Energy-aware Scheduling and Input Buffer Overflow Prevention for Energy-harvesting Systems

Harsh Desai
harshd@andrew.cmu.edu
Carnegie Mellon University
Pittsburgh, USA

Xinye Wang
xinyew@andrew.cmu.edu
Carnegie Mellon University
Pittsburgh, USA

Brandon Lucia
blucia@andrew.cmu.edu
Carnegie Mellon University
Pittsburgh, USA

Abstract

Modern energy-harvesting devices [23] use on-device compute to discard data uninteresting to the application, improving energy availability. These devices capture data at fixed rate, and process captured data at a rate that varies with environmental factors like input power and event activity. If capture rate exceeds processing rate, new inputs are stored in a small on-device input buffer (hundreds of kBs). When the input buffer fills up, the device discards newer inputs, missing potentially interesting events. Energy-harvesting devices must avoid such input buffer overflows (IBO) to avoid missing interesting events. A static solution to IBOs is impossible given dynamic variations in processing rate, and prior research fails to provide a suitable dynamic solution. We propose Quetzal, a new hardware-software solution targeted at avoiding IBOs. Quetzal's software has two parts: a new energy-aware scheduler that selects jobs with the lowest end-to-end latency (including energy recharging), and a runtime which uses queueing-theory to predict if the selected job will cause IBOs. Quetzal reacts to imminent IBOs by degrading the scheduled job. Quetzal's scheduler and runtime use a simple, system-agnostic hardware circuit to measure power at runtime. Quetzal reduces events missed due to IBOs by up to 4.2× compared to several baselines.

CCS Concepts: • **Computer systems organization** → **Embedded software**; **Embedded hardware**; Sensor networks; • **Hardware** → *Sensor applications and deployments*; *Sensor devices and platforms*; • **Mathematics of computing** → **Queueing theory**.

Keywords: energy-harvesting devices; input-buffer overflows; intermittent computing; energy-aware scheduling

ASPLOS '25, Rotterdam, Netherlands

ACM ISBN 979-8-4007-1079-7/25/03
https://doi.org/10.1145/3676641.3715995

ACM Reference Format:
Harsh Desai, Xinye Wang, and Brandon Lucia. 2025. Energy-aware Scheduling and Input Buffer Overflow Prevention for Energy-harvesting Systems. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '25), March 30-April 3, 2025, Rotterdam, Netherlands.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3676641.3715995

1 Introduction

Energy-harvesting devices enable scalable, affordable and environmentally-safe deployments in applications like wildlife tracking and city surveillance. These devices replace bulky, expensive, and poisonous batteries with energy harvesters to collect clean environmental energy (e.g. solar, RF). Typical energy-harvesting devices have sensors (e.g. camera), a compute core (e.g. microcontroller) and peripherals (e.g. radio), all powered through energy harvested and stored in a small supercapacitor. Figure 1 shows an example device operation (like [23]), periodically capturing image data. While earlier devices transmitted every input, recent works [23, 31, 46] use on-device compute to identify inputs 'interesting' to the application. Such devices transmit fewer uninteresting inputs, recovering energy and wireless bandwidth for interesting inputs. The example above uses a cheap-discarding step to eliminate images that are definitely uninteresting (e.g. empty background), storing the rest in a input buffer to be processed further. The stored images are then evaluated for 'interesting'-ness using complex processing (e.g. ML); the radio transmits images deemed 'interesting'.

Problem Overview: If the device processes images slower than the storing rate, the input buffer starts to fill up. Since energy-harvesting devices have severely limited memory (few hundred KBs), the input buffer easily overflows. Once the input buffer is full, the device cannot store new data, missing events and reducing sensing effectiveness. Therefore, it is critical to prevent the input buffer from overflowing.

Challenge: Statically avoiding input buffer overflows (IBOs) is challenging since processing time varies dynamically. End-to-end processing time in energy-harvesting devices varies due to two main reasons: input power and event activity. *Input power* causes energy-recharging time to vary, in turn controlling end-to-end processing [22]. On-device processing draws energy from a supercapacitor; if the supercapacitor discharges below a threshold, the device must wait to recharge

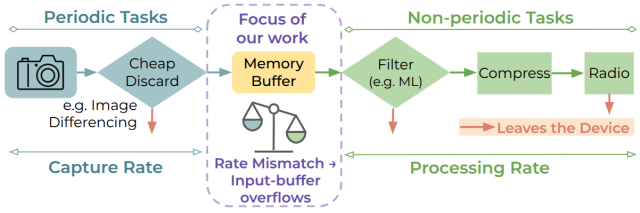


Figure 1. An example energy-harvesting system that senses periodically, discarding some inputs and buffering the rest for processing. If inputs are stored faster than they are processed, the input buffer overflows, discarding newer, potentially interesting inputs.

energy before continuing execution. Energy recharging is therefore on the critical path of end-to-end processing time, causing it to vary dynamically with input power. *Event activity* affects the contents of the captured data. High event activity (fewer inputs cheaply discarded) requires complex processing on more inputs, leading to slower processing rates. Low event activity (more inputs cheaply discarded) has lower complex processing requirements, leading to faster processing rates. Given these dynamic variations in end-to-end processing time, it is challenging to statically avoid IBOs, requiring a dynamic runtime solution.

Limitations of Prior Work: Energy-harvesting devices are attracting increasing interest from researchers, with several recent works on programming models and tools [16, 17, 39, 49, 59, 61, 73, 83, 84], architectural interfaces [18, 74] and energy-minimal architectures [28–30]. Some works propose schedulers [43, 62] that guarantee the execution of periodic events (e.g. image-capture, diff) under variable energy-harvesting conditions. However, existing works fail to identify and address the challenge that IBOs pose for periodic energy-harvesting devices. *The main goal of this work is therefore to reduce the events missed due to IBOs.*

Our approach: Quetzal is an energy-aware, queueing-theory-based software system that reduces events missed due to IBOs. Quetzal is composed of a job scheduling algorithm (Energy-aware Shortest Job First – SJF) and runtime software that detects and avoids potential IBOs for the scheduled job. Quetzal uses the Energy-aware SJF algorithm to minimize the end-to-end processing time at the given environmental conditions. Since the Energy-aware shortest job can still cause an IBO, Quetzal dynamically detects potential IBOs and uses degradation of task quality to reduce end-to-end processing time and consequently, the risk of an IBO. Quetzal uses simple hardware support to dynamically monitor harvested and execution power. To use Quetzal, a programmer first annotates *tasks* and *jobs* in their application. A task is any computation that processes a periodic input, such as machine-learning (ML) inference. Some tasks are *degradable*, which can adaptively vary their time and energy cost, for example, using differently quantized ML models. A *job* is composed as a sequence of several tasks, one of which is degradable. One job can spawn another job by inserting its input into the device's input input buffer. Quetzal schedules jobs by extending the Shortest Job First (SJF) policy to account for energy collection time. Quetzal addresses two key challenges with avoiding IBOs in energy-harvesting devices.

First, identifying the shortest job is hard. Job processing time varies with input power. With low input power, energy collection dominates; ML inference uses less energy and is thus faster than sending a radio packet. With high input power, compute time dominates and it is faster to send a packet. The environment also influences scheduling, because frequent interesting events will activate more tasks and slow processing. Quetzal addresses this challenge by dynamically tracking input power and per-task execution probability when determining end-to-end job processing time.

Second, the shortest job may still cause an IBO under low input power and/or high event activity. Quetzal uses Little's Law to predict imminent IBOs, estimating how many new inputs will be stored over the execution of the shortest job. If an IBO is imminent, Quetzal degrades the job's degradable task to decrease its time and energy requirements. Quetzal evaluates all available degradation options for a task, selecting the highest-quality option that avoids the IBO, if any. Quetzal requires multiple integer divisions, which is expensive on ultra-low-power microcontrollers like the Texas Instruments' MSP430 [86] (100s of clock cycles per division). We present a novel, simple hardware circuit that measures input and execution power on any system in a manner that eliminates integer divisions. Our circuit uses two diodes, a multiplexer and a low-power 8-bit analog-to-digital converter (ADC).

The contributions of our work, *Quetzal*, are as follows:

- A new scheduler, Energy-aware SJF, that incorporates energy-recharging time to identify jobs with shortest *end-to-end* processing times.
- A new IBO-detection and reaction engine to predict if the shortest job will cause an IBO, and to degrade tasks only as much as required to avoid the IBO.
- Quetzal's hardware-software solution that simplifies computations using execution- and input power.
- An evaluation that shows Quetzal reduces the events missed due to input buffer overflows by up to 4.2× compared against several baselines.

2 Background

The main goal of Quetzal is to reduce the events missed due to IBOs in periodic energy-harvesting devices. We provide context for this problem with an overview of a typical energy-harvesting device (2.1) and the risk presented by IBOs (2.2). We then quantitatively motivate Quetzal by showing that naive approaches to the problem are ineffective (2.3)

2.1 Primer on Energy-harvesting Sensors

Sensor devices convert physical information from their environments into digital data, enabling applications like monitoring infrastructure, public space [14, 67], and wildlife [34, 55]. Many applications require large-scale deployments, often in remote, inaccessible locations. Battery-powered devices fail to meet the needs of such applications. Batteries increase device volume, mass, and cost, require frequent replacements, and can leak hazardous chemicals into the environment. Sensors powered using harvested energy (e.g. solar, RF) are a promising alternative to battery-powered devices, enabling smaller sizes, and large-scale, long-term deployments with little maintenance. Energy-harvesting sensor devices may be "sense-and-transmit" or may process data locally. Sense-and-transmit devices are simple, and indiscriminately transmit all sensed data to a base station. Recent work [23, 31, 46] showed the benefit of on-device compute to identify and transmit data of interest to an application, discarding uninteresting data. On-device compute avoids the energy wasted transmitting uninteresting data and reduces network congestion in large deployments.

Figure 1 shows an example of on-device compute in energy-harvesting devices [23, 46], where a camera periodically captures images. The device only stores images different from the previous one, discarding unchanged images. Stored images are processed on the device using application-specific logic. The shown example uses on-device ML to identify interesting data (e.g. images of people or exotic birds) to be subsequently compressed and transmitted. A key challenge in such devices is that images arrive strictly periodically and a device needs to process frames as they arrive, regardless of harvestable energy or interesting activity in the environment. Failure to keep up with the arrival of new inputs results in input buffer overflows and loss of data. A flurry of recent work sought to enable and optimize on-device compute in energy-harvesting devices, including schedulers [43, 62], programming models and tools [16, 17, 39, 49, 59, 61, 73, 83, 84], architectural interfaces [18, 74] and energy-minimal computer architectures [28–30]. These systems make great progress toward realizing capable on-device computing for energy-harvesting systems, but none directly addresses the input buffer overflows problem that is the focus of Quetzal.

2.2 Understanding Input Buffer Overflows

It is challenging for designers to match the rate of on-device computing with data collection in energy-harvesting devices, given their uncertain environments. Even when computation latency is predictable, total processing time may vary due to variations in input power and event activity in the device's environment (Figure 2a). *Input power* affects processing time because an energy-harvesting device must collect its operational energy. Energy collection happens as a device operates, but if harvestable input power falls below operating power, then a device will eventually exhaust its energy. After exhausting its energy, the device must pause and spend time collecting enough energy to resume operating and complete processing the input. As input power varies, energy collection time varies, and with it, the time to process an input. *Event activity* affects processing time because an application may handle different inputs with different processing tasks. In the Figure 1 example, an input collected when there is more activity in the environment is more likely to be passed on to the ML inference and radio transmission tasks.

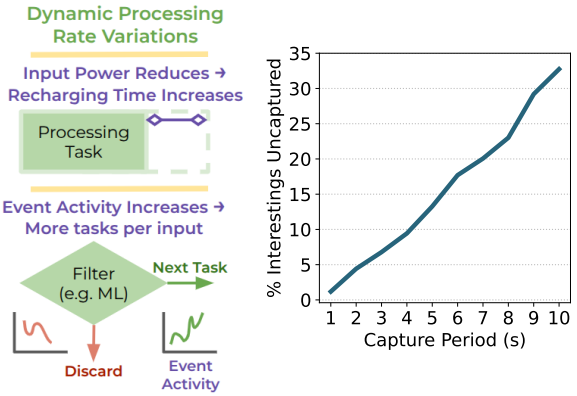


Figure 2. (a) Processing rate dynamically varies with Input-Power and Event-Activity. These variations can cause the device to process inputs slower than the capture rate, missing newer events. (b) Reducing capture rate still misses a large number of events.

End-to-end processing time variations due to input power and event activity can be substantial. For example, the end-to-end times for a radio task we evaluated range from 0.8s at high power to over 50s at low power. If end-to-end processing is slower than capture rate, the device must buffer inputs for later processing. Buffering is precarious because memories are small (100s of kB) in common energy-harvesting devices [3, 30, 81, 86]. Small memories can hold only a few (e.g., 5–10 in [23]) inputs before the buffer fills. Inputs that arrive to a full buffer are lost; solving this input buffer overflow problem is therefore the main focus of this work.

2.3 Naive solutions are ineffective at tackling input buffer overflows

Simple solutions like reducing the capture rate [62] or indiscriminately degrading tasks [7, 44] do not solve the input buffer overflow problem, as we quantitatively show in Figure 3. We simulated a solar-powered smart camera capturing images at 1FPS (like [23]), that discards background images and detects people using MobileNetv2 [78]. We study two forms of workload degradation. ML degradation uses LeNet [50] instead of MobileNetv2, which has a

higher chance of misclassifying images. Radio degradation sends a single byte indicating an 'interesting' event, instead of sending an entire image. Section 6 details our setup. The *Ideal* bar models an infinite input buffer that never overflows, only discarding interesting inputs due to ML model misclassifications. *NoAdapt* (NA) takes no action with a full input buffer, leading to many lost, interesting inputs. Many existing energy-harvesting systems [7, 23, 26, 43, 44, 46, 62] are similarly non-adaptive. Prior works present several policies for adaptation; we show that none address the IBO problem.

Degrade processing tasks? We evaluated a system that always degrades both ML and radio tasks (AD). The data show that always degrading reduces 'interesting' inputs discarded due to IBOs, but instead loses a lot of 'interesting' inputs to ML misclassifications while only reporting low-quality data.

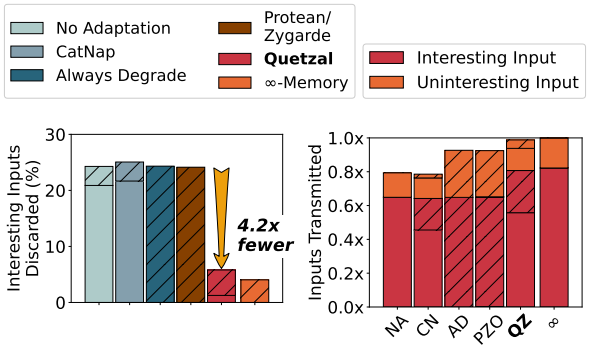
Degrade when the input buffer is full? CatNap (CN) [62] degrades tasks only when the input buffer is full. The data show that fixed-threshold degradation fails to adapt in time to avoid input loss, missing a lot of events to IBOs.

Degrade when input power is low? Zygarde [44] and Protean [7] (PZO) degrade tasks when input power falls below a fixed threshold, even if the input buffer is fairly empty. The data shows that adapting to fixed input-power thresholds results in unnecessary task degradations, with many 'interesting' inputs lost to ML misclassifications.

Decrease the capture rate? We implemented *NoAdapt* with capture rate degradation (also like CatNap [62]) and show different capture periods between 1s and 10s in Figure 2b. Unsurprisingly, with less frequent captures, the device fails to even capture a large fraction of interesting data.

The plot also shows that Quetzal reduces interesting inputs discarded by upto 4.2×. These benefits owe to Quetzal's energy-aware SJF scheduling policy, which selects tasks to run based on estimated end-to-end latencies, and degradation decisions made based on predicted input buffer overflows.

Paper Overview: Quetzal makes several contributions (Figure 4). The energy-aware SJF scheduling policy is rooted in queueing theory, and is the first to consider *both* execution time and energy collection time when scheduling tasks. The key idea is to use (measured) input power and (modeled) processing time to find the next shortest job to execute. Section 3 provides an overview of the scheduling policy and its underlying theory. Quetzal implements this policy in its scheduler, which is a runtime software system. A programmer links the scheduler into their application and the scheduler decides which job to execute and with what degree of task degradation. Section 4 describes the scheduler in detail. Quetzal's scheduler requires measurements of input power and estimates of job processing time, both of which Quetzal collects using a novel, yet simple hardware circuit (Section 5).



(a) Buffer Overflows (Solid), ML misclassifications (Hatch)

(b) High-quality (Solid), Low-quality (Hatch)

Figure 3. Naive solutions discard many interesting inputs. (a) Some solutions adapt too late or not at all, suffering many IBOs. (b) Others degrade tasks unnecessarily and suffer many ML misclassifications. Quetzal dynamically identifies IBO risk and degrades tasks only when required to avoid IBOs, discarding fewer interesting inputs and reporting more high-quality interesting inputs.

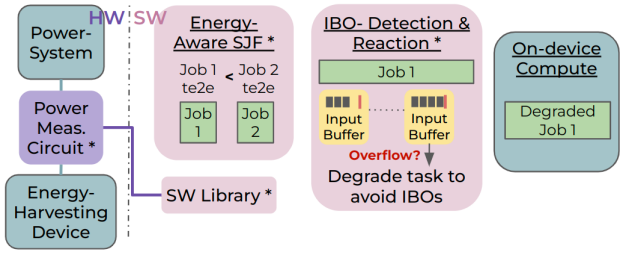


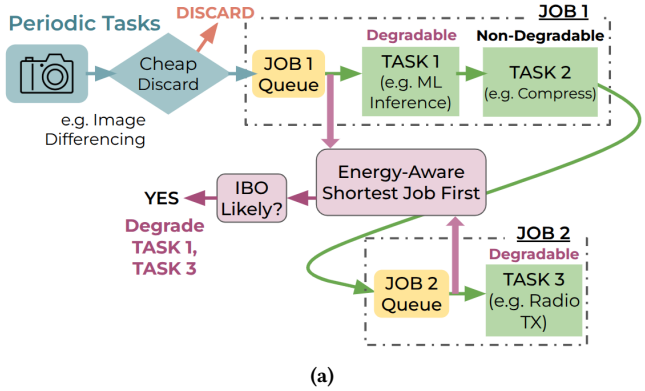
Figure 4. Overview of the proposed Quetzal software system with enabling hardware module (∗ : our contributions).

3 Modeling Energy-Harvesting Devices

Quetzal develops an energy-aware queueing model of a periodic energy-harvesting system, using it to compute the expected service time of a job. Quetzal implements the energy-aware shortest-job-first (SJF) scheduling policy, selecting jobs with the shortest expected service time. Quetzal then uses the service time estimates, along with Little's Law to predict if the input buffer will overflow during the execution of the scheduled job. In the event of a predicted overflow, Quetzal adapts a job's behavior using task degradation (Sec. 4).

3.1 Modeling the Input Buffer as a Queue

Quetzal models the input buffer as a queue [33], using this as a mathematical foundation for predicting input buffer overflows. An input arrives in the queue when the device buffers it in memory, possibly after a low-cost filtering step (e.g., discarding unchanged images). The queue thus has an input arrival rate, λ, which is based on the input period and input



(a)

Figure 5. An example execution flow for a Quetzal system with periodic tasks entering data into job queues. When data exits a queue, Quetzal first runs the Energy-aware SJF scheduler to select which job to execute. Quetzal then predicts if the selected job might lead to IBOs, degrading the degradeable task if an IBO is imminent.

pre-filtering. λ is higher (lower) during periods of higher (lower) event activity. The queue has a fixed capacity, limited by device memory size. The number of queued inputs, N, must remain below the queue capacity, or else new inputs are lost. A job is a sequence of tasks that process a queued input. A task is an application-specific, programmer-defined part of an application that may process input data or manipulate peripherals such as radios and sensors. When a job completes processing an input, that input leaves the queue. If the input needs additional processing, it can be re-inserted into the queue by the previous job. Some tasks are *degradable*, which means that they offer options of different time and energy cost for how they execute. Figure 5 shows the jobs and tasks in an example system.

3.2 Modeling End-to-end Service Time

A job has an expected service time, which is the sum of the service time of each of its constituent tasks weighted by the probability of executing each task. A task's service time is the end-to-end time taken to run that task for an input. A task's service time varies with input power, because when operating power exceeds harvestable power, energy collection time dominates, but when harvestable power exceeds operating power, processing time dominates. Equation (1) models a task's end-to-end processing time (S_{e2e}) as dependent on harvestable input power (P_{in}), given a task's time (t_{exe}) and energy (E_{exe}) cost. Quetzal uses this end-to-end service time definition for two things. First, the S_{e2e} of each task is a key variable in the energy-aware SJF algorithm described in Section 4. Second, S_{e2e} enables Quetzal to predict input buffer overflows, which we explain next.

Algorithm 1 Energy-aware SJF Algorithm

```
P_pred ← predictInputPower()
input_arrival_rate ← λ
min_job, min_E ← 0
for all job_i in jobs do
    E[S] ← 0
    for all task_i ← job_i_start, job_i_end do
        E[S]+ = [getProbability(task_i) * getSe2e(task_i, P_pred)]
    end for
    if E[S] < min_E then
        min_E ← E[S]
        min_job ← job_i
    end if
end for
```

$$S_{e2e} = max(t_{exe}, t_{chg}) = max(t_{exe}, \frac{E_{exe}}{P_{in}}) = max(t_{exe}, \frac{t_{exe} \times P_{exe}}{P_{in}}) \quad (1)$$

3.3 Predicting Input Buffer Overflows

Quetzal directly predicts when an input buffer overflow is likely by computing the expected number of inputs in the queue after running a particular job. If the expected number of inputs in the queue is larger than the queue's capacity, an overflow is likely. The system uses Little's Law (Equation (2)) to compute the expected occupancy of the queue at the completion of a scheduled job.

$$E[N] = \lambda \times S_{e2e} \quad (2)$$

Computing Little's Law requires the input arrival rate, λ and the expected service time for the scheduled job (S_{e2e}). The scheduler approximates input arrival rate by explicitly tracking how many of a window of recent inputs were inserted into the queue, i.e not discarded. A core contribution of this work is the use of Quetzal's *energy-aware* model of end-to-end service time to predict queue occupancy.

4 Quetzal System Design

Quetzal avoids input buffer overflows using our novel energy-aware SJF scheduling combined with its IBO-detection and reaction engine. Quetzal uses a set of software and hardware mechanisms to track input arrival rate, service time, input power, and power consumption.

4.1 Energy-aware SJF Scheduling

Energy-aware SJF (Algorithm 1) schedules the job with the shortest end-to-end service time (i.e., E[S]). SJF minimizes the mean wait time for other buffered inputs [33], alleviating pressure on the input buffer. E[S] for a job varies with input power and with the tasks active for a particular input; not all tasks processes every input. For example, in Figure 5, Job1:Task2 will only process inputs that are positively classified by Job1:Task1. Quetzal predicts E[S] for a job by summing the expected S_{e2e} for each task, weighted by its likelihood of executing. For jobs with the same E[S], Quetzal chooses the job that processes an older input. The scheduler

needs several pieces of information to identify the shortest job: P_{in}, t^i_{exe}, P^i_{exe}, $execution\_probability_i$, and input-arrival rate (λ), for each task i in the job. Quetzal predicts each based on measured historical values. For P_{in}, Quetzal measures the instantaneous input power using the hardware mechanism described in Section 5. For t_{exe} and P_{exe}, Quetzal executes and profiles each task in a separate profiling phase. Quetzal assumes that each task has a consistent t_{exe} and P_{exe} that can be profiled in advance. To predict a task's execution probability, Quetzal tracks the number of times a task executed over a window of jobs executed overall. The fraction of jobs that executed that task is Quetzal's estimate of the task's execution probability. To predict λ, the system tracks the number of times an input was stored in the input buffer from a previous window of captured inputs.

4.2 IBO-detection and reaction engine

Quetzal predicts imminent input buffer overflows by using Little's Law to estimate the number of buffered inputs after a job's completion. If the number of buffered inputs after a shortest job's completion exceeds the buffer's capacity, then the job's degradable task should be degraded to reduce the job's service time. Figure 5 shows this overflow prediction mechanism in an example. Algorithm 2 shows an algorithmic version of Quetzal's IBO-detection and reaction engine.

Reacting to Overflows: Quetzal reacts to imminent overflow by degrading a task to reduce the expected S_{e2e} for the degradeable task, consequently reducing the job's E[S]. As described in Section 5.2, Quetzal expects the programmer to provide quality-ordered degradation options for each degradeable task. Quetzal then steps down the degradation option list in the provided order, evaluating if the selected job avoids imminent IBOs with each option. Quetzal then executes the first degradation option that is predicted to avoid imminent IBOs. By selecting the highest-quality degradation option, Quetzal avoids unnecessarily degrading job quality. If no option removes the imminent IBO risk, Quetzal uses the option with the lowest S_{e2e} in order to reduce E[N].

Prediction and Reaction Requirements: Predicting and reacting to IBOs requires access to predicted P_{in}, as well as $\{t_{\text{exe}}, P_{\text{exe}}, execution\_probability\}$ for each degradation option. We also need the input-arrival rate (λ), and the current occupancy and maximum capacity of the input buffer.

4.3 Mitigating Prediction Error

Quetzal predicts per-job E[S] using historical values for several quantities, and can suffer from prediction errors. We mitigate these errors using a Proportional-Integral-Derivative (PID) controller, given its simplicity and low overheads. The PID controller provides an output that is proportional to current error (proportional constant K_p), the recent history of error (integral constant K_i) and the rate of error change (derivative constant K_d). We measure current error as the difference between predicted and observed values of E[S]; we then add the PID output to future E[S] predictions. If error is positive, the job executed for longer and the input buffer might be fuller than we predicted. The device then needs to be more conservative in future predictions. A positive PID output inflates future predictions and increases likelihood of task degradations. If the error is negative, the job finished sooner than anticipated, and the device might have more space in the input buffer. The device can therefore afford maintaining a higher task quality.

Algorithm 2 IBO-Detection and Reaction Algorithm

```
\* IBO-Detection * \
input_arrival_rate ← λ
current_occupancy                                   ▷ # inputs currently stored
buffer_limit                                        ▷ Memory capacity
E[S] ← min_E                                        ▷ From SJF Calculations
if input_arrival_rate × E[S] ≥ buffer_limit − current_occupancy then
 ▷ Little's Law
    IBO_predicted ← True
    \* IBO-Reaction * \
    E[S] ← ∑ non_deg_task_Se2e
    for all option_i in degradation_options do
        E[S]_temp = [getProbability(deg_task) * getSe2e(option_i, P_pred)]
        if input_arrival_rate × (E[S] + E[S]_temp) ≥ buffer_limit − current_occupancy then
            IBO_predicted ← True
        else
            IBO_predicted ← False
            < Select this Degradation Option >
            break
        end if
    end for
end if
```

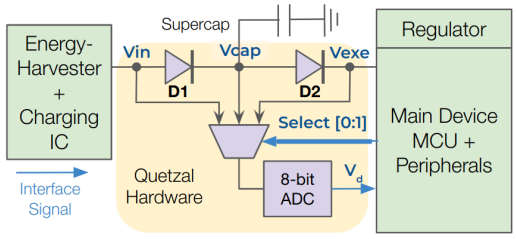


Figure 6. Overview of the proposed Quetzal hardware module, which uses four cheap components to compute the $\frac{P_{exe}}{P_{in}}$ ratio with simple operations, without using expensive divisions.

5 Quetzal Implementation

Quetzal uses a hardware-software solution to estimate and track the quantities required in implementing energy-aware SJF and input buffer overflow prediction. Both the software and hardware mechanisms are generic, making few assumptions of the power system, and operate with very low online computational and logic cost (i.e. avoiding division), supporting even the simplest microcontrollers (MCU).

5.1 Quetzal's Hardware-Software Solution

Quetzal's hardware measures input power (P_{in}) and operating power (P_{exe}) at runtime. Directly measuring these quantities in hardware allows Quetzal to account for drops and component wearout [74]. Quetzal's software measures task latency, t_{exe}, using hardware timers common in low-end microcontrollers. Quetzal's power measurement hardware (described below) makes profiling P_{exe} simple and accessible through a simple software interface. Quetzal computes the $\frac{P_{exe}}{P_{in}}$ ratio to estimate S_{e2e} for a task, when $P_{\text{exe}} \geq P_{\text{in}}$. Quetzal uses hardware support because evaluating this ratio hundreds of times per second would incur substantial overhead, especially in a low-end microcontroller lacking hardware support for division (e.g. MSP430 [86], ARM M0 [81]).

Algorithm 3 Computing S_{e2e} using our hardware

```
1: V_D1 ← (V_in - V_cap)
2: V_D2 ← recorded_voltage[task]
3: if V_D2 ≤ V_D1 then
4:     S_e2e ← t_exe [0]
5: else
6:     delta ← (V_D2 − V_D1)
7:     S_e2e ← t_exe [delta AND 0x03] * (1«(delta»3))
8: end if
```

Hardware for Power Measurement: We provide a new hardware circuit to simplify evaluating Eq (1). Our circuit uses four components: two diodes [24], one multiplexer [87] and one 8-bit ADC [25]. A microcontroller interfaces with our circuit using two I/O signals, one to select between three voltage measurements (V_{in}, V_{cap} and V_{exe}) and the other to receive 8-bit ADC measurements. Figure 6 shows our circuit. Eq (1) requires computing the $\frac{t_{exe} \times P_{exe}}{P_{in}}$ ratio when $P_{exe} \geq P_{in}$. We leverage semiconductor physics to compute this ratio, using the *Diode Law* to convert the division into simple arithmetic operations. We first reduce $\frac{P_{exe}}{P_{in}}$ to $\frac{I_{exe}}{I_{in}}$ by performing all measurements at the same voltage. Next, we transform the computation into logarithms: $2^{log_2(\frac{I_{exe}}{I_{in}})} = 2^{log_2(I_{exe}) - log_2(I_{in})}$. *Diode Law*: $V_d = \frac{kT}{q} * ln\frac{I}{I_0}$ then requires our hardware to only measure the diode voltage V_d, where k, q and T are the Boltzmann constant, the elementary charge and the Kelvin temperature respectively. Assuming V_{D1} and V_{D2} are the voltages across diodes D1 and D2 collected using a 8-bit ADC, we can convert $\frac{I_{exe}}{I_{in}}$ to $2^{\frac{q log_2(e) * V_{ADCMax}}{kT * 255} \times (V_{D2} - V_{D1})}$, where V_{ADCMax} is the maximum ADC voltage. The system measures V_{D2} (execution power) during profiling, and V_{D1} (input power) dynamically at run time. Setting V_{ADCMax} to 0.6V reduces the exponent term to $1/8 \times (V_{D2} - V_{D1})$ for temperatures between 25-50 C. We then rewrite the original ratio as: $2^{a.b} = 2^a \times 2^{0.b}$, where a and b are the integer and fractional parts. a is simply $(V_{D2} - V_{D1}) >> 3$, giving $2^a = 1 << ((V_{D2} - V_{D1}) >> 3)$. Since the exponent divides by 8, b can only take eight possible values (0, 0.125, ..). We pre-multiply the t_{exe} for each task at profile-time with all eight values; the lowest three bits in $(V_{D2} - V_{D1})$ decide which pre-multiplied t_{exe} is to be used for computing E[S]. Computing $\frac{t_{exe} \times P_{exe}}{P_{in}}$ then requires one subtraction, one lookup, two shift operations and one multiplication. On microcontrollers lacking hardware support for division, this sequence of operations is over 10× faster. The above computations are summarized in Algorithm 3.

Tracking task *execution_probability* and input-arrival rate: Our software library tracks task execution probability by maintaining a bit-vector of size $< task{-}window >$, where a 1/0 represents that the task did/did not execute for a given input. This bit-vector is atomically updated for all tasks on job completion, appending 1s (0s) for tasks that executed (did not execute). The bit-vectors for all tasks represent a history over the last $< task{-}window >$ completely processed inputs. Our software library tracks input-arrival rate in a similar fashion to task *execution_probability*, using a bit-vector of size $< arrival - window >$, where it appends 1s (0s) for inputs that are stored (not stored) in the memory-queue. Our software library maintains a 1-counter for each of these bit-vectors, which counts the number of 1s in the vector and is updated only when the bit-vector is modified.

Costs and Overheads: Our software library supports a maximum of 32 tasks, with 4 degradation options for each task, resulting in a memory footprint of 2,360 bytes. Our new module predicts the $\frac{P_{exe}}{P_{in}}$ ratio with $\leq$ 5.5% error for temperatures between 25-50 C. At each invocation, Quetzal requires multiple integer divisions ($num\_tasks + num\_degradation\_options$). With 10 Quetzal invocations per second and 32 tasks with 4 degradation options, our hardware module reduces Quetzal overheads on MSP430 [86] from 6.2% to 0.4%, while incurring 0.02% overheads on the Apollo 4 [3]. Our new hardware module directly reduces the energy cost associated with computing the $\frac{t_{exe} \times P_{exe}}{P_{in}}$ ratio. When Quetzal runs on devices lacking a hardware divider (e.g. MSP430), our module (12 cycles, 3.75nJ) reduces the ratio-computation energy cost by 92.5% compared with using software division (158 cycles, 49.37nJ). Even when Quetzal runs on devices with a hardware divider (e.g. Apollo 4), our module (5 cycles, 0.16nJ) reduces the ratio-computation energy cost by 62% compared with using the native hardware divider (13 cycles, 0.4nJ).

5.2 Programming Model

Quetzal provides a simple programmer interface, where energy-harvesting applications are written as simple *tasks* grouped into *jobs*. Tasks can coarsely associate with different processing kernels used in common energy-harvesting device firmwares; e.g. in [23], ML inference, compression and radio transmission can all be labelled as a separate task. Some tasks need to be degradable (e.g. ML and/or radio) to support reacting to IBOs. For this work, we require the programmer to combine tasks into a job, where each job has exactly one degradable task that is responsible for preventing IBOs for the entire job. A job can have other, non-degradable jobs.

Degradation Options: The programmer must provide a quality-ordered list of degradation options that Quetzal can profile at runtime, and react to IBOs with. Since quality is an application-specific metric, we assume that the application developer can reason about quality for different compute degradation options; Quetzal only requires an ordered list.
Application Requirements for Quetzal: Quetzal targets applications which cannot suffer reduced capture rates, can provide quality degradation options (at least for some tasks), and have dynamically variable end-to-end processing and/or capture rates (typical for energy-harvesting devices). Quetzal also relies on consistent t_{exe} and P_{exe} for each task; extending Quetzal to support variable execution costs represents an interesting future research direction.

6 Methodology

We demonstrated Quetzal's advantages over several baseline systems, including systems proposed in prior research (6.1). We used two rounds of experiments in our evaluation. We first implemented an end-to-end hardware experiment to show the benefits of Quetzal's dynamic adaptations (6.2). We then used a custom simulator to study a large design space in a repeatable and reliable manner (6.3).

6.1 Baselines

Non-adaptive: We motivate Quetzal's dynamic adaptation with baselines that do not adapt task quality at runtime. A *NoAdapt* (NA) system represents a vast majority of prior energy-harvesting systems [23, 46], running tasks at highest available quality. We also study a *Always Degrade* (AD) system, which executes tasks at their lowest quality level.
Fixed-threshold-adaptive: We motivate Quetzal's degradation protocol with baselines that degrade tasks when the input buffer is filled to a static threshold, expressed as a percentage of the input buffer (0-100%). One such example was proposed in CatNap (CN) [62], which degrades tasks *after* the input buffer is full, i.e. threshold=100%. Two prior works (Zygarde [44],Protean [7]) degrade tasks when input power falls below static thresholds computed as fixed fractions of the harvester maximum. We observed that the real-world input power traces we used in our experiments commonly stayed below these thresholds, demonstrating a fundamental flaw in using datasheet maximums (ZGO). Therefore, we also studied an idealized version (ZGI) of these baselines using the maximum observed power in an experiment to establish the static threshold, which is practically unimplementable given the need for oracular knowledge of the future.
Scheduling Policies: We motivate scaling S_{e2e} with input power by studying a *Avg. S_{e2e}* system which uses an average of past S_{e2e} measurements for each task in Algorithms 1 and 2. We also motivated our Energy-aware SJF policy by comparing against commonly-used policies: First-Come-First-Served (FCFS) and Last-Come-First-Served (LCFS).

6.2 Hardware Experiment

We motivate Quetzal with an end-to-end hardware experiment, implementing its Energy-aware SJF and IBO-detection and reaction engine. We studied a representative person-detection application (like [23, 46]), where a device identifies and reports images containing people.
Hardware: We used an Ambiq Apollo 4 MCU [3] as the main MCU for our system. We interfaced this MCU with an ultra-low-power camera [40] and a LoRa radio [42]. We implemented an energy-harvester circuit using the BQ25504 [88] and multiple voltage regulators to supply power to our setup, with a 33mF supercapacitor [5] to store harvested energy. Figure 7 shows our hardware experiment setup.
Software: We implemented the software pipeline shown in Figure 1, periodically capturing images at 1 FPS. Each captured image is compared with the previous image using pixel-wise differencing, with 'different' images stored into the memory buffer. Each stored image is then processed by a pipeline consisting of ML models trained for person-detection, image compression and radio transmission. The *NoAdapt* system processed each stored image in the order they were captured, while Quetzal employed its Energy-aware SJF policy. Our Quetzal implementation used the bit-vectors described in 5.1 to track required quantities.
Time-Varying Environment: We represented the environment in two dimensions: harvestable power and sensing-event activity. Using a real energy-harvesting dataset [32], we emulated a solar harvester with a programmable power supply [79] as input to the BQ25504, adjusting the power-supply output according to the dataset trace. We modeled sensing events in terms of their durations and interarrival times, where some events are 'interesting' to the application, with the rest being 'uninteresting'. The interarrival period between events represents inactivity. 6.4 details our application. We implemented events using a secondary MCU to ensure precise repeatability of our experiments, similar to the setup in [23]. The secondary MCU asserted two input-output (I/O) pins to indicate the presence and interestingness of an event. The main system recorded both pins at the time of image capture. Images captured when the first pin was HIGH were treated as 'different' and stored in the buffer. Images captured when the second pin was also HIGH were treated as 'interesting'. The main system used the ML models' misclassification rates to process 'different' inputs, discarding 'interesting' ones at the false negative rate and transmitting 'uninteresting' ones at the false positive rate. Although using the I/O setup, the main system still executed every scheduled job to incur their time and energy costs.

6.3 Simulation

We performed a wider range of studies using a custom simulator based on fixed-increment time progression (1ms steps).

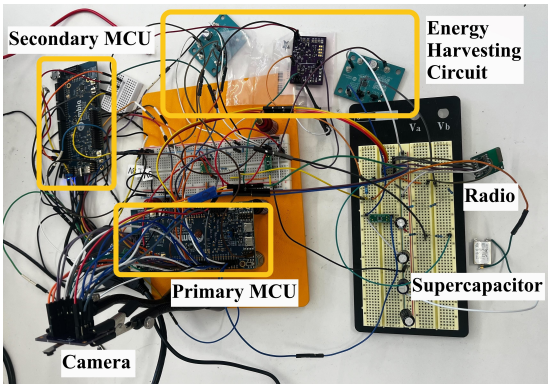


Figure 7. Our end-to-end hardware experiment setup.

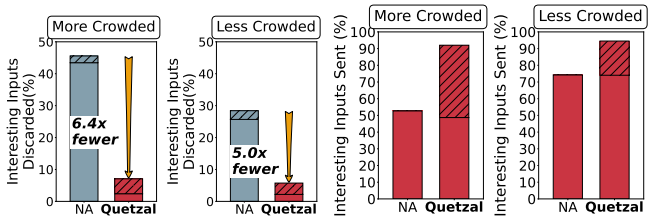
Table 1. Experiment Details

Component	Values
Compute	[HW & Sim.] Ambiq Apollo 4 (Input buffer=10imgs)
Expt. Config.	Capture Rate=1FPS, Maximum 'Interesting' Duration: , More Crowded: 600s, Crowded: 60s, Less Crowded: 20s
App. Details	High-Q ML=MobileNetV2 [78], Low-Q ML=LeNet [50], High-Q Radio=Full JPEG Image, Low-Q Radio=Single Byte
Compute	[Sim.] MSP430FR5994 (Input buffer=10imgs)
Expt. Config.	Capture Rate=1FPS, Maximum 'Interesting' Duration: 10s
App. Details	High-Q ML=Int-16 LeNet, Low-Q ML=Int-8 LeNet, Radio: Same as Apollo
Quetzal Params	<task-window>=64, <arrival-window>=256, PID Controller: K_p=5e-6, K_i=1e-6, K_d=1

Our simulations mirrored our hardware experiment, studying a person-detection application. We used the same input power and event traces in our simulations. We represented the actual device as a set of tasks characterized by their latency and energy values, measured on real hardware using the Saleae Logic Analyzer [75] and Otii Ace Pro [70]. We also modeled an energy storage element, to which we add harvested energy every simulator time step. We 'run' a task by incrementing the simulation timer by the task's latency, and subtracting the task's energy from the energy storage. We implemented a just-in-time (JIT) checkpointing system [8, 9, 47, 61, 64] to support intermittent computing. Before selecting a job to run, we evaluated any scheduling policy and degradation-logic pertaining to the simulated system, incurring its overheads. Our simulated device ran in parallel to the simulated environment, with the device 'capturing' an event if it overlapped with a sensing task.

6.4 Application Details:

We implemented sensing-event activity using an open-source video surveillance dataset [67] to randomly draw event durations and interarrival times. Our hardware and simulation experiments used 100 and 1000 events respectively. The systems we studied generated more interesting inputs the longer an interesting event lasted. We generated multiple unique sensing environments using limits on the event durations.



(a) Buffer Overflows (Solid), ML misclassifications (Hatch)

(b) High-quality (Solid), Low-quality (Hatch)

Figure 8. Hardware experiment showing that Quetzal reduces the total interesting inputs discarded compared to the *NoAdapt* system across two sensing environments. Quetzal degrades tasks in response to imminent IBOs, reporting more interesting inputs.

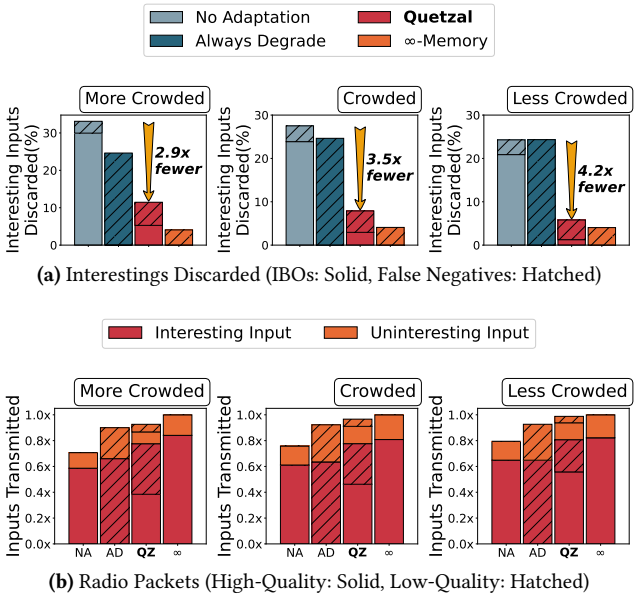
Per-experiment limits are detailed in Table 1. Our hardware experiment used one MCU (Ambiq Apollo 4), while our simulations studied two (Apollo 4 and the MSP430FR5994 [86]). We used two ML models as high and low-quality options for the ML-inference task, and sending full images vs single bytes as high and low-quality options for the radio task. Full images can be audited by the receiver, hence represent higher quality. We trained all ML models using the EuroCity pedestrian dataset [14]. We used JPEG [65] to compress images. The Apollo 4 MCU can efficiently compress images; all systems therefore always compress images before storing in the input buffer. We modeled the solar harvester as 6 cells of a commercial product [45]; we study the effect of different harvester cells in 7.3. We implemented our PID controller as described in [69], tuning it according to [89].

7 Evaluation

Quetzal's goal is to reduce the interesting inputs missed by an energy-harvesting device due to input buffer overflows (in our evaluation, 'interesting' inputs contain people). We show Quetzal meets this goal using two rounds of experiments. First, we show Quetzal outperforms *NoAdapt* in an end-to-end hardware experiment (7.1), across two different sensing environments. Second, we present a wider simulation-based study that shows Quetzal outperforming multiple baselines in different sensing environments (7.2). Finally, we study Quetzal's sensitivity to different system parameters and schedulers (7.3).

7.1 End-to-end Hardware Experiment

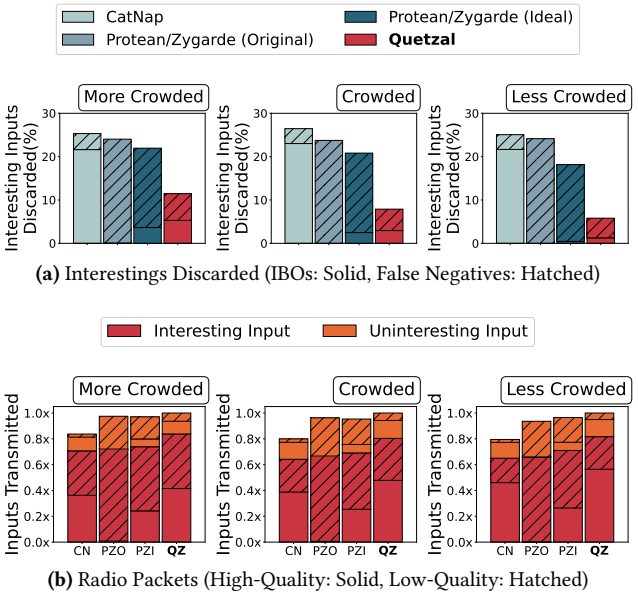
We evaluated Quetzal's ability to reduce discarded interesting inputs in an end-to-end energy-harvesting experiment (setup in 6.2). Figure 8a shows that Quetzal reduces discarded interesting inputs compared to a *NoAdapt* system by 6.4× and 5× in two sensing environments. Figure 8b shows that by reducing input buffer overflows, Quetzal is able to report



(a) Interestings Discarded (IBOs: Solid, False Negatives: Hatched)

(b) Radio Packets (High-Quality: Solid, Low-Quality: Hatched)

Figure 9. Quetzal reduces the total interesting inputs discarded (% of all interesting inputs) compared to a *NoAdapt* baseline. We show an *Ideal* baseline (∞-memory) for reference. (b) shows Quetzal reporting a larger number of interesting inputs (norm. to ∞-memory) by identifying opportunities to degrade task quality.



(a) Interestings Discarded (IBOs: Solid, False Negatives: Hatched)

(b) Radio Packets (High-Quality: Solid, Low-Quality: Hatched)

Figure 10. (a) Quetzal reduces the % of interesting inputs discarded compared to prior work [7, 44, 62]. (b) Quetzal reports a larger number of high-quality interesting inputs by degrading task quality only when IBOs are imminent.

74% and 27% more interesting inputs. In contrast to a *NoAdapt* system, Quetzal dynamically identifies energy-aware shortest jobs and detects imminent IBOs, degrading tasks in response. These degradations, represented by the hatched bars in Figure 8a (False negatives) and Figure 8b (Low-quality radio packets), allow Quetzal to significantly reduce the total interesting inputs missed by an energy-harvesting system.

7.2 Advantages of Quetzal

Figures 9 to 11 demonstrate Quetzal's advantages over several baselines, collected using our custom simulator. Our results show that Quetzal is able to reduce total interesting inputs discarded by up to 4.2×, compared to *NoAdapt*.

vs *NoAdapt, Always Degrade*: Figure 9 contrasts Quetzal (QZ) with three baselines: *NoAdapt* (NA) and *Always Degrade* (AD) systems which run all tasks at highest and lowest available quality respectively, and a system with infinite input buffer entries. Across three sensing environments (from More Crowded to Less), Quetzal discards 2.9×, 3.5× and 4.2× fewer interesting inputs than *NoAdapt*. Quetzal reduces the interesting inputs discarded due to just IBOs by 5.7×, 8.1× and 16.6×. Quetzal's combination of energy-aware SJF scheduling and dynamic task degradation based on imminent IBOs allows it to outperform *NoAdapt*. *Always Degrade* reduces IBOs but its degraded ML causes a large number of misclassifications (false negatives). Compared to *Always Degrade*, Quetzal discards 2.2×, 3.1× and 4.2× fewer interesting inputs. Further, *Always Degrade* only reports low-quality inputs (single-byte), severely degrading application quality. Quetzal applies task degradation only when IBOs are imminent, reporting 49.6%, 59.5% and 69.1% of transmitted interesting inputs at high quality. Finally, Quetzal reports 92%, 96% and 98% of the interesting inputs reported by an ∞-memory baseline, highlighting the impact of Quetzal's Energy-aware SJF and IBO-detection and reaction engine.

vs Prior Work: We also implemented several solutions presented in prior work: CatNap [62] (CN), Protean [7], Zygarde [44]. Figure 10 shows that existing systems cannot tackle the IBO problem as effectively as Quetzal. CatNap degrades tasks only when the input buffer is 100% full, adapting too late to avoid IBOs successfully. In contrast, Quetzal dynamically adapts to imminent IBOs irrespective of the current buffer occupancy, discarding 2.2× (4.1×), 3.4× (7.8×) and 4.3× (17.2×) fewer total (IBOs-only) interesting inputs than CatNap in three sensing environments. Protean/Zygarde react to fixed input-power levels, degrading tasks when the input power falls below static thresholds. As described in 6.1, we study two variants: PZO is what the works proposed and PZI is an idealized, unimplementable version. Our results show that both variants unnecessarily degrade tasks when input power crosses the static thresholds, even when no IBOs are imminent. By dynamically adapting to imminent IBOs, Quetzal discards 1.9×, 2.6× and 3.1× fewer total interesting

inputs than even the unrealizable PZI baseline, outperforming PZO by an even larger margin. By degrading tasks only when required, Quetzal is able to report 1.7×, 1.9× and 2.1× more high-quality interesting inputs than PZI.

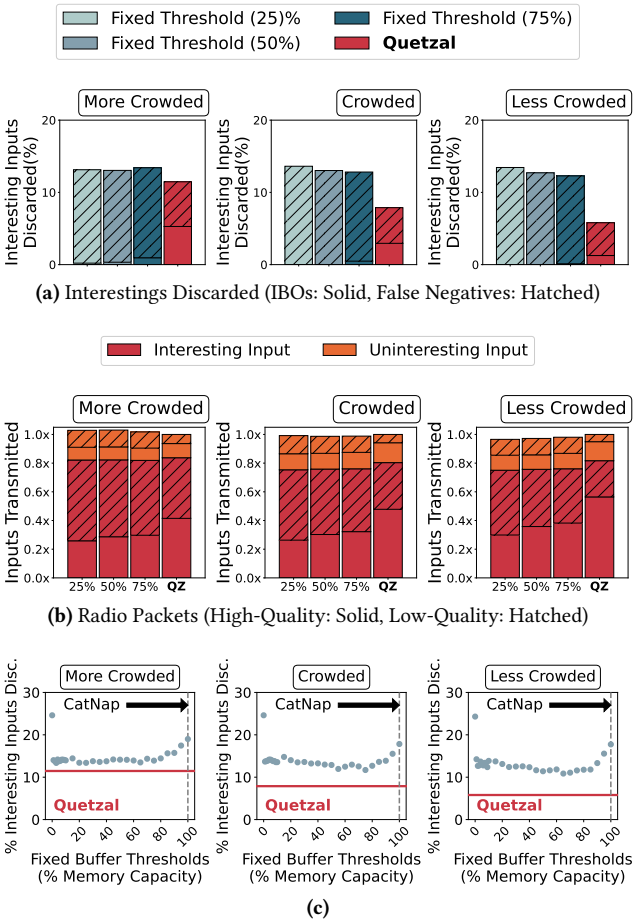
vs Fixed-threshold-adaptive systems: We motivate Quetzal's dynamic task degradation with systems that degrade tasks when the input buffer is filled to a fixed threshold. Figure 11a shows Quetzal outperforming three different fixed thresholds (25%, 50%, 75%). Computing a geometric mean over the three baselines, Quetzal discards 1.15×, 1.67× and 2.2× fewer total interesting inputs across three sensing environments respectively. Responding to a fixed buffer-occupancy threshold leads to unnecessary degradations, increasing ML misclassifications and low-quality radio packets. Figure 11b shows that Quetzal degrades tasks only when its engine indicates imminent IBOs, sending (geometric mean) 48%, 62% and 64% more high-quality interesting inputs. Across the entire range of thresholds, Figure 11c shows that Quetzal outperforms fixed-threshold adaptive systems no matter what threshold is used, as *energy-harvesting systems must adapt only when an IBO is imminent to reduce discarded interesting inputs while maintaining high application quality.*

7.3 Sensitivity Analysis

Energy-aware S_{e2e}: Quetzal predicts per-task S_{e2e} by scaling t_{exe} and E_{exe} values to the current input power. We motivate this scaling by comparing to a *Avg. S_{e2e}* system that uses an average of previously observed S_{e2e} values with Quetzal's Energy-aware SJF and IBO-detection and reaction engine. Not scaling S_{e2e} to input power causes *Avg. S_{e2e}* to incorrectly predict IBOs, degrading tasks unnecessarily. Figure 12 shows S_{e2e} scaling reduces discards interesting inputs by 2.2×, 3.1× and 4.2× compared to using average S_{e2e} values.

Benefits of Energy-aware SJF: We evaluate Quetzal systems equipped with different scheduling policies in Figure 12. Compared to using the common FCFS and LCFS policies, Energy-aware SJF is able to reduce discarded interesting inputs by 1.8×, 2.3×, 3× and 1.5×, 2×, 2.7× respectively. Energy-aware SJF's advantages are derived from identifying shortest jobs at the given input power to empty the input buffer faster. Processing inputs in the same order as they are captured is also unable to reduce mean waiting times, and Energy-aware SJF discards 1.4×, 1.8× and 2.6× fewer interesting inputs. Using Energy-aware SJF allows Quetzal*to reduce the mean waiting time, reducing the risk of the input buffer being full and subsequent IBOs.*

Quetzal is versatile: We show Quetzal's versatility by implementing several systems on a MSP430 microcontroller in Figure 13. The baselines either discard a large number of interesting inputs (*NoAdapt* and CatNap) or overly degrade the quality of tasks (*Always Degrade*, fixed-threshold-adaptive, Protean/Zygarde). Using a MSP430 microcontroller, Quetzal discards 2.8× fewer interesting inputs than *NoAdapt*, while



(a) Interestings Discarded (IBOs: Solid, False Negatives: Hatched)

(b) Radio Packets (High-Quality: Solid, Low-Quality: Hatched)

(c)

Figure 11. Quetzal reduces the total interesting inputs discarded compared to fixed-threshold-adaptive systems while sending a higher number of high-quality interesting inputs. Quetzal's advantages hold across the entire range of fixed thresholds, motivating the need for a system that dynamically adapts to imminent IBOs.

sending 40% more high-quality interesting inputs compared to the best baseline (fixed-thresholding at 75%).

System Parameters: Figure 14 shows how Quetzal's results vary with three configurable system parameters using Apollo 4: harvester cell count, <arrival-window> and <task-window> for the 'More Crowded' environment. We selected parameter values that maximized total interesting events while maintaining a large amount of high-quality reporting.

8 Related & Future Work

Quetzal relates to energy-harvesting sensors, on-device computing, runtime modeling and dynamic task adaptations.

Energy-Harvesting Devices: Recent work has investigated several aspects of energy-harvesting devices [1, 20, 21, 23,

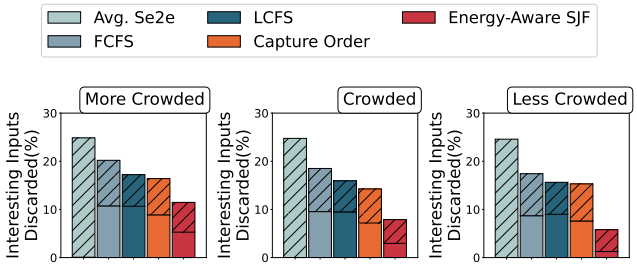
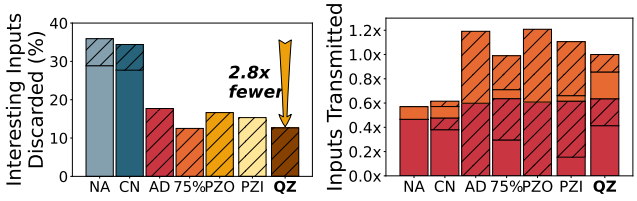


Figure 12. Quetzal using different scheduling policies, all equipped with its IBO-detection and reaction engine. Using the Energy-aware SJF policy allows Quetzal to reduce the interesting inputs discarded, both due to IBOs (solid) and False Negatives (hatched).



(a) Solid: IBO, Hatched: False Negatives

(b) Solid: High-quality, Hatched: Low-quality

Figure 13. (a) Interesting inputs discarded and (b) Radio packet distribution (Red: interesting, Orange: uninteresting) for Quetzal and multiple baselines running on a MSP430 microcontroller. Quetzal provides a general solution that is microcontroller-agnostic.

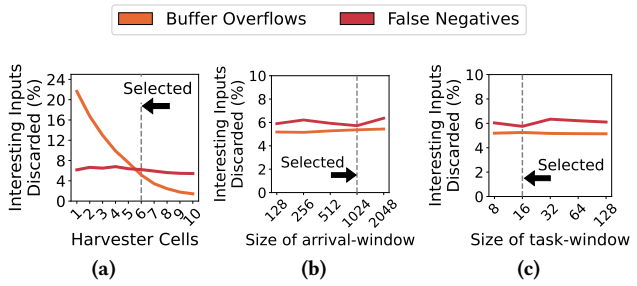


Figure 14. Impact of varying several system parameters. Vertical dashed line shows the values we used for our primary experiments.

26, 35, 46, 66, 76, 91], including reconfigurable power systems [18, 37, 38], debugging and evaluation tools [15, 36, 63, 83] and real-time clocking systems [19, 39, 49, 71]. Culpeo is a recent work that exposes power-system properties (ESR) to software [74]. Quetzal is agnostic of power-system details, incorporating any ESR-effects in its measurements.

Computing in resource-constrained devices: Recent work motivates complex, on-device compute, even on resource-constrained energy-harvesting devices. Some propose new energy-minimal architectures [28–30], while others propose software optimizations for complex processing tasks (e.g. ML) [31, 53, 54]. Quetzal is directly related to these works, enabling complex processing within tight memory constraints.
Intermittent Computing: Prior work studies intermittent computing, where energy-harvesting devices are designed to tolerate mid-computational power-failures. These works presented new programming models and hardware solutions for intermittent computing [4, 6, 8, 9, 11, 13, 16, 17, 47, 48, 56, 59–61, 64, 72, 73, 83–85, 90]. Quetzal is orthogonal to intermittent computing, operating on tasks that atomically complete within a single charge of the energy-storage device.
Runtime Modeling: Quetzal models several quantities for it's Energy-aware SJF policy and IBO-detection and reaction engine. Quetzal could employ existing performance models [51, 57, 68, 82] and runtime energy prediction [12, 51, 80, 82] to improve it's predictions. Quetzal could incorporate intermittent computing costs in it's E[S] estimates using existing models [27, 77]. CleanCut [17] models the energy cost distribution of different paths through a program to generate intermittent-safe task decompositions. Future versions of Quetzal could similarly leverage time and power distributions to support tasks with variable execution costs.
Dynamic Adaptations in Energy-harvesting Devices: Recent energy-harvesting systems [7, 44, 62] propose dynamic scheduling and adaptation; Quetzal outperforms these systems as they do not target avoiding IBOs. Other works dynamically adapt to maintain operating power under a power cap [10, 41, 52, 92], which we interpret to be the same as the Protean/Zygarde baseline in our context. One prior work [58] considers adapting computation quality to input power and data freshness, as well as opportunistically employing SIMD computations when older and newer data processing aligns. However, this work does not adapt processing quality to *input buffer* state, and therefore cannot directly address the IBO challenge. AdaMICA [2] proposes parallel, multicore computing in energy-harvesting systems; Quetzal assumes a single-core device in this paper. Multicore parallelism can provide Quetzal with even more powerful adaptation options, helping it further reduce IBOs.

9 Conclusion

We presented Quetzal, a new system to reduce events missed due to IBOs, while avoiding unnecessary task degradations. Quetzal's Energy-aware SJF policy selected the job with the lowest end-to-end time at current environmental conditions, allowing the input buffer to empty more quickly. Quetzal's IBO-detection and reaction engine predicts if the selected job would cause an IBO, and reacts to imminent IBOs by degrading the selected job. Our new hardware circuit simplifies

Quetzal's computations with cheap components. Our evaluation shows that Quetzal outperforms baselines in different environments, reducing missed events by up to 4.2×.

Acknowledgements

We thank our reviewers for their helpful feedback and guidance. We thank members of the *abstract* research group at Carnegie Mellon University for contributing important critical feedback at all stages of the project. This work was supported in part by National Science Foundation Award #2106862.

References

[1] Mikhail Afanasov, Naveed Anwar Bhatti, Dennis Campagna, Giacomo Caslini, Fabio Massimo Centonze, Koustabh Dolui, Andrea Maioli, Erica Barone, Muhammad Hamad Alizai, Junaid Haroon Siddiqui, and Luca Mottola. 2020. Battery-Less Zero-Maintenance Embedded Sensing at the Mithræum of Circus Maximus. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems* (Virtual Event, Japan) *(SenSys '20)*. Association for Computing Machinery, New York, NY, USA, 368–381. https://doi.org/10.1145/3384419.3430722

[2] Khakim Akhunov and Kasim Sinan Yildirim. 2022. AdaMICA: Adaptive Multicore Intermittent Computing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6, 3, Article 98 (sep 2022), 30 pages. https://doi.org/10.1145/3550304

[3] Ambiq. [n.d.]. Apollo 4 Blue Plus. https://ambiq.com/apollo4-blue-plus/.

[4] A. Arreola, D. Balsamo, G. Merrett, and A. Weddell. 2018. RESTOP: Retaining External Peripheral State in Intermittently-Powered Sensor Systems. *Sensors (Basel)* 18, 1 (Jan. 2018), 172. https://doi.org/10.3390/s18010172

[5] AVX. 2019. BestCap® Ultra-low ESR High Power Pulse Supercapacitors. http://catalogs.avx.com/BestCap.pdf.

[6] Sara S. Baghsorkhi and Christos Margiolas. 2018. Automating Efficient Variable-Grained Resiliency for Low-Power IoT Systems. In *Proceedings of the 2018 International Symposium on Code Generation and Optimization* (Vienna, Austria) *(CGO 2018)*. Association for Computing Machinery, New York, NY, USA, 38–49. https://doi.org/10.1145/3168816

[7] Abu Bakar, Rishabh Goel, Jasper de Winkel, Jason Huang, Saad Ahmed, Bashima Islam, Przemyslaw Pawelczak, Kasim Sinan Yildirim, and Josiah Hester. 2023. Protean: Adaptive Hardware-Accelerated Intermittent Computing. *GetMobile: Mobile Comp. and Comm.* 27, 1 (may 2023), 5–10. https://doi.org/10.1145/3599184.3599186

[8] D. Balsamo et al. 2015. Hibernus: Sustaining Computation During Intermittent Supply for Energy-Harvesting Systems. *IEEE Embedded Systems Letters* 7, 1 (March 2015), 15–18. https://doi.org/10.1109/LES.2014.2371494

[9] D. Balsamo et al. 2016. Hibernus++: A Self-Calibrating and Adaptive System for Transiently-Powered Embedded Devices. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35, 12 (2016), 1968–1980. https://doi.org/10.1109/TCAD.2016.2547919

[10] Domenico Balsamo, Benjamin J. Fletcher, Alex S. Weddell, Giorgos Karatziolas, Bashir M. Al-Hashimi, and Geoff V. Merrett. 2019. Momentum: Power-Neutral Performance Scaling with Intrinsic MPPT for Energy Harvesting Computing Systems. *ACM Trans. Embed. Comput. Syst.* 17, 6, Article 93 (Jan. 2019), 25 pages. https://doi.org/10.1145/3281300

[11] G. Berthou, T. Delizy, K. Marquet, T. Risset, and G. Salagnac. 2019. Sytare: A Lightweight Kernel for NVRAM-Based Transiently-Powered Systems. *IEEE Trans. Comput.* 68, 9 (Sep. 2019), 1390–1403. https://doi.org/10.1109/TC.2018.2889080

[12] Abhishek Bhattacharjee and Margaret Martonosi. 2009. Thread Criticality Predictors for Dynamic Performance, Power, and Resource Management in Chip Multiprocessors. In *Proceedings of the 36th Annual International Symposium on Computer Architecture* (Austin, TX, USA) *(ISCA '09)*. Association for Computing Machinery, New York, NY, USA, 290–301. https://doi.org/10.1145/1555754.1555792

[13] Abhishek Bhattacharyya, Abhijith Somashekhar, and Joshua San Miguel. 2022. NvMR: Non-Volatile Memory Renaming for Intermittent Computing. In *Proceedings of the 49th Annual International Symposium on Computer Architecture* (New York, New York) *(ISCA '22)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3470496.3527413

[14] Markus Braun, Sebastian Krebs, Fabian B. Flohr, and Dariu M. Gavrila. 2019. EuroCity Persons: A Novel Benchmark for Person Detection in Traffic Scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019), 1–1. https://doi.org/10.1109/TPAMI.2019.2897684

[15] Alexei Colin, Graham Harvey, Brandon Lucia, and Alanson P. Sample. 2016. An Energy-Interference-Free Hardware-Software Debugger for Intermittent Energy-Harvesting Systems. In *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems* (Atlanta, Georgia, USA) *(ASPLOS '16)*. Association for Computing Machinery, New York, NY, USA, 577–589. https://doi.org/10.1145/2872362.2872409

[16] Alexei Colin and Brandon Lucia. 2016. Chain: Tasks and Channels for Reliable Intermittent Programs. In *Proceedings of the 2016 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications* (Amsterdam, Netherlands) *(OOPSLA 2016)*. ACM, New York, NY, USA, 514–530. https://doi.org/10.1145/2983990.2983995

[17] Alexei Colin and Brandon Lucia. 2018. Termination Checking and Task Decomposition for Task-Based Intermittent Programs. In *Proceedings of the 27th International Conference on Compiler Construction* (Vienna, Austria) *(CC 2018)*. Association for Computing Machinery, New York, NY, USA, 116–127. https://doi.org/10.1145/3178372.3179525

[18] Alexei Colin, Emily Ruppel, and Brandon Lucia. 2018. A Reconfigurable Energy Storage Architecture for Energy-harvesting Devices. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems* (Williamsburg, VA, USA) *(ASPLOS '18)*. ACM, New York, NY, USA, 767–781. https://doi.org/10.1145/3173162.3173210

[19] 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* (Lausanne, Switzerland) *(ASPLOS '20)*. Association for Computing Machinery, New York, NY, USA, 53–67. https://doi.org/10.1145/3373376.3378464

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

[21] Bradley Denby, Emily Ruppel, Vaibhav Singh, Shize Che, Chad Taylor, Fayyaz Zaidi, Swarun Kumar, Zac Manchester, and Brandon Lucia. 2022. Tartan Artibeus: A Batteryless, Computational Satellite Research Platform. (2022).

[22] Harsh Desai and Brandon Lucia. 2020. A Power-Aware Heterogeneous Architecture Scaling Model for Energy-Harvesting Computers. *IEEE Computer Architecture Letters* 19, 1 (2020), 68–71. https://doi.org/10.1109/LCA.2020.2989440

[23] 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

[24] Diodes Incorporated. [n.d.]. SDM40E20LC-7. https://www.diodes.com/assets/Datasheets/SDM40E20L_S_C_A.pdf.

[25] Maged ElAnsary, Jianxiong Xu, José Sales Filho, Gairik Dutta, Liam Long, Aly Shoukry, Camilo Tejeiro, Chenxi Tang, Enver Kilinc, Jaimin Joshi, et al. 2021. 28.8 Multi-Modal Peripheral Nerve Active Probe and Microstimulator with On-Chip Dual-Coil Power/Data Transmission and 64 2 nd-Order Opamp-Less ΔΣ ADCs. In *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, Vol. 64. IEEE, 400–402.

[26] Francesco Fraternali, Bharathan Balaji, Yuvraj Agarwal, Luca Benini, and Rajesh Gupta. 2018. Pible: Battery-free Mote for Perpetual Indoor BLE Applications. In *Proceedings of the 5th Conference on Systems for Built Environments* (Shenzen, China) *(BuildSys '18)*. ACM, New York, NY, USA, 168–171. https://doi.org/10.1145/3276774.3282822

[27] Fatemeh Ghasemi, Lukas Liedtke, and Magnus Jahre. 2023. PES: An Energy and Throughput Model for Energy Harvesting IoT Systems. In *2023 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 13–23. https://doi.org/10.1109/ISPASS57527.2023.00011

[28] Graham Gobieski et al. 2019. MANIC: An Energy-Efficient, Parallel Architecture for Ultra-Low-Power Embedded Systems. In *Proceedings of International Symposium on Microarchitecture*.

[29] Graham Gobieski, Ahmet Oguz Atli, Kenneth Mai, Brandon Lucia, and Nathan Beckmann. 2021. Snafu: An Ultra-Low-Power, Energy-Minimal CGRA-Generation Framework and Architecture. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. 1027–1040. https://doi.org/10.1109/ISCA52012.2021.00084

[30] Graham Gobieski, Souradip Ghosh, Marijn Heule, Todd Mowry, Tony Nowatzki, Nathan Beckmann, and Brandon Lucia. 2022. RipTide: A Programmable, Energy-Minimal Dataflow Compiler and Architecture. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 546–564. https://doi.org/10.1109/MICRO56248.2022.00046

[31] Graham Gobieski, Brandon Lucia, and Nathan Beckmann. 2019. Intelligence Beyond the Edge: Inference on Intermittent Embedded Systems. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems* (Providence, RI, USA) *(ASPLOS '19)*. ACM, New York, NY, USA, 199–213. https://doi.org/10.1145/3297858.3304011

[32] M. Gorlatova, A. Wallwater, and G. Zussman. 2011. Networking Low-Power Energy Harvesting Devices: Measurements and Algorithms. In *Proc. IEEE INFOCOM'11*.

[33] Mor Harchol-Balter. 2013. *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge University Press. https://doi.org/10.1017/CBO9781139226424

[34] Timm Haucke and Volker Steinhage. 2021. Exploiting Depth Information for Wildlife Monitoring. arXiv:cs.CV/2102.05607

[35] Josiah Hester, Travis Peters, Tianlong Yun, Ronald Peterson, Joseph Skinner, Bhargav Golla, Kevin Storer, Steven Hearndon, Kevin Freeman, Sarah Lord, Ryan Halter, David Kotz, and Jacob Sorber. 2016. Amulet: An Energy-Efficient, Multi-Application Wearable Platform. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM* (Stanford, CA, USA) *(SenSys '16)*. ACM, New York, NY, USA, 216–229. https://doi.org/10.1145/2994551.2994554

[36] 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* (Memphis, Tennessee) *(SenSys '14)*. Association for Computing Machinery, New York, NY, USA, 1–15. https://doi.org/10.1145/2668332.2668336

[37] Josiah Hester, Lanny Sitanayah, and Jacob Sorber. 2015. Tragedy of the Coulombs: Federating Energy Storage for Tiny, Intermittently-Powered Sensors. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems* (Seoul, South Korea) *(SenSys '15)*. Association for Computing Machinery, New York, NY, USA, 5–16. https://doi.org/10.1145/2809695.2809707

[38] 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* (Delft, Netherlands) *(SenSys '17)*. ACM, New York, NY, USA, Article 19, 13 pages. https://doi.org/10.1145/3131672.3131674

[39] Josiah Hester, Kevin Storer, and Jacob Sorber. 2017. Timely Execution on Intermittently Powered Batteryless Sensors. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems* (Delft, Netherlands) *(SenSys '17)*. ACM, New York, NY, USA, Article 17, 13 pages. https://doi.org/10.1145/3131672.3131673

[40] Himax Technologies, Inc. 2018. HM01B0 Ultra Low Power camera sensor. https://github.com/cjosephson/backcam/blob/master/hardware/datasheets/HM01B0_DS_preliminary_v06.pdf.

[41] Henry Hoffmann, Stelios Sidiroglou, Michael Carbin, Sasa Misailovic, Anant Agarwal, and Martin Rinard. 2011. Dynamic Knobs for Responsive Power-Aware Computing. In *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems* (Newport Beach, California, USA) *(ASPLOS XVI)*. Association for Computing Machinery, New York, NY, USA, 199–212. https://doi.org/10.1145/1950365.1950390

[42] HopeRF. 2019. RFM95W LoRa Module. https://www.hoperf.com/data/upload/portal/20190801/RFM95W-V2.0.pdf.

[43] Bashima Islam and Shahriar Nirjon. 2020. Scheduling Computational and Energy Harvesting Tasks in Deadline-Aware Intermittent Systems. In *2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. 95–109. https://doi.org/10.1109/RTAS48715.2020.00-14

[44] Bashima Islam and Shahriar Nirjon. 2020. Zygarde: Time-Sensitive On-Device Deep Inference and Adaptation on Intermittently-Powered Systems. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 3, Article 82 (sep 2020), 29 pages. https://doi.org/10.1145/3411808

[45] IXYS Corporation. [n.d.]. SM700K10L. https://ixapps.ixys.com/DataSheet/SM700K10L.pdf.

[46] Neal Jackson et al. 2019. Capacity over Capacitance for Reliable Energy Harvesting Sensors. In *Proceedings of the 18th International Conference on Information Processing in Sensor Networks* (Montreal, Quebec, Canada) *(IPSN '19)*. ACM, New York, NY, USA, 193–204. https://doi.org/10.1145/3302506.3310400

[47] H. Jayakumar, A. Raha, and V. Raghunathan. 2014. QUICKRECALL: A Low Overhead HW/SW Approach for Enabling Computations across Power Cycles in Transiently Powered Computers. In *2014 27th International Conference on VLSI Design and 2014 13th International Conference on Embedded Systems*. 330–335. https://doi.org/10.1109/VLSID.2014.63

[48] Hrishikesh Jayakumar, Arnab Raha, Jacob R. Stevens, and Vijay Raghunathan. 2017. Energy-Aware Memory Mapping for Hybrid FRAM-SRAM MCUs in Intermittently-Powered IoT Devices. *ACM Trans. Embed. Comput. Syst.* 16, 3, Article 65 (April 2017), 23 pages. https://doi.org/10.1145/2983628

[49] Vito Kortbeek, Kasim Sinan Yildirim, Abu Bakar, Jacob Sorber, Josiah Hester, and Przemysław Pawełczak. 2020. Time-Sensitive Intermittent Computing Meets Legacy Software. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems* (Lausanne, Switzerland) *(ASPLOS '20)*. Association for Computing Machinery, New York, NY, USA, 85–99. https://doi.org/10.1145/3373376.3378476

[50] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324. https://doi.org/10.1109/5.726791

[51] Benjamin C. Lee and David M. Brooks. 2006. Accurate and Efficient Regression Modeling for Microarchitectural Performance and Power Prediction. In *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems* (San Jose, California, USA) *(ASPLOS XII)*. Association for Computing Machinery, New York, NY, USA, 185–194. https://doi.org/10.1145/1168857.1168881

[52] C. Li, W. Zhang, C. Cho, and T. Li. 2011. SolarCore: Solar energy driven multi-core architecture power management. In *2011 IEEE 17th*

International Symposium on High Performance Computer Architecture. 205–216. https://doi.org/10.1109/HPCA.2011.5749729

[53] Ji Lin, Wei-Ming Chen, Han Cai, Chuang Gan, and Song Han. 2021. MCUNetV2: Memory-Efficient Patch-based Inference for Tiny Deep Learning. arXiv:cs.CV/2110.15352

[54] Ji Lin, Wei-Ming Chen, Yujun Lin, John Cohn, Chuang Gan, and Song Han. 2020. MCUNet: Tiny Deep Learning on IoT Devices. arXiv:cs.CV/2007.10319

[55] Ting Liu, Christopher M. Sadler, Pei Zhang, and Margaret Martonosi. 2004. Implementing Software on Resource-constrained Mobile Sensors: Experiences with Impala and ZebraNet. In *Proceedings of the 2Nd International Conference on Mobile Systems, Applications, and Services* (Boston, MA, USA) *(MobiSys '04)*. ACM, New York, NY, USA, 256–269. https://doi.org/10.1145/990064.990095

[56] Brandon Lucia and Benjamin Ransford. 2015. A Simpler, Safer Programming and Execution Model for Intermittent Systems. In *Proceedings of the Conference on Programming Language Design and Implementation* (Portland, OR, USA) *(PLDI '15)*. ACM, New York, NY, USA, 11. https://doi.org/10.1145/2737924.2737978

[57] Andrew Lukefahr, Shruti Padmanabha, Reetuparna Das, Faissal M. Sleiman, Ronald Dreslinski, Thomas F. Wenisch, and Scott Mahlke. 2012. Composite Cores: Pushing Heterogeneity Into a Core. In *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture* (Vancouver, B.C., CANADA) *(MICRO-45)*. IEEE Computer Society, USA, 317–328. https://doi.org/10.1109/MICRO.2012.37

[58] Kaisheng Ma, Xueqing Li, Jinyang Li, Yongpan Liu, Yuan Xie, Jack Sampson, Mahmut Taylan Kandemir, and Vijaykrishnan Narayanan. 2017. Incidental Computing on IoT Nonvolatile Processors. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture* (Cambridge, Massachusetts) *(MICRO-50 '17)*. ACM, New York, NY, USA, 204–218. https://doi.org/10.1145/3123939.3124533

[59] Kiwan Maeng, Alexei Colin, and Brandon Lucia. 2017. Alpaca: Intermittent Execution Without Checkpoints. *Proc. ACM Program. Lang.* 1, OOPSLA, Article 96 (Oct. 2017), 30 pages. https://doi.org/10.1145/3133920

[60] Kiwan Maeng and Brandon Lucia. 2018. Adaptive Dynamic Checkpointing for Safe Efficient Intermittent Computing. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. USENIX Association, Carlsbad, CA, 129–144. https://www.usenix.org/conference/osdi18/presentation/maeng

[61] Kiwan Maeng and Brandon Lucia. 2019. Supporting Peripherals in Intermittent Systems with Just-in-time Checkpoints. In *Proceedings of the Conference on Programming Language Design and Implementation* (Phoenix, AZ, USA) *(PLDI 2019)*. ACM, New York, NY, USA, 1101–1116. https://doi.org/10.1145/3314221.3314613

[62] 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* (London, UK) *(PLDI 2020)*. Association for Computing Machinery, New York, NY, USA, 1005–1021. https://doi.org/10.1145/3385412.3385998

[63] Andrea Maioli, Luca Mottola, Muhammad Hamad Alizai, and Junaid Haroon Siddiqui. 2021. Discovering the Hidden Anomalies of Intermittent Computing. In *Proceedings of the 2021 International Conference on Embedded Wireless Systems and Networks (EWSN '21)*.

[64] A. Mirhoseini, E. M. Songhori, and F. Koushanfar. 2013. Idetic: A high-level synthesis approach for enabling long computations on transiently-powered ASICs. In *2013 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. 216–224. https://doi.org/10.1109/PerCom.2013.6526735

[65] Moodstocks. 2016. jpec - a JPEG encoder in C. https://github.com/Moodstocks/jpec.

[66] S. Naderiparizi, A. N. Parks, Z. Kapetanovic, B. Ransford, and J. R. Smith. 2015. WISPCam: A battery-free RFID camera. In *2015 IEEE International Conference on RFID (RFID)*. 166–173. https://doi.org/10.1109/RFID.2015.7113088

[67] Sangmin Oh, Anthony Hoogs, Amitha Perera, Naresh Cuntoor, Chia-Chih Chen, Jong Taek Lee, Saurajit Mukherjee, J. K. Aggarwal, Hyungtae Lee, Larry Davis, Eran Swears, Xioyang Wang, Qiang Ji, Kishore Reddy, Mubarak Shah, Carl Vondrick, Hamed Pirsiavash, Deva Ramanan, Jenny Yuen, Antonio Torralba, Bi Song, Anesco Fong, Amit Roy-Chowdhury, and Mita Desai. 2011. A large-scale benchmark dataset for event recognition in surveillance video. In *CVPR 2011*. 3153–3160. https://doi.org/10.1109/CVPR.2011.5995586

[68] Shruti Padmanabha, Andrew Lukefahr, Reetuparna Das, and Scott Mahlke. 2013. Trace Based Phase Prediction for Tightly-Coupled Heterogeneous Cores. In *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture* (Davis, California) *(MICRO-46)*. Association for Computing Machinery, New York, NY, USA, 445–456. https://doi.org/10.1145/2540708.2540746

[69] pms67. 2020. PID controller implementation written in C. https://github.com/pms67/PID.

[70] Qoitech. [n.d.]. Otii Ace Pro. https://www.qoitech.com/otii-ace/.

[71] Amir Rahmati, Mastooreh Salajegheh, Dan Holcomb, Jacob Sorber, Wayne P. Burleson, and Kevin Fu. 2012. TARDIS: Time and Remanence Decay in SRAM to Implement Secure Protocols on Embedded Devices without Clocks. In *21st USENIX Security Symposium (USENIX Security 12)*. USENIX Association, Bellevue, WA, 221–236. https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/rahmati

[72] Benjamin Ransford, Jacob Sorber, and Kevin Fu. 2011. Mementos: System Support for Long-running Computation on RFID-scale Devices. In *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems* (Newport Beach, California, USA) *(ASPLOS XVI)*. ACM, New York, NY, USA, 159–170. https://doi.org/10.1145/1950365.1950386

[73] Emily Ruppel and Brandon Lucia. 2019. Transactional Concurrency Control for Intermittent, Energy-harvesting Computing Systems. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation* (Phoenix, AZ, USA) *(PLDI 2019)*. ACM, New York, NY, USA, 1085–1100. https://doi.org/10.1145/3314221.3314583

[74] 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)*. 318–335. https://doi.org/10.1109/MICRO56248.2022.00034

[75] Saleae. [n.d.]. Saleae Logic Pro 8. https://www.saleae.com/.

[76] Alanson P Sample, Daniel J Yeager, Pauline S Powledge, Alexander V Mamishev, and Joshua R Smith. 2008. Design of an RFID-based battery-free programmable sensing platform. *IEEE Transactions on Instrumentation and Measurement* 57, 11 (2008), 2608–2615.

[77] Joshua San Miguel, Karthik Ganesan, Mario Badr, Chunqiu Xia, Rose Li, Hsuan Hsiao, and Natalie Enright Jerger. 2018. The EH Model: Early Design Space Exploration of Intermittent Processor Architectures. In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 600–612. https://doi.org/10.1109/MICRO.2018.00055

[78] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2019. MobileNetV2: Inverted Residuals and Linear Bottlenecks. arXiv:cs.CV/1801.04381

[79] Siglent. [n.d.]. SPD1305X Series Programmable Linear DC Power Supply. https://siglentna.com/product/spd1305x/.

[80] Thannirmalai Somu Muthukaruppan, Anuj Pathania, and Tulika Mitra. 2014. Price Theory Based Power Management for Heterogeneous Multi-Cores. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems* (Salt Lake City, Utah, USA) *(ASPLOS '14)*. Association for Computing Machinery, New York, NY, USA, 161–176. https://doi.org/10.1145/2541940.2541974

[81] STMicroelectronics. [n.d.]. STM32G071RB. https://www.st.com/en/microcontrollers-microprocessors/stm32g071rb.html.

[82] Bo Su, Junli Gu, Li Shen, Wei Huang, Joseph L. Greathouse, and Zhiying Wang. 2014. PPEP: Online Performance, Power, and Energy Prediction Framework and DVFS Space Exploration. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture* (Cambridge, United Kingdom) *(MICRO-47)*. IEEE Computer Society, USA, 445–457. https://doi.org/10.1109/MICRO.2014.17

[83] Milijana Surbatovich, Limin Jia, and Brandon Lucia. 2019. I/O Dependent Idempotence Bugs in Intermittent Systems. *Proc. ACM Program. Lang.* 3, OOPSLA, Article 183 (Oct. 2019), 31 pages. https://doi.org/10.1145/3360609

[84] Milijana Surbatovich, Limin Jia, and Brandon Lucia. 2021. Automatically Enforcing Fresh and Consistent Inputs in Intermittent Systems. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation* (Virtual, Canada) *(PLDI 2021)*. Association for Computing Machinery, New York, NY, USA, 851–866. https://doi.org/10.1145/3453483.3454081

[85] Milijana Surbatovich, Naomi Spargo, Limin Jia, and Brandon Lucia. 2023. A Type System for Safe Intermittent Computing. *Proc. ACM Program. Lang.* 7, PLDI, Article 136 (jun 2023), 25 pages. https://doi.org/10.1145/3591250

[86] Texas Instruments. [n.d.]. MSP430FR5994. https://www.ti.com/product/MSP430FR5994.

[87] Texas Instruments. [n.d.]. TS5A3359. https://www.ti.com/product/TS5A3359.

[88] Texas Instruments. 2011. Ultra Low Power Boost Converter with Battery Management for Energy Harvester. https://www.ti.com/product/BQ25504.

[89] THORLABS. [n.d.]. Driver PID Settings. https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=9013.

[90] Joel Van Der Woude and Matthew Hicks. 2016. Intermittent Computation Without Hardware Support or Programmer Intervention. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation* (Savannah, GA, USA) *(OSDI'16)*. USENIX Association, Berkeley, CA, USA, 17–32. http://dl.acm.org/citation.cfm?id=3026877.3026880

[91] Hong Zhang, Jeremy Gummeson, Benjamin Ransford, and Kevin Fu. 2011. Moo: A batteryless computational RFID and sensing platform. *Department of Computer Science, University of Massachusetts Amherst., Tech. Rep* (2011).

[92] Huazhe Zhang and Henry Hoffmann. 2016. Maximizing Performance Under a Power Cap: A Comparison of Hardware, Software, and Hybrid Techniques. In *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems* (Atlanta, Georgia, USA) *(ASPLOS '16)*. Association for Computing Machinery, New York, NY, USA, 545–559. https://doi.org/10.1145/2872362.2872375