltage V_4 is pulled up by a 200 mV step, to reach V_1, V_2, V_3. Finally, we stopped

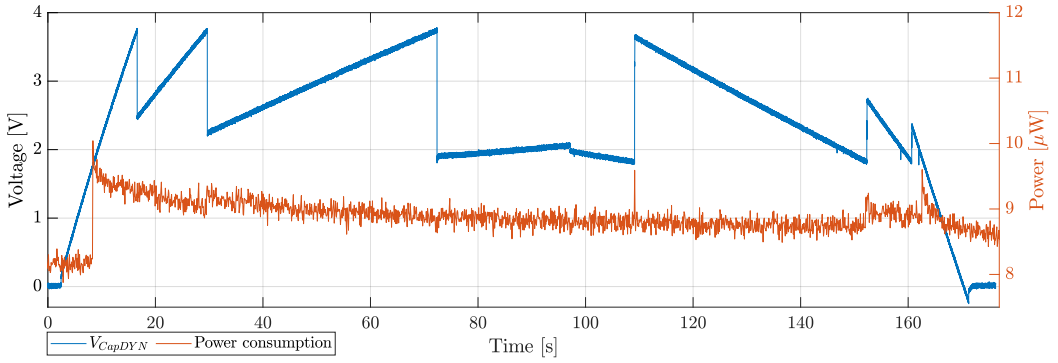


Fig. 12. Hardware measurement of *CapDYN*'s components' power consumption during charging/discharging of 4-stage *CapDYN*.

sourcing current from the SMU and started sinking 50 mA, discharging *CapDYN*. *CapDYN* gradually moves towards the lower stages as the capacitor bank is discharged. When the voltage level of the current stage drops under the low threshold voltage of 1.8 V, *CapDYN* switches the capacitor bank to a lower stage. After each down-scale switch, the transition gives way to a voltage boost through capacitor rearrangement in a more serial configuration. This boosting mechanism allows *CapDYN* to reduce its leftover energy below 1.8 V compared to a single capacitance tailored to the same energy consumption. This does improve *CapDYN*'s energy efficiency with respect to a single capacitor tailored to the same task, and also its cold start speed, as *CapDYN* requires a smaller capacitance to supply the same task. This can be clearly appreciated in Section 5.2, where *CapDYN* can sustain T4 with 8 mF, while the single capacitor requires 10.35 mF to sustain the same task. Additionally, *CapDYN* maintains the load voltage's operating range (green area, Figure 11) during all switching operations.

Focusing now on the top plot of Figure 11, showcasing the energy stored within the bank throughout charging and discharging, we estimate the switching efficiency of each stage transition, as we did for the simulation (Section 5.1). The stage transitions are faultless, as they cause no variation of the stored energy: only the transitions from stage 2 to stage 3 and from stage 3 to stage 4 feature a near-faultless switching efficiency of 97.89% and 96.27%, respectively. The small losses in the S2 to S3 and S3 to S4 transitions are due to voltage equalization transients, which dissipate energy in charge transfer among capacitors. To estimate the post-switch energy of these transient-affected transitions, the transient must be over: we execute a linear fit to avoid considering both the energy stored during the transient and the transient effects themselves (inset A, B, Figure 11). The fitting was not necessary in the simulation, as the transient is shorter due to the smaller capacitance and series resistance of the bank capacitors in the simulated CapDYN (Section 5.1). We conclude that the hardware switching efficiency measurements largely confirm the simulated switching efficiency results of Table 2.

Ultimately, we characterize the power consumption of the bank and the progressive scaling logic. We recorded the infrastructural power consumption values at the backup capacitor input through another Keithley 2450 SMU. As shown in Figure 12, *CapDYN* has an almost constant power consumption of 8 µW. The slight consumption increase of about 1 µW at around 10 s can be linked to the variation of logic levels due to the voltage monitor lower threshold being overcome. As the capacitor bank is fully discharged and the logic returns to its initial stage, this power increase is reverted. The power consumption evaluation obtained with the simulator is on the safe side, as it slightly overestimates the bank's consumption, rating it to 12.7 µW.

As the LTspice model correctly represents both the *CapDYN*'s hardware prototype's power consumption and switching efficiency, we believe it is a fundamental tool for hardware-software co-design. In fact, once a desired task set has been profiled, and a first *CapDYN* prototype has been developed, we can further improve the efficiency of the simulator by further matching it to the developed hardware. For example, we could include each capacitor's RLC models. This could be useful to test how the prototype behaves with different energy harvesters, so different current inputs.

6 Discussion

In this section, we discuss the limits of *CapDYN* and future work.

Hardware implementation: The relationship between capacitors to make the D1 bank design more efficient is to be further studied. The D1 bank is a promising solution, as D1 is easier to scale up in stage number with respect to D2, and D1 can be adapted to loads with tighter voltage ranges. The development of a new capacitor dimensioning solution could fully remove D1's small equalization transients.

Besides optimizing the capacitor bank, we can improve overall energy efficiency and cold start times by reducing infrastructural power consumption, thus shrinking the required size of the backup capacitor. To begin with, charge-pump like voltage boosting could be more efficient than including an *harvester-to-DC* conditioning stage, in the case of low-voltage DC sources. On the other hand, charge-pump boosting is only efficient if no voltage equalization transients appear, and harvester-to-DC feature integrated maximum power point tracking algorithms, which can make up for the additional power expenditure by harvesting more energy from the source. Furthermore, a VLSI implementation has the potential to significantly reduce power requirements and the overall area occupied by the electronic circuitry. This advancement would expand the range of applications to even lower power and more compact scenarios, further reducing *CapDYN*'s power consumption. This would expand the lowest current value we can harvest, without sacrificing reactivity. For the sake of modularity and design freedom, we could also integrate all the components needed to expand one stage into one single chip.

A large number of stages could also have interesting applications, as it could provide both fine-grained capacitance control and pre-regulation. This would allow designers to reduce charging overheads by perfectly matching requested task energy to the energy available at a certain stage. The tradeoff between the benefits offered by an increased capacitance granularity and higher consumption is to be investigated. The effect of the capacitor bank's series resistance on the circuit's operation is also to be fully understood.

Energy and Task-based Intermittent Computing: *CapDYN* employs the task-based intermittent programming model, which fits best with the staged capacitor bank. The proper mapping of the tasks' energy consumption to the stages' energy availability, fully eliminates the power-failure interrupts during task execution. Power outages can only happen between tasks while waiting for charging, which introduces neither a non-termination issue nor memory inconsistency. Moreover, depending on the application, *CapDYN*'s energy state interface allows for scheduling the flow and order of the tasks according to the available energy in the current stage of the capacitor bank. For instance, a task with a high priority can be executed earlier than other minor tasks. In this sense, also time-constrained tasks can take a chance to run intermittently. Besides, we could also develop a scheduler that prioritizes certain tasks depending on the power input level by measuring the charging times at the load side through persistent timekeepers [8], or within *CapDYN* itself. We leave the implementation of such a scheduler for future work.

***CapDYN* with Checkpoints:** Although the task-based model is more suitable for *CapDYN* due to its energy-awareness, checkpointing approaches can also be efficiently utilized as an intermittent

runtime. For static checkpointing, the programmer must consider the first-stage energy availability of *CapDYN* to determine the placement of checkpoints. In this way, *CapDYN* can ensure forward processing by completing at least one checkpoint with the worst-case energy availability. Moreover, dynamic checkpointing can be implemented by utilizing the stage information coming from *CapDYN* hardware. If the current stage has enough energy to complete the next checkpoint, the current checkpoint operation can be omitted at runtime to optimize energy consumption.

7 Conclusions

This paper introduces *CapDYN*, an innovative energy storage architecture that enables fully autonomous reconfiguration of its capacitor bank. We leverage readily available COTS components to constantly monitor the bank voltage and accordingly rearrange its capacitors in different series/-parallel configurations, achieving different levels of equivalent bank capacitance.

By eliminating the need for MCU intervention, we solve the trade-off between energy efficiency and reactivity to bank voltage changes, expanding the harvesting scenarios to highly unpredictable sources and environments. *CapDYN* consumes only 11.6 µW in its 2-stage implementation, while its 4-stage version consumes 12.83 µW, improving the previous state of the art by 73%. Comparing our storage with a fixed capacitor solution, we reduce time to operational voltage of up to 98% and can schedule tasks up to 38% faster.

Besides this, the capacitor array promptly provides the application layer with an indication of the stored energy. A dedicated library has been integrated to incorporate this information, into the firmware running on the *CapDYN*'s load. Our experiment results outperform those reported in the most recent articles and clearly demonstrate the benefits of autonomous control, particularly when dealing with low input power rates. This achievement highlights the importance and advantages of implementing *CapDYN* in almost-zero-energy battery-less IoT devices.

References

[1] Rehan Ahmed, Bernhard Buchli, Stefan Draskovic, Lukas Sigrist, Pratyush Kumar, and Lothar Thiele. 2019. Optimal power management with guaranteed minimum energy utilization for solar energy harvesting systems. *ACM Trans. Embed. Comput. Syst.* 18, 4, Article 30 (jun 2019), 26 pages. https://doi.org/10.1145/3317679

[2] Saad Ahmed, Bashima Islam, Kasim Sinan Yildirim, Marco Zimmerling, Przemysław Pawełczak, Muhammad Hamad Alizai, Brandon Lucia, Luca Mottola, Jacob Sorber, and Josiah Hester. 2024. The internet of batteryless things. *Commun. ACM* 67, 3 (2024), 64–73.

[3] Arwa Alsubhi, Simeon Babatunde, Nicole Tobias, and Jacob Sorber. 2024. Stash: Flexible Energy Storage for Intermittent Sensors. *ACM Trans. Embed. Comput. Syst.* (jan 2024). https://doi.org/10.1145/3641511 Just Accepted.

[4] Abu Bakar, Rishabh Goel, Jasper de Winkel, Jason Huang, Saad Ahmed, Bashima Islam, Przemysław Pawełczak, Kasım Sinan Yıldırım, and Josiah Hester. 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*. 207–221.

[5] Jongouk Choi, Hyunwoo Joe, and Changhee Jung. 2022. CapOS: Capacitor error resilience for energy harvesting systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41, 11 (2022), 4539–4550.

[6] Alexei Colin and Brandon Lucia. 2016. Chain: Tasks and channels for reliable intermittent programs. In *Proc. OOPSLA* (Oct. 30 – Nov. 4). ACM, Amsterdam, Netherlands, 514–530.

[7] Alexei Colin, Emily Ruppel, and Brandon Lucia. 2018. A reconfigurable energy storage architecture for energy-harvesting devices. In *Proc. of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*. 767–781.

[8] x Jasper de Winkel, Carlo Delle Donne, Kasim Sinan Yildirim, Przemysław Pawełczak, and Josiah Hester. 2020. Reliable Timekeeping for Intermittent Computing. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'20)*. Association for Computing Machinery, New York, NY, USA, 53–67. https://doi.org/10.1145/3373376.3378464

[9] Jasper de Winkel, Vito Kortbeek, Josiah Hester, and Przemyslaw Pawelczak. 2020. Battery-free Game Boy. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 111 (2020), 1–26.

[10] Harsh Desai, Matteo Nardello, Davide Brunelli, and Brandon Lucia. 2022. Camaroptera: A Long-range Image Sensor with Local Inference for Remote Sensing Applications. *ACM Trans. Embed. Comput. Syst.* 21, 3, Article 32 (May 2022), 25 pages. https://doi.org/10.1145/3510850

[11] Maria Doglioni, Matteo Nardello, and Davide Brunelli. 2024. Plant Microbial Fuel Cells: Energy Sources and Biosensors for battery-Free Smart Agriculture. *IEEE Transactions on AgriFood Electronics* 2, 2 (2024), 460–470. https://doi.org/10.1109/TAFE.2024.3417644

[12] Firdaous El Mahboubi, Marise Bafleur, Vincent Boitier, and Jean-Marie Dilhac. 2018. Energy-Harvesting Powered Variable Storage Topology for Battery-Free Wireless Sensors. *Technologies* 6, 4 (2018).

[13] Vasiliki Gogolou, Stylianos Siskos, and Theodore Laopoulos. 2019. A reconfigurable storage unit for autonomously powered IoT nodes. In *2019 Panhellenic Conference on Electronics & Telecommunications (PACET)*. 1–4.

[14] Josiah Hester, Timothy Scott, and Jacob Sorber. 2014. Ekho: Realistic and repeatable experimentation for tiny energy-harvesting sensors. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*. 330–331.

[15] Josiah Hester, Lanny Sitanayah, and Jacob Sorber. 2015. Tragedy of the coulombs: Federating energy storage for tiny, intermittently-powered sensors. In *Proc. of the 13th ACM Conference on Embedded Networked Sensor Systems*. 5–16.

[16] Josiah Hester and Jacob Sorber. 2017. Flicker: Rapid prototyping for the batteryless internet-of-things. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. 1–13.

[17] Josiah Hester and Jacob Sorber. 2017. The future of sensing is batteryless, intermittent, and awesome. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. 1–6.

[18] David Kotz, Tristan Henderson, Ilya Abyzov, and Jihwang Yeo. 2009. CRAWDAD dataset dartmouth/campus (v. 2009-09-09). Available for download on IEEE DataPort. (September 2009). https://doi.org/10.15783/C7F59T

[19] Sławomir Kurpaska, Jarosław Knaga, Hubert Latala, Jakub Sikora, and Wiesław Tomczyk. 2018. Efficiency of solar radiation conversion in photovoltaic panels. *BIO Web of Conferences* 10 (01 2018), 02014. https://doi.org/10.1051/bioconf/20181002014

[20] He Li, Kaoru Ota, and Mianxiong Dong. 2018. Energy Cooperation in Battery-Free Wireless Communications with Radio Frequency Energy Harvesting. *ACM Trans. Embed. Comput. Syst.* 17, 2, Article 44 (feb 2018), 17 pages. https://doi.org/10.1145/3141249

[21] Kiwan Maeng, Alexei Colin, and Brandon Lucia. 2017. Alpaca: Intermittent execution without checkpoints. *Proceedings of the ACM on Programming Languages* 1, OOPSLA (2017), 1–30.

[22] Kiwan Maeng and Brandon Lucia. 2020. Adaptive Low-Overhead Scheduling for Periodic and Reactive Intermittent Execution. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2020)*. Association for Computing Machinery, New York, NY, USA, 1005–1021. https://doi.org/10.1145/3385412.3385998

[23] Matteo Nardello, Luca Caronti, and Davide Brunelli. 2023. Intermittent Intelligent Camera with LEO sensor-to-satellite Connectivity. In *Proceedings of the 11th International Workshop on Energy Harvesting & Energy-Neutral Sensing Systems (ENSsys'23)*. Association for Computing Machinery, New York, NY, USA, 79–85. https://doi.org/10.1145/3628353.3628550

[24] Matteo Nardello, Harsh Desai, Davide Brunelli, and Brandon Lucia. 2019. Camaroptera: A Batteryless Long-Range Remote Visual Sensing System *(ENSsys'19)*. Association for Computing Machinery, New York, NY, USA, 8–14.

[25] Emily Ruppel, Milijana Surbatovich, Harsh Desai, Kiwan Maeng, and Brandon Lucia. 2022. An Architectural Charge Management Interface for Energy-Harvesting Systems. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 318–335.

[26] Adnan Sabovic, Ashish Kumar Sultania, Carmen Delgado, Lander De Roeck, and Jeroen Famaey. 2022. An Energy-Aware Task Scheduler for Energy-Harvesting Batteryless IoT Devices. *IEEE Internet of Things Journal* 9, 22 (2022), 23097–23114. https://doi.org/10.1109/JIOT.2022.3185321

[27] A. Siskos, F. El Mahboubi, V. Boitier, Th. Laopoulos, and M. Bafleur. 2018. A power management system using reconfigurable storage scheme for batteryless wireless sensor nodes. In *2018 7th International Conference on Modern Circuits and Systems Technologies (MOCAST)*. 1–4. https://doi.org/10.1109/MOCAST.2018.8376615

[28] Texas Instruments. 2019. MSP430FR58xx, MSP430FR59xx, MSP430FR68xx, and MSP430FR69xx Family User's Guide. http://www.ti.com/lit/ug/slau367o/slau367o.pdf. (2019). Last accessed: September 2019.

[29] Dries Van Leemput, Adnan Sabovic, Khodr Hammoud, Jeroen Famaey, Sofie Pollin, and Eli De Poorter. 2023. Energy Harvesting for Wireless IoT Use Cases: A Generic Feasibility Model and Tradeoff Study. *IEEE Internet of Things Journal* 10, 17 (2023), 15025–15043. https://doi.org/10.1109/JIOT.2023.3263543

[30] Fan Yang, Ashok Samraj Thangarajan, Sam Michiels, Wouter Joosen, and Danny Hughes. 2021. Morphy: Software defined charge storage for the IoT. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems (SenSys'21)*. Association for Computing Machinery, New York, NY, USA, 248–260. https://doi.org/10.1145/3485730.3485947

[31] Kasım Sinan Yıldırım, Amjad Yousef Majid, Dimitris Patoukas, Koen Schaper, Przemyslaw Pawelczak, and Josiah Hester. 2018. Ink: Reactive kernel for tiny batteryless sensors. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*. 41–53.

[32] Eren Yildiz, Saad Ahmed, Bashima Islam, Josiah Hester, and Kasim Sinan Yildirim. 2023. Efficient and safe I/O operations for intermittent systems. In *Proceedings of the Eighteenth European Conference on Computer Systems*. 63–78.

[33] Eren Yıldız, Lijun Chen, and Kasim Sinan Yıldırım. 2022. Immortal threads: Multithreaded event-driven intermittent computing on {Ultra-Low-Power} microcontrollers. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*. 339–355.

[34] Eren Yildiz and Kasim Sinan Yildirim. 2020. Defragmenting energy storage in batteryless sensing devices. In *Proc. of the 8th International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems*. 36–42.

Received 19 April 2024; revised 16 April 2025; accepted 8 May 2025