Continuous Sensing on Energy-harvesting Battery-less Sensors

Anonymous Author(s)

ABSTRACT

The main obstacles to achieve truly ubiquitous sensing are (i) the limitations of battery technology - batteries are short-lived, hazardous, bulky, and costly - and (ii) the unpredictability of ambient power. The latter causes sensors to operate intermittently, violating the availability requirements of many real-world applications.

In this paper, we present the *Coalesced Intermittent Sensor* (CIS), an intermittently powered "sensor" that senses continuously! Our key observation being that, the power cycles of energy-harvesting battery-less devices do not show correlation even when they are drawing energy from the same source, running the same application, and in close proximity. We challenged our observation using different real hardware and energy sources and showed that this observation holds.

Another important finding is that a CIS designed for certain (minimal) energy conditions requires no explicit spreading of awake times due to embedded randomness in the powering subsystem. However, when the available energy exceeds the design point, nodes employing a sleep mode (to extend their availability) do wake up collectively at the next event. This synchronization leads to problems as multiple responses will be generated, and -worse- subsequent events will be missed as nodes will now recharge at the same time. To counter this unwanted behavior we designed an algorithm to estimate the number of active neighbors and respond proportionally to an event. We show that when intermittent nodes randomize their responses to events, in favorable energy conditions, the CIS reduces the duplicated captured events by 50% and increases the percentage of capturing entire bursts above 90%.

CCS CONCEPTS

• **Human-centered computing** → *Empirical studies in ubiquitous and mobile computing*.

KEYWORDS

Embedded systems, Energy harvesting, Ubiquitous computing, Intermittent Systems

ACM Reference Format:

Anonymous Author(s). 2019. Continuous Sensing on Energy-harvesting Battery-less Sensors. In *Proceedings of ACM Conference (Conference'19).* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Conference'19, July 2019, New York, NY, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Batteries may compromise the viability of sensor nodes in various ways. Batteries are bulky, short-lived, hazardous, and expensive. To ameliorate the battery problem, researchers have been investigating different alternatives to extend lifetime and reduce costs and form factor. The reduction in power consumption of recent microcontrollers and the advances in energy-harvesting circuitry have enabled the emergence of battery-free energy-harvesting sensors. These sensors elide the constraints of batteries and extract power from ambient energy sources such as sunlight and RF emissions.

Ambient energy sources provide perpetual power. However, ambient power is usually too weak to directly power a sensor node [21]. Therefore, an energy-harvesting node first buffers the harvested energy until a usable amount has been accumulated; then it operates, for a short period of time, until the buffered energy has been exhausted [22]. Consequently, battery-less energy-harvest sensors operate intermittently (Figure 1).

Intermittent power introduces a set of new challenges that are under ongoing investigation. For example, [3, 8, 22, 30, 31] studied the intermittent computation problem, which is concerned with the preservation of application progress and data consistency under frequent power failures; Hester et al. [15] investigated the timely operation challenge, which is concerned with data freshness after a power interrupt; and Yıldırım et al. [41] introduced event-driven execution for the intermittent domain, which deals with input and output operations under arbitrarily-timed power loss.

Despite these notable advances, intermittently-powered sensors suffer from a new fundamental shortcoming: *the intermittent availability of the system.* Being frequently off charging compromises the value of these devices. For example, a sensor that has a low probability (e.g., 10% [**?**]) to be available (on) when an event of interest occurs has no value. Overcoming the intermittent availability challenge without changing the size of the device or re-including batteries requires a novel approach that explores new design dimensions.

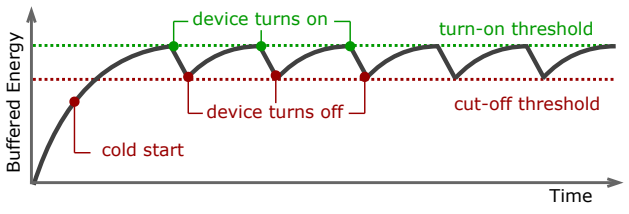


Figure 1: Ambient power is weak; therefore, it is usually buffered. The buffered energy is then consumed to operate the device. The operation period is often short as power consumption is much higher than the energy harvesting rate.

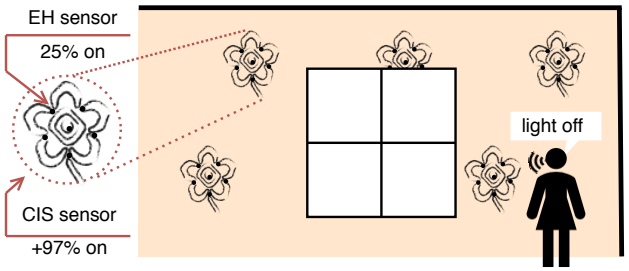


Figure 2: A Coalesced Intermittent Sensor (CIS) is a group of intermittently-powered nodes that sense continuously despite the intermittent power supply. CIS exploits the inherent randomization of energy harvesting systems, if available, and introduces artificial randomization, when needed, to preserve continuous sensing.

1.1 Vision and Applications

Battery-powered sensors are reliable, but they are short-lived; energy-harvesting sensor operate intermittently, but they are long-lived. Our goal is to combine the desirable features of these two type of sensors to create a new "sensor" that operates permanently (no batteries) and reliably (continuously available). Moreover, the form factors of such sensors should be small to facilitate embedding them in various objects.

Sensors with such characteristics would allow us to add a cheap and maintenance-free sensing layer to many objects, making them smart and interactive. For example, one can imagine developing smart wallpaper that users can interact with. Smart wallpaper with embedded microphones can enable direct in-building human-to-object communication (Figure 2). Such a permanently operating sensor can be deployed, for example, in kids' playgrounds to monitor their occupancy. These battery-less sensors can enable interactive and safe-to-dispose sports rugs (that count how many times a person has jumped on them) or play rugs for kids. In short, we would like to develop small sensors with permanent and continuous sensing capabilities.

1.2 Research Challenges

Many sensing applications require the sensor to be available when there is a change in the monitored environment. Energy-harvesting sensors can provide cheap and maintenance-free sensing, but the do not meet the availability requirements of many real-world applications.

C1-Approach continuous availability on intermittent power: An energy-harvesting battery-less sensor is frequently off, spending most of the time charging. One way to increase the system availability is by using more than one node, hoping that at least one is on at all times. However, only increasing the number of the sensors is an insufficient condition to increase their collective availability as their on/off cycles may become correlated—after all, they draw power from the same energy source and have the same characteristics—, and therefore, they will turn on and off at approximately the same time, compromising the overall availability of the system. As such, *the challenge to approach continuous availability given a certain number of intermittent devices with certain duty cycles is to break any emerging correlation between nodes' power cycles.* Answering this question would also allow us to specify the minimum number of nodes needed to achieve a certain collective duty cycle required by an application.

C2-Continuous sensing on intermittently powered sensors: Even when the collective availability of intermittent sensors approaches 100%, the emerging overall sensing behavior may still be intermittent. In event-based sensing applications, sensor nodes sleep in low-power mode to consume their buffered energy efficiently. In favorable energy conditions, the energy harvesting rates of these sensors may equal (or approximate) their power consumption rates at the sleeping mode, making the nodes available for an extended period of time. In such conditions, nodes do wake up collectively at the incoming external event and consequently exhaust their energy at roughly the same time. This unwanted synchronization compromises the overall availability of the system, as nodes will respond to the same events and (what is even worse) miss subsequent ones. In particular, this is a significant problem when events arrive in a bursty fashion (e.g., a command of a few words).

C3-Efficient sensing on intermittent sensors: One of the main factors that determine the intermittency pattern of an energy-harvesting battery-less sensor is the richness of ambient energy. For example, at mid-noon under direct sunlight, even a small solar panel can power a sensor node continuously. In such energy conditions (favorable energy conditions), using plenty of intermittent sensors would only result in duplicated work that leads to duplicated messages when these nodes communicate the processed data to a sink node: normally, a battery-powered node acts as a gateway for such sensors to communicate with other layers of the Internet of Things. These duplicated messages cause the sink node to consume power that we wanted to save in the first place (if we assume that the battery of the sink node needs to be replaced every y months when capturing an event generates a single message. And, there is a system that generates x messages for each captured event. Then, the time to replace the battery of the sink will be reduced to $\frac{y}{x}$ months).

1.3 Contributions

In this paper, we tackle the paradox of continuous sensing on intermittently-powered sensors. We studied the inter-relationship between the power cycles of energy-harvesting battery-less devices, the emerging collective behavior, and the effect of changes in ambient energy levels on the sensors collective emerging behavior.

In particular, this paper makes the following key contributions:

- We show how continuous availability of a group of intermittently-powered devices can be approached for different scenarios, i.e., explicit and implicit control of nodes' awake time. We **modeled** the collective emerging on-time (system availability) of intermittently-powered sensors and **validate** our model on real hardware and against different energy sources.

- We introduce a new type of virtual sensor, showing its capabilities and limitations. The **Coalesced Intermittent Sensor (CIS)** is the abstraction of a group of intermittently-powered sensors that achieves maximum statistical availability by exploiting (inherent) randomization to spread nodes' awake times uniformly.
- Contrary to common sense, we show how favorable energy conditions can deteriorate the performance of a CIS. We, therefore, equipped the CIS with **an new algorithm** that makes it ambient energy aware. This algorithm enables the nodes to determine their own duty cycles (without requiring additional hardware), and the average number of alive nodes (without requiring communication). This information can effectively be used by the nodes to decide when to back off to avoid duplicate event detection and availability interruptions (implicit synchronization in favorable harvesting conditions).
- We prototype, evaluate, and demonstrate the feasibility of the Coalesced Intermittent Sensor concept in the form of voice-control application recognizing individual words on solar-powered nodes equipped with a microphone.

2 BACKGROUND

Here, a minimal amount of background information is presented to clarify some of our design decisions.

2.1 Energy-harvesting Devices

Why miniturized sensors. A small sensor is a less intrusive device and compared to a bigger one; therefore, many applications prefer sensors of small form factors. Long sensors' lifetimes lead to less maintenance costs, and therefore, many applications demand sensors with long liftimes. However, long lifetime and small form factor are conflict goals. For example, rechargeable batteries paired with energy harvesters can continuously power sensor nodes for a relatively long time. But, they obviously increase nodes' sizes and costs. Additionally, rechargeable batteries wear out after a few hundreds of charging cycles, which means they also limit sensors' lifetimes [1]. However, if an application's requirements put hard constraints on the size of the sensors, then removing the batteries is one of the first options to be considered. Battery-less energy-harvesting sensors have the potential to operate for very long time (decades long) while being tiny; but, they operate intermittently. They charge a small capacitor to guarantee uninterrupted operations for a certain minimum duration. This guarantee facilitates the software architecture design. An intermittent program is usually decomposed into a chain of small atomic regions that are glued together with checkpoints [31].

Do big capacitors enable continuous operation?. Big capacitors allow longer operational periods, but they also need more time to charge. As such, the on/off duty cycle of a big or small capacitor may still be the same. Additionally, very big capacitors might violate the size requirements of an application. For example, the WISPCam—an RF powered camera— [26] uses a supercapacitor that is about 50 times the size of the original WISP's capacitor—a computational RFID tag— [34].

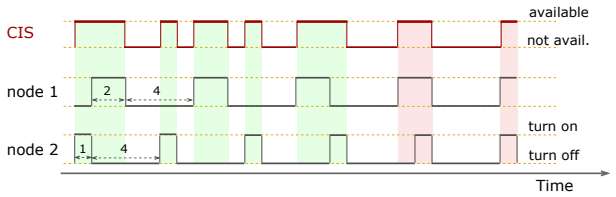


Figure 3: A Coalesced Intermittent Sensor's availability is the emerging collective on-time of its intermittent nodes' on-times. The difference between the power cycles leads to a constant relative shift between the nodes duty cycles. This, in turn, causes their on-times to be uniformly distributed on the overall power cycle. The red bars indicate a minimum CIS time span—CIS's nodes are overlapping—whereas the green bars show the maximum time span of the CIS.

[charging big capacitors vs small ones] [cold start of big capacitors vs small ones]

2.2 Speech types

Why isolated word speech type. Speech recognition algorithms can be classified based on the type of speech that they can recognize into *spontaneous speech, continuous speech, connected word,* and *isolated word* [12]. Systems with *continuous* or *spontaneous speech* recognition are the closest to natural speech but are the most difficult to create because they need special methods to detect words boundaries [12]. This is less the case for the *connected word* type, where a minimum pause between the words is required. The type with the least complexity is the *isolated word* type. It requires a period of silence on both sides of a spoken word and accepts only single words.

Voice is a natural way for the human to interact with small devices. However, implementing speech recognition on resources—memory, computation power, and energy—limited platforms is challenging, to say the least. Therefore, we attempt to recognize, with our command recognizer prototype, the simplest type of speech, isolated words.

3 COALESCED INTERMITTENT SENSOR

An energy-harvesting intermittent node frequently switches between off and on, charging energy and operating. We can characterize, from a time perspective, this charge-discharge cycle (or power cycle) using the following notation: (t_{on}, t_p), where t_{on} refers to a node's uptime interval, and $t_p := t_{on} + t_{off}$, where t_{off} refers to a node's charging time interval.

The Coalesced Intermittent Sensor (CIS) is the abstraction of a group of intermittent senor nodes seeking to mimic the continuous sensing availability characteristic of a battery-powered sensor. The design of CIS needs to consider three main aspects: (i) how the nodes' awake time is distributed, (ii) the effect of the environment on CIS's availability and what is needed to mitigate it, and (iii) the consequence of approaching continuous sensing availability in a discrete fashion.

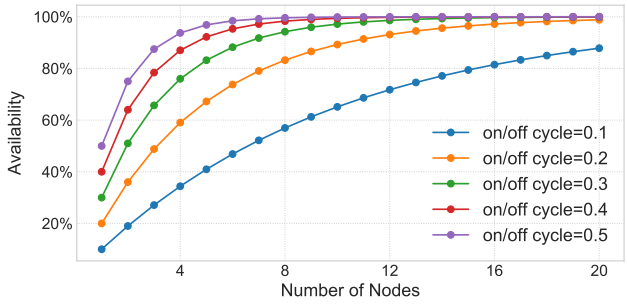


Figure 4: Coalesced Intermittent Sensor availability percentage for a different number of nodes and different duty cycles. The nodes are uniformly distributed and the CIS on-time evolves, when adding new nodes, according to the equation 1.

3.1 Coalesced availability

The CIS's on-time (availability) is the projection of its underlying intermittent nodes' on-times on the time axis. Thus, it can anywhere between a minimum (when all nodes' on-times cluster together) and a maximum (when the overlapping between on-times cannot be any smaller), depending on the nodes spreading strategy being employed. In general, we can imagine two broad spreading strategies:

Explicit on-time division strategy: Ultra-low-power timers have enabled intermittent nodes to time their t_{on} and t_{off} [15]. Recent advancements in passive light (or RF) communication have enabled nodes to efficiently backscatter messages on top of ambient light (or RF signals) [21**?**]. A CIS can build on top of these advancements to apply a time-division multiplexing strategy to minimize on-times overlapping. For example, a node calculates its average on-time $\overline{t_{on}}$ and off-time $\overline{t_{off}}$ for N power cycles. Then it measures the time difference between its power-up and the intended transmitting time Δt. Subsequently, it encodes the information $(\overline{t_{off}}, \overline{t_{on}}, \Delta t)$ in a message and broadcasts it. When a node receives this message it will have full knowledge about the transmitting node's power cycle. It can then alter its power cycle, relative to the transmitting nodes cycle, by either increasing (or decreasing) its power consumption to shorten (or lengthen) its on-time and subsequently shifting its power cycle to a different time slot.

With such explicit on-times control strategy, a CIS of N nodes with average $\overline{t_{on}}$ and off-time $\overline{t_{off}}$ will have an average availability of $N \times \overline{t_{on}}$ (or 100% if the total length of the on-times exceeds the nodes longest power cycle).

Implicit on-time division strategy: With no information being exchanged between intermittent nodes, the best CIS can do is to uniformly distribute its node's on-times and maintaining this distribution over time. The key observation to approach uniform distribution is to ensure that the node's power cycles are randomized.

Idealized power cycles. Let us assume that we have two nodes with ideal power cycles characterized by (t_{on}, t_p), and they have the same initial conditions. The collective on-time of these two nodes will still be t_{on} as they are in perfect synchronization (the two nodes wake up and power down together). To desynchronize their awake times a node should alter its power consumption, by sleeping for example, to shift its on-time to a different time slot. If we relax our assumptions, however, saying that the initial conditions are unknown, then shifting the on-time a constant number of times may cause the initially desynchronized nodes to be synchronized, reducing the CIS's availability instead of extending it. However, if the node constantly extends its on-time (and subsequently its power cycle), then its on-time will shift over the entire power cycle of the other node, spending $\frac{t_{off}}{t_p}$ of the time overlapping with off-time and $\frac{t_{on}}{t_p}$ overlapping with the on-time of the other node. This behavior is illustrated in Figure 3, where node 1 has a power cycle of (2,6) and node 2 has (1,5) power cycle. Following the time axis from the left to the right, we can see that the position of the on-time of node 2 is shifted by 1 unit of time after each power cycle of node 2. This implies that the on-times of the two nodes are $\frac{1}{3}$ of the time cluster together and $\frac{2}{3}$ of the time they are apart. If we extend the previous scenario to three or more nodes then the on-time of the resulting CIS can be described with the following formula,

$$t_{on}(N) = t_{on}(N-1) + \frac{t_{off}(N-1)}{t_{off}(N-1) + t_{on}(N-1)} \times t_{on}(1), \qquad (1)$$

where $N \in \mathbb{N}$ and $t_{on}(N)$ is the on-time of a CIS with N intermittent nodes. For the initial case where $N = 1$ we define $t_{on}(0) := 0$ and $t_{off}(0) := 1$.

In addition to characterizing the availability of a CIS, equation 1 also states that the benefit of adding a node to the CIS is proportional to the CIS's off-time. In Figure 4 CIS availability percentage for different duty cycles and different number of intermittent nodes are shown.

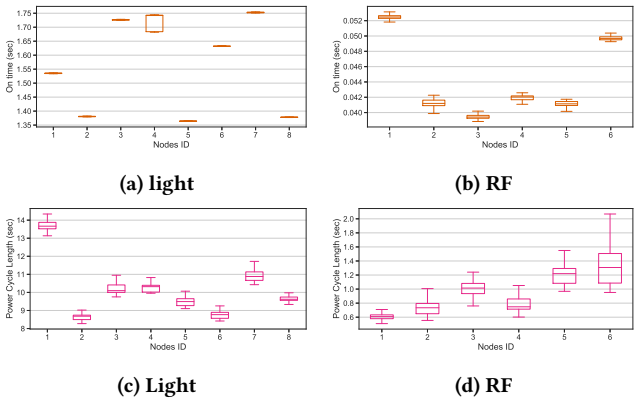
Natural power cycles. In reality, intermittent nodes power cycles are different, and they are not constant (Figure 5). However, differences between the power cycles and their embedded randomness are sufficient conditions to force their on-times to be in constant shift relative to each other and approach uniform distribution (Figure 1).

There is a clear trade-off between the aforementioned methods. While the explicit control method provides fine control over the system distribution and therefore requires less number of nodes than the implicit control method, the implicit control method does not depend on the ability to communicate between the nodes and therefore it is simpler and more energy efficient. Since inter-node communication is beyond the capabilities of most of today's intermittent nodes, **we focus on the implicit approach**.

3.2 Events classification

[improve] Given the unique characteristics of CIS, it is important to classify the events, identifying the categories that CIS can monitor. Form CIS's perspective, we can classify the events into three categories:

- *Short events*: are events that can be captured using single intermittent node. For example, a spoken word can be seen as a short event if the energy needed to recorded is less than what the energy buffer, i.e., the capacitor, of the intermittent sensor can store.



(a) light (b) RF

(c) Light (d) RF

Figure 5: nodes' power cycles length for different energy sources, and different energy buffer sizes.

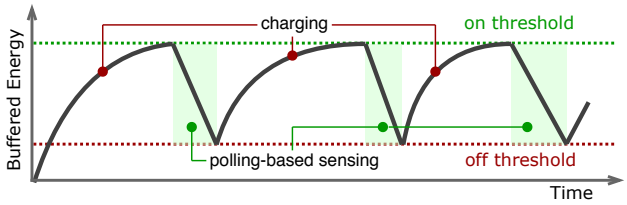
- *Long events*: are events that need more energy to be completely captured than what the energy buffer can store. Long events can be subdivided into two types:
  - *complex*: is a long event and capturing part is insufficient to extract the embedded information. For example, a spoken word can be seen as a long complex event if its recording requires more energy than what the energy buffer can store.
  - *simple* is a long event and capturing part of it is sufficient to extract the embedded information. For example, a rotating part of an engine that produce a particular sound.

3.3 Energy consumption

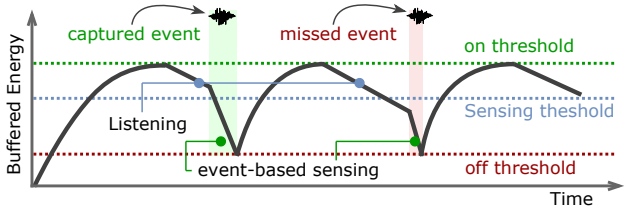
[consider remove Figure 6] An intermittent sensor has a limited energy budget per power cycle. When it is tasked with a polling-based sensing activity, its energy consumption, generally, switches between two levels: zero when charging and maximum when it activates its microcontroller for data acquisition and processing, see Figure 6a. (Note that we assume that the microcontroller is the dominant energy consumer module of a node.) However, in event-based sensing, a node puts its microcontroller into low-power mode and waits (or listens) for an external event to wake up the microcontroller. For example, in our prototype, the command recognizer, we exploit the microphone's wake-on-sound feature to send an interrupt to the microcontroller, which will then start recording the sound samples from the microphone. This wake-on-event style of operation is important as the minimal energy consumption during sleep significantly prolongs the period during which an event can be handled (for example, our prototype consumes 7 times less energy at sleep mode as compared to its active mode power consumption), see Figure 6b. This distinction between the energy consumption pattern of event-based and polling-based sensing will have an important consequence on how we model the probability of successful event capturing in these two different sensing strategy.

3.4 Power States

A CIS can experience a wide range of ambient power intensities. For example, a solar-powered CIS may harvest no energy at night,



(a) When CIS does polling-based sensing, its energy consumption profile has, generally, two distinct rates: zero when it is charging, and a maximum when it is sensing.



(b) When CIS does event-based sensing, it stays in low-power mode waiting for an external event to wake up the node. Consequently, it has three distinct energy consumption rates: zero when it is charging, a maximum when it is sensing, and an in-between when it is sleeping. In favorable energy conditions, the sleeping mode may cause intermittent nodes to synchronize their power cycles on external events and miss the next ones.

Figure 6: Coalesced Intermittent Sensor (CIS) energy profile for different sensing strategies. Green bars highlight successful sensing operations while the red bar shows a failed sensing attempt due to insufficient buffered energy.

modest energy from artificial light, and abundant energy from direct sunlight. Generally, we can identify four different CIS powering states:

- *Targeted power state*—These are the powering conditions that the CIS is designed for. In these conditions, the CIS should work intermittently and have sufficiently randomized power cycles to uniformly distribute its intermittent nodes on-times and meet the desired availability percentage (Figure 4). In general, the targeted powering conditions should be near worst energy harvesting conditions to ensure that the system is properly functioning for the majority of the time.
- *Under-targeted power state*—Ultimately, the ambient energy is an uncontrollable power source, and it is not hard to imagine scenarios where a CIS will be under-powered or even comes to complete and long power down (for example, a solar CIS will come to a perpetual power down in the darkness). In general, for under-targeted energy conditions, the CIS behavior can be considered as undefined.
- *Hibernating power state*—In event-based sensing scenarios, the intermittent nodes of a CIS sleep in low-power mode waiting for an external event to wake them up. If the energy conditions are relatively higher than the targeted conditions,

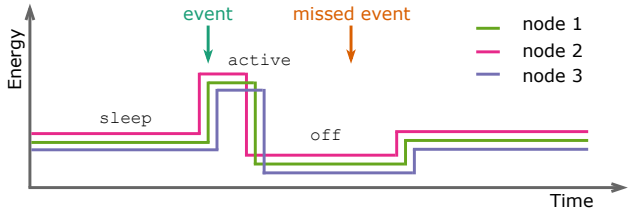


Figure 7: Coalesced Intermittent Sensor is in a hibernating power state when the energy harvesting rate approximates the energy consumption rate at the sleeping (or low-power) mode. In this state, the intermittent nodes lose the randomization in their power cycles. Thus, all the nodes capture the same event and power down shortly after missing the subsequent ones. Consequently, the CIS senses intermittently and does not take advantage of its redundant intermittent nodes to approach continuous sensing.

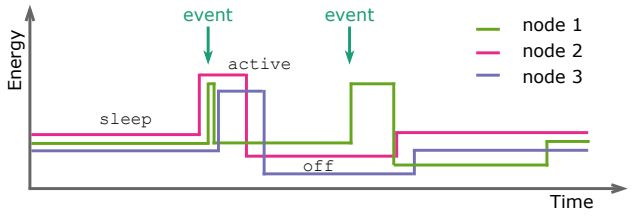


Figure 8: Randomized response helps in mitigating the hibernate-power-state problem. Also, it reduces the number of duplicated captured events when the CIS is overpowered. However, effective randomization strategy must be energy aware.

the nodes may not die and sustain their sleeping power consumption. This will cause them to synchronize their wake-ups on the first incoming event and their power down as the event capturing process depletes their energy buffers quickly. Consequently, the CIS may miss the next incoming events (specially if the events happens to arrive in bursts) causing it to sense intermittently instead of continuously, see Figure 7.

- *Continuous power state*—Under direct mid-noon sun even a tiny solar panel can continuously power a sensor. In such conditions, the CIS will sense continuously without the need for randomization. Therefore, the job of a single node will be repeated N times, and instead of sending a single message to a battery-powered or tethered sink—to push the data to the internet—N identical messages will be sent which waste a lot of energy.

The inefficiencies highlighted in the hibernating and continuous power states can be mitigated by enforcing randomization on the response of intermittent nodes (Figure 8): when a node is woken up by an external event it responds to that event with a certain probability. However, if the randomized response is enforced all the time, then the CIS will have a lower probability of catching

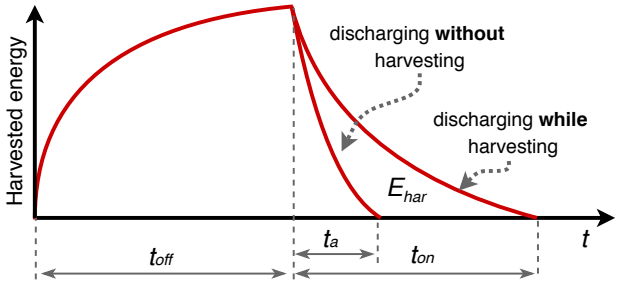


Figure 9: ....

events during the targeted energy conditions. Therefore, the CIS has to distinguish between the targeted and above-targeted energy conditions and randomize its response only during the hibernating and continuous power states.

Choosing a fix response probability is an inefficient way of dealing with the over-powering problem as the number of active intermittent nodes at a given moment is a function of the total number of intermittent nodes and the power intensity at that time. Therefore, efficient randomization requires intermittent nodes to estimate the number of active nodes at the moment of an external event arrival (which is discuss it next) and respond proportionally.

3.5 Intermittent Timing

Timing is a key building block of sensing systems. However, it is missing on intermittent nodes unless an additional dedicated timer circuit is added to them [15]. Here we would like to propose an alternative way that does not require additional timer hardware. Obviously, the on-time of an intermittent node can be measured using the microcontroller's built-in timers. However, the difficulty is *how an intermittent node can time its own off-time?*. Actually, answering this question does not only enable timing on intermittent devices but also enables them to estimate the richness of the ambient energy.

Timing the death. Algorithm 1 shows how a node can estimate its off-time. The basic idea is that a node measures its on-time while harvesting (Line 9) and compares it to the time required to drain the super capacitor without charging. The additional on-time Δt is the result of the energy accumulated while executing. (Line 12). By assuming a relatively stable charging rate, a node can calculate how long it will be off charging (Line 13-15). Obviously, in order for the time estimation to be correct, the reference time and the on-time measurement must be done with same load (a).

3.6 Nodes overlapping

In order for a node to estimate the number of active nodes at a given moment, first, it has to know the total number of nodes (N) in its CIS, which we assume to be known to the nodes before deployment. Second, this analysis is built on the observation that a node's on-time is a good indicator about the on-times of other nodes in the CIS, see Figure 10. A node can measure its on-time t_{on} and off-time t_{off} using Algorithm 1 (or an external dedicated timer [15]). Then, it can estimate the maximum time span (t_{max}) of its CIS, which is

Algorithm 1 off-time estimation

```
1:  f_REBOOT(u) = u++                          ▷ power reboot counter
2:  i ← f_REBOOT(i)                            ▷ i is a persistent variable
3:  E_buf                                      ▷ Size of energy buffer
4:  t_a                  ▷ time of discharging E_buf at load a, no harvesting
5:  t_i ← x                                    ▷ timing every x power cycles
6:  if (i mod t_i) = 0 then
7:      i = 0
8:      f_LOAD(a)                              ▷ set node load to a
9:      t_on ← t_PERS()                        ▷ persistent infinite loop
10: end if
11: if i = 0 then
12:     Δt = t_on − t_a                        ▷ time difference due to charging
13:     E_har ← (E_buf ÷ t_a) × Δt             ▷ harvested energy
14:     P_in ← E_har ÷ t_on                    ▷ incoming power
15:     t_off ← E_buf ÷ P_in
16: end if
```

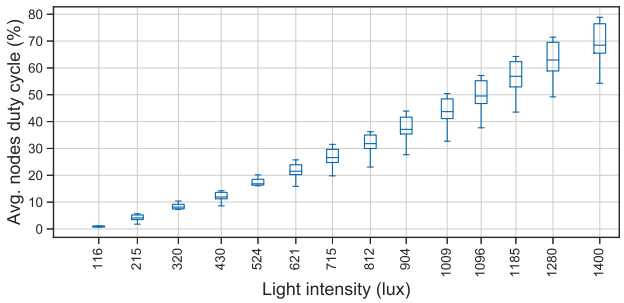


Figure 10: The average duty cycles of eight intermittently powered nodes for different light intensity. In general, an individual node's duty cycle is a good indicator of the average duty cycle of its CIS.

Table 1: Measuring intermittent nodes overlapping of a CIS of 8 intermittent nodes for different light intensities.

light intensity (lux)	mean	std
300	1.01	0.85
500	1.63	0.98
800	2.88	1.50
1200	5.05	1.08

the total duration of the nodes' on-times when they are aligned next to each other, as follows

$$t_{\max} = N \times t_{\text{on}}. \tag{2}$$

Then, from Equation 1, the node calculates the CIS expected on-time ($t_{\text{on}}(N)$). As we argued in Section 3.1, nodes on-times are uniformly distributed over the CIS power cycle. Thus, the overlapping on-time is also uniformly distributed over the CIS on-time. Then, a node can calculate the average number of active intermittent nodes n_{active} using the following formula,

$$n_{\text{active}} = t_{\max} \div t_{\text{on}}(N). \tag{3}$$

and choose the proper randomization factor. If a second event, however, happens shortly after the first one, a node needs to update N as follows,

$$N = N - n_{\text{active}} - 1$$

the -1 is because the node itself decided not to react on the first event.

Table 1 shows the average number of overlaps of an 8-nodes CIS for different light intensities. These measurements validate that nodes overlapping time is uniformly distributed over the CIS on-time. For example, at 1200 lux an individual node of our CIS has a duty cycle of $\approx 62\%$. If we multiply it by the number of nodes (Equation 2) we get about 500%. Figure 4 indicates that a CIS with eight nodes of duty cycles above 50% has near 100% availability. From equation 3, we find that the expected number of clustered nodes is 5 which is what Table 1 also shows.

3.7 Successful Detection Probability

Approaching continuous availability does not mean that CIS can successfully capture all events. It can happen that an event is being only partially captured by one or more nodes. This may lead to unsuccessful event detection. The important question is what is the probability of successfully capture an event give particular CIS characteristics. First, let us start with the simple case, a single intermittent node and a single event. Let us assume that a node has a power cycle that spans t_{p} unit of time, and the node is on for t_{on} unit of time. Thus, the percentage of the time the node is on is $\frac{t_{\text{on}}}{t_{\text{p}}}$ Further, we do not assume any prior knowledge about the event arrival time. Therefore, the probability of the event arrives at any moment in time is the same (mathematically, the event arrival time is a random variable drawn from uniform distribution). Let us say the event duration is t_{e}, and it is given that it will happen in this power cycles [1] Consequently, we can compute the probability of successful event capturing as follows:

$$P_c = \frac{t_{\text{on}}}{t_{\text{p}}} \times \frac{t_{\text{e}}}{t_{\text{p}}} \times \frac{t_{\text{on}}}{t_{\text{e}}}$$

Since the intermittent nodes are

4 IMPLEMENTATION—COALESCED INTERMITTENT COMMAND RECOGNIZER

We have developed a prototype of a coalesced intermittent command recognizer (CICR): an instant of a Coalesced Intermittent Sensor. The CICR consists of eight batteryless intermittent nodes. Each node is capable of performing isolated words recognition.

The reason behind developing a CICR is threefold: (i) voice is a natural and convenient way for human to interact with miniaturized devices; (ii) demonstrating *the world's first* batteryless intermittently-powered command recognizer, which shades light on the potential of batteryless intermittent systems; and (iii) facilitating testing with different sensing strategies and different type of external events arrival (i.e., regular or burst).

[1] Note that, an intermittent node's power cycle is repeating indefinitely; thus, when an event happens it will at least partially be within one power cycle. We are focusing on this power cycle

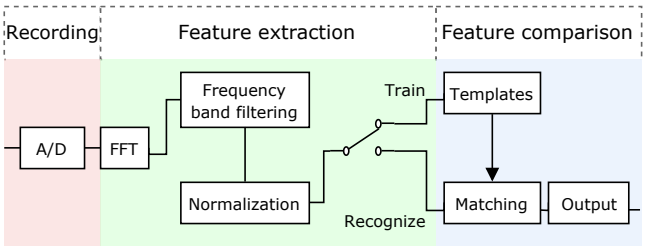


Figure 11: Coalesced Intermittent Command Recognizer: an instant of a Coalesced Intermittent Sensor. CICR features a power failure immune word recognition algorithm. First a word is recorded. Then, its spectral features are extracted. The resulting features vector is compared against previously-stored words' templates for recognition. The comparison is done using a liner distance matching algorithm

4.1 Hardware

A CICR node consists of thee main parts: a microphone, a microcontroller, and a harvester. MSP430RF5994 [37], an ultra-low-power microcontroller, is used for data acquisition and processing. This microcontroller has a 16-bit RISC processor running on 1 MHz, 8KB of SRAM (volatile), 256KB of FRAM (non-volatile), and a 12-bit analog to digital converter (ADC). It also features a Low Energy Accelerator (LEA), which offloads the main CPU for specific operations, such as FFT. For recording we use the PMM-3738-VM1010-R piezoelectric MEMS microphone, which features Wake on Sound and ZeroPower listening technologies [29], allowing both the microcontroller and the microphone to sleep in a low-power mode until a sound wave is detected. The microcontroller and microphone are powered by a BQ25570 solar power harvester [36] connected to an IXYS SLMD121H04L solar cell [18] and a super-capacitor of 470 µF. For debugging we used the Saleae logic analyzer [33].

4.2 System Description

The CICR has a power interrupts immune command recognizer. The recognizer is capable of recognizing isolated-word type of speech. The main parts of the recognizer are illustrated in Figure 11 and explained below:

4.2.1 Data acquisition. The *Wake-on-Sound* feature of the microphone triggers the data acquisition process once the energy level in the sound signal crosses a certain level. The ADC, then, samples the output of the microphone at 8 kHz. This sampling rate is sufficient to cover most of the frequency range of the human voice. To determine the end of the recording we relied on the characteristics of the targeted vocabulary. In particular, we identified experimentally the minimum effective recording length, which is 285 ms for the chosen set of words. By exploiting the Wake-on-Sound feature and using the minimum effective recording length, we eliminate the need for an endpoint detection algorithm, greatly improving the processing time and system efficiency from the energy perspective.

4.2.2 Feature Extraction. Once a recording has finished, framing and data processing begin. CICR divides the digitized signal into non-overlapping frames of 256 samples (≈ 33 milliseconds). This size is beneficial for doing a Fast Fourier Transform and short enough for the voice-features to be considered constant inside a frame.

Table 2: Profiling of features matching algorithms: Dynamic Time Warping (DTW) and Linear Distance Matching (LDM).

Section	LDM (ms)	DTW (ms)
Recording	285	285
Feature extraction	501	501
Feature matching	99	1251
Total	885	2037

To extract the spectral features of a frame, CICR divides the frequency of interest into 12 bands (as in [17]). The first five bands have a bandwidth of 200 Hz. The next three have a bandwidth of 300 Hz which are followed by two bands of 500 Hz. Finally, the last two bands have a 600 Hz bandwidth. This division is motivated by how the energy is concentrated in human speech [17]. Then, CICR computes the 256-point Fast Fourier Transform for each frame. The resulting feature vector contains the amount of energy concentrated in each frequency band defined earlier. This feature vector forms the basis for the words identifying process once it is normalized.

We normalize the feature vectors to reduce detection errors that result from differences in the amplitude of the speech input. To normalize a feature vector, CICR computes the binary logarithm of each entry of that vector. Then it computes the mean of the resulting vector. Finally, it subtracts the computed mean from each entry of the resulting vector. This is summarized in the following equation:

$$f_i = \log(\hat{f}_i) - \frac{\sum_{i=1}^{S} \log(\hat{f}_i)}{S}, \tag{4}$$

where f_i is the normalized output for the i^{th} spectral band of a feature vector, $\hat{f}_i$ is the energy in the i^{th} spectral band of the frame, and S is the number of spectral bands (12 in our case).

4.2.3 Feature Matching. Feature matching is achieved by computing the distances between the normalized feature vectors of the recorded word and the feature vectors of the words stored during the training phase (templates). CICR computes the squared Euclidean distance between vectors as follows:

$$d_j = \sum_{i=1}^{S} (f_{s,i} - f_{r,i})^2, \tag{5}$$

where d_j is the distance between the j^{th} stored and recorded vectors. $f_{s,i}$ is the normalized output of the i^{th} spectral band of a stored vector, $f_{r,i}$ is the normalized output of the i^{th} spectral band of a recorded vector. The total distance between two words is calculated as follows:

$$D_k = \sum_{j=1}^{l} d(j) \tag{6}$$

where D_k is the distance between the k^{th} stored word and the recorded word, and l is the recording length measured in frames.

Table 3: Code statistics: lines of code

Language	Files	Blank	Comment	Code
C	7	264	173	736
C/C++ Header	8	62	40	237
Total	15	326	213	973

Table 4: Power usage

Section	Current (μA)	Voltage (V)	Power (μW)
Sleeping	64 ±20	2.008	128 ±40
Recording	423 ±20	2.008	849 ±40
Processing	282 ±20	2.008	566 ±40

Once the recorded word has been compared to all CICR template words, the template with the smallest distance to the recorded word is considered the correct word. However, if the smallest distance is bigger the garbage threshold which we experimentally set, then the CICR will return "undefined word".

It should be emphasized that in the linear distance matching algorithm (LDM) the feature vectors of two words are compared successively, not accounting for differences in pronunciation speed. This is sufficient for our case as we are targeting isolated words and speaker dependent speech recognition type. We also implemented the Dynamic Time Warping algorithm which better handles the difference in the speed of speech. However, it is slower than the linear matching algorithm (Table 2) and the detection accuracy was comparable in our case. Therefore, we default our implementation to LDM.

4.2.4 Power Failure Protection. [consider adding Coala] In order to preserve the progress state and to protect CICR data against randomly timed power failures, we manually split CICR program into 19 atomic regions. We ensured the each of these regions requires less energy than what the energy buffer can provide with a single charge. The program progress state is saved in the non-volatile memory (FRAM) on the transition between these regions. This prevents the program from falling back to its starting point (`main()`) after each power failure. Data in the non-volatile memory with Write-After-Read dependency is double buffered to ensure data integrity when the power supply is interrupted.

4.3 Code profiling

The entire command recognition software was written in the `C` programming language. The total program consists of 973 lines of code, excluding the FFT function from the Texas Instrument DSP library. See Table 3 for more information.

The memory footprint on the microcontroller is 20,064 bytes of FRAM and 1,134 bytes of SRAM. Execution times are shown in Table 2.

The power usage of a node differs according to it's activity. When a node is waiting for a voice event, it is in low-power mode. When data needs to be processed or recorded it is in active mode. When recording, the microphone and ADC consume additional power. The power consumption rates are determined by measuring the current with a Monsoon power monitor [25] and shown in Table 4.

5 EVALUATION

To evaluate the performance of the Coalesced Intermittent Sensor, we conducted several experiments in different energy conditions and with different types of events.

5.1 Experiment setup

5.2 system availability

Figure 12 shows the availability of CISs when they are powered by different energy source and for a different number of intermittent nodes. We see that (i) the energy sources (sunlight, artificial light, and RF) power the CIS intermittently, (ii) the CIS availability increases with number of nodes, and (iii) this addition is proportional to the CIS off-time.

The dashed lines represent Equation 1 expectation about the CIS availability for certain power cycles (10% for the RF powered system, and 15% for the light powered one). By comparing these lines to the measured ones we can conclude that these energy sources provide sufficient randomness to cause each node to have a slightly different power cycle which, in turn, causes their uptimes to be uniformly distributed in time.

availability on a fine scale. [Because of the differences in intermittent nodes power cycles, their duty cycles are constantly shifting relative to each other (Figure 3). However, this study does zoom in on the CIS availability on short time scale and the effect of length of the differences between the nodes power cycles on the system short term availability.]

After validating our observation on natural light and office artificial light, we designed a testbed with controllable light intensity for clarity and results reproducible. To this end, we blocked uncontrollable light sources with a box of 60×40 cm. On the ceiling of the box, we attached a light strip of 2.5 m with 150 LEDs that can produce 15 different light intensities. On the bottom of the box, we placed a coalesced intermittent command recognizer of 8 intermittent nodes (the hardware is described in Section 4.1).

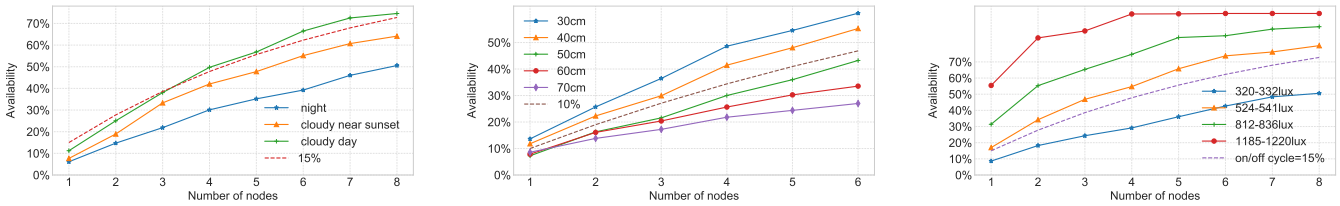
The events in our experiments are spoken words (Table 6). We recorded different patterns of isolated words to emulate the arrival of bursts or individual events with varying inter-event and inter-bust timing. We used a Bluetooth speaker [19] to replay a certain record. The data were collected using logic anaylyzer [33] and processed on a laptop running Ubuntu 16.04 LTS.

5.3 Events detection rate

These experiments show the total number of detected events and the number of uniquely detected ones with and without randomization. Also, they were conducted for a different type of events.

Regular events. Figure 13 shows the percentage of the total captured events and the uniquely captured ones. In this experiment we vary the light intensity from 300 lux to 1400 lux and the inter-event time from 1 sec to 6 sec.

We clearly see a positive correlation between light intensity and the number of detected events. In particular, we see that the number of duplicated detected events rises dramatically when light intensity increases, demonstrating the overpowering problem. Moreover, increasing the inter-event arrival time also surges the number of duplicated events. The reason for this phenomenon is that when the



(a) The system is powered by uncontrollable light sources—artificial light during the night, and sunlight during the day.

(b) The system is powered by radio frequency waves (RF). The RF source is located 30-70 cm aware from the RF tags (WISPs).

(c) The system is powered by radio frequency waves (RF). The RF source is located 30-70 cm aware from the RF tags (WISPs).

Figure 12: Measuring Coalesced Intermittent Sensor availability for a differed number of intermittent nodes. Generally, adding a node increases the system availability. This increment, however, is proportional to the CIS off-time.

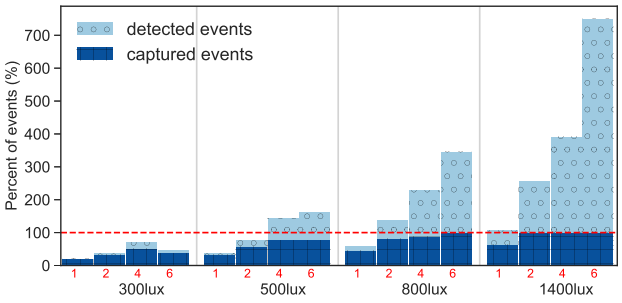


Figure 13: The number of detected and captured events by the coalesced intermittent command recognizer with eight intermittent nodes. The total number of external events is 240. In general, we see that when the light intensity increases, the number of detected and captured events rise too. Moreover, there is a positive correlation between the length of the inter-event arrival time and the detection and capture rates. Red numbers indicate events arrival interval.

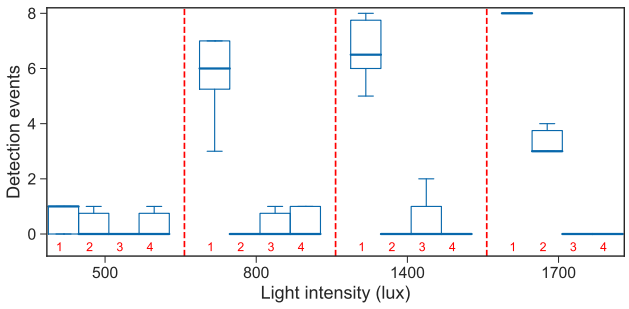


Figure 14: When capturing a burst of events without randomized response, the majority of the nodes react to the first event of a burst and power down, missing the rest of the burst. Red numbers indicate events index in a burst.

time between events increases, the intermittent nodes sleep longer in low-power mode, and this reduces the inherent randomization

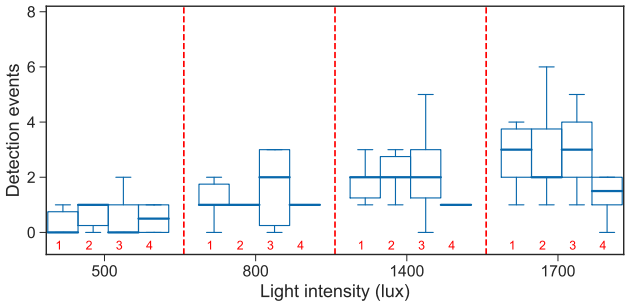


Figure 15: Response randomization enables a CIS to capture the entire burst of events with high capturing rates. It also reduces the number of duplicated events. Red numbers indicate events index in a burst.

of the intermittent nodes and leads them to the *hibernating power state* (Section 3.4).

Indirectly, these results show how a CIS can achieve a much higher duty cycle than its individual intermittent nodes—Figure 10 shows that with a light intensity of 800 lux an intermittent node is active with a duty cycle of 30% while Figure 13 shows that a CIS of 8 nodes captures 100% of the unique events when the time between them is 6 s.

Bursty events. Figure 14 shows the capturing behavior of a CIS when the events arrive in bursts. A burst of four events with one second between the individual events was fired every 20 seconds. Each burst was repeated 10 times and for four different light intensities. The nodes sleep in low-power mode when they finish processing, waiting for the next event.

In general, we observe that the intermittent nodes react to the first event of a burst and power down shortly after missing other events in the burst. This results validate our theory about the side effect of the *hibernating power state* (Section 3.4). These results also demonstrate the hibernating power problem on a wide range of power intensities, showing the significance of this problem. Next, we will show how randomized response can mitigate these problems.

(lux, sec)	(800, 6)	(1400, 4)	(1400, 6)
randomization	205/432	236/675	223/493
no randomization	240/831	240/938	240/1802

Table 5: Randomized response reduces the number of duplicated detected events, when the CIS is overpowered, by 50% while losing only 7% of the unique events. The results are presented in the following format *unique/total* detected events.

5.4 Events detection rate with randomization

Regular events. Table 5 shows the number of detected events for three different scenarios. We see that randomized response reduces duplicated events by an average of ≈50%, while only marginally lowers the number of the uniquely detected events. The intermittent nodes were responding to events with a probability of 65% for the scenario of 800 lux and 6 seconds arrival time and the scenario of 1400 lux and 4 seconds arrival time. However, for the highest energy level and the longest inter-event arrival time a responding probability of 30% was used.

Bursty events. Figure 15 shows that a CIS with randomized response spreads its resources–as compared to Figure 14–and captures the entire burst with a probability of above 90%. We also observe a positive impact of randomized response when the system is under-powered (500 lux).

To randomize during bursty events, a node reacts with a certain probability on a event. This probability is different for each event since the node become active after the last recharge. In order to spread the nodes over the events, the probabilities need to increase for subsequent events, since some nodes have reacted already on previous events, and therefore the number of nodes still available is smaller after each event. A node reacts with a probability of 40% on the first event, with 50% on the second event, 70% on the third event and 100% on the fourth event.

5.5 Coalesced intermittent command recognizer word detection accuracy

Table 6: Testing set

on	off	stop	clear	load
go	pause	resume	edit	cancel

For evaluating the coalesced intermittent command recognizer accuracy, we used the word set in Table 6. Each word was pronounced by a single speaker 20 times and recorded on a PC. One of these recordings was stored as a template on the CICR, while the remaining 19 were played back through a Bluetooth speaker [29] for testing.

The ratio between detected events and successfully recognized events per node is shown in Figure 16 and it averages out at 76.7%. The difference between detection and capture is primarily caused by nodes that have insufficient buffered energy to finish recording.

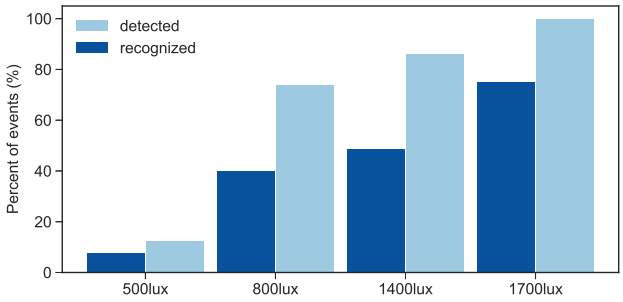


Figure 16: Average number of successfully recognized words per node and average number of detected words per node, as percentages of the total number of played words. Words are relatively long events and therefore some of their recordings do not complete due to insufficient harvested energy.

6 RELATED WORK AND BACKGROUND

Recent advances in ultra-low-power microcontrollers along with the development of energy harvesters have enabled the creation of stand-alone battery-free sensors. Ambient energy that powers these sensors is volatile. Thus, these sensors operate intermittently.

6.1 Energy-harvesting systems

Ideally, energy harvesters power devices indefinitely as they collect energy from perpetual energy sources. Sunlight, vibration, and radio frequency (RF) waves are examples of such energy sources. The power harvested from these sources vary wildly, for example, RF harvestable power ranges from nW-scale when harvested from ambient signals to μW-scale when collected form a dedicated RF signal emitter, and solar power varies from tens of μW to tens of mW when it is harvested by a solar panels of a few cm^2 illumination surface [22, 32].

Many battery-less energy-harvesting platforms have been proposed. Some of them rely on dedicated external energy sources such as WISP -and its variants-, a general wireless sensing and identification platform [34, 42, 44]; WISPcam, an RF powered camera [26] and, the battery-free cellphone [35]. Others, harvest from ambient sources such as the ambient backscatter tag [21], and the solar-powered tag [24]. Other platforms that facilitate the development of batteryless energy-harvesting systems have also been proposed. For instance, Fliker [14], a prototyping platform for batteryless devices; EDB [7] an energy-interference-free debugger for intermittent devices; and Capybara [9], a re-configurable energy storage architecture for energy-harvesting devices.

However, there is no energy-harvesting platform that considers the abstraction of many intermittent sensors (or nodes) and exploits the statistical energy harvesting differences between them to provide reliable sensing.

6.2 Intermittent execution

Intermittent execution models enable applications to progress despite frequent power failure [5, 8, 13, 23, 38]. To this end, they decompose an application into several small pieces and save the state of the computation progress on the transitions between these

code segments. Therefore, intermittent applications do not return to the same execution point (e.g., `main()`) after each power failure—in contrast to the applications that assume continuous power supply—instead they resume from the last successfully saved state of execution.

Intermittent systems are regarded as the successor of energy-aware systems. Dewdrop [6] is an energy-aware runtime for (Computational) RFIDs such as WISP. Before executing a task, it goes into low-power mode until sufficient energy is accumulated. QuarkOS [43] divides the given task (i.e., sending a message) into small segments and sleeps after finishing a segment for charging energy. However, these systems are not power disruption tolerant. In other words, if a system could not sustain the energy consumption of low-power-mode and powers down, then all the computation progress will be lost.

Mementos [31] proposed a volatile memory checkpointing-based approach to enable long-running applications on intermittently powered devices. DINO [30] enables safe non-volatile memory access despite power failures. Chain [8] minimizes the amount of data need to be protected by introducing the concepts of atomic tasks and data-channels. Hibernus [2, 3] measures the voltage level in the energy buffer to reduce the number of checkpoints. Ratchet [39] uses compiler analysis to eliminate the need for programmer intervention or hardware support. HarvOS [5] uses both compiler and hardware support to optimize checkpoint placement and energy consumption. Mayfly [15] enables time-aware intermittent computing. InK [41] introduces event-driven intermittent execution. All the aforementioned studies focus their development on a single intermittent node. The paper is the first that considers the abstraction of a group of intermittent nodes and investigates the emerging collective duty cycle of the system.

6.3 Speech recognition

Speech recognition consists of several steps. The basic steps are: *Speech recording and signal digitization*—a microphone records the sound waves and an ADC converts the microphone signal into a digital signal. A sampling rate of about 8 kHz is required to capture the frequencies of a human voice (100-4000Hz [4]). *Framing*—after that the digitized signal is divided into blocks of usually 10-30 ms [10–12] called frames. *Features extraction*—for each frame a feature vector is extracted containing all the relevant acoustic information. *Feature matching*—finally the extracted features are matched against features known to the recognizer.

The speech recognition problem has been tackled from many angles and has experienced many breakthroughs. For example, the dynamic time warping (DTW) algorithm enables matching voice signals with different speed (or time) [40]. Approaches based on Hidden Markov Models showed much better performance than DTW-based ones [20]. Hence, they became the standard techniques for general-purpose speech recognition until artificial intelligent algorithms [16], however, outperform them. Furthermore, many specialized hardware architectures for speech recognition have been proposed to, for instance, reduce energy consumption [27, 28].

Speech recognition algorithms can be classified based on the type of speech that they can recognize into *spontaneous speech, continuous speech, connected word,* and *isolated word* [12]. Systems with *continuous* or *spontaneous speech* recognition are the closest to natural speech, but are the most difficult to create because they need special methods to detect words boundaries [12]. This is less the case for the *connected word* type, where a minimum pause between the words is required. The type with the least complexity is the *isolated word* type. It requires a period of silence on both sides of a spoken word and accepts only single words.

Voice is a natural way for the human to interact with small devices. However, implementing speech recognition on resources—memory, computation power, and energy—limited platforms is challenging, to say the least. Therefore, we attempt to recognize, with our command recognizer prototype, the simplest type of speech, isolated words.

7 DISCUSSION AND FUTURE WORK

[MAC for backscattering and favorable energy conditions]

Intermittent timing algorithm when the harvested power is very low the accuracy of inferring the charging time from the discharging degrades. However, for the Coalesced Intermittent Sensor this is not a serious problem as the intermittent nodes need to randomize their response to external events in favorable energy conditions.

Coalesced Intermittent Sensor Speech recognition on intermittent devices In this paper, we have shown the feasibility of speech recognition on intermittent power. We also demonstrated the possibility of recognizing burst of events (in our case four words). However, the type of speech we targeted is the simplest, isolated words. Next, we may attempt recognizing a more complicated type of speech and for a larger number of words than the number chosen for this study.

Additionally, the command recognition rate could further be improved by using an estimation of the energy left in the energy buffer, to start recharging early. This will prevent a detection when there is not enough harvested energy to record for a long enough time, letting a node recharge earlier and coming back with sufficient energy.

8 CONCLUSION

We presented the *Coalesced Intermittent Sensor* (CIS), an intermittently powered "sensor" that senses continuously! CIS is built around the observation that multiple intermittent nodes distribute themselves uniformly in time. This observation enables us to accurately model, and validate on real hardware, the CIS availability—the collective on-time of its intermittent nodes.

An important finding is that favorable energy conditions may cause sleeping intermittent nodes to synchronize their power cycles on the arrival of the first event. Consequently, they react to the same event, start recharging at the same time, and missing the next event. To counter this unwanted behavior we designed an algorithm to estimate the number of sleeping neighbors and respond proportionally to an event. We show that the Coalesced Intermittent Sensor is able to distribute bursts of events on its nodes "evenly" and capture the entire burst with above 90% detection accuracy.

Additionally, we prototype a battery-less coalesced intermittent command recognizer and show that it can successfully capture events of multiple words.

REFERENCES

[1] Jayam Prabhakar Aditya and Mehdi Ferdowsi. 2008. Comparison of NiMH and Li-ion batteries in automotive applications. In *2008 IEEE Vehicle Power and Propulsion Conference*. IEEE, 1–6.

[2] Domenico Balsamo, Alex S Weddell, Anup Das, Alberto Rodriguez Arreola, Davide Brunelli, Bashir M Al-Hashimi, Geoff V Merrett, and Luca Benini. 2016. Hibernus++: a self-calibrating and adaptive system for transiently-powered embedded devices. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35, 12 (2016), 1968–1980.

[3] Domenico Balsamo, Alex S. Weddell, Geoff V. Merrett, Bashir M. Al-Hashimi, Davide Brunelli, and Luca Benini. 2015. Hibernus: Sustaining Computation During Intermittent Supply for Energy-harvesting Systems. *IEEE Embedded Syst. Lett.* 7, 1 (March 2015), 15–18.

[4] C. Bernal-Ruiz, F.E. Garcia-Tapias, B. Martin-del Brio, A. Bono-Nuez, and N.J. Medrano-Marques. 2005. Microcontroller Implementation of a Voice Command Recognition System for Human-Machine Interface in Embedded Systems. *2005 IEEE Conference on Emerging Technologies and Factory Automation* 1 (2005), 587–591. https://doi.org/10.1109/ETFA.2005.1612576

[5] Naveed Anwar Bhatti and Luca Mottola. 2017. HarvOS: Efficient code instrumentation for transiently-powered embedded sensing. In *Information Processing in Sensor Networks (IPSN), 2017 16th ACM/IEEE International Conference on*. IEEE, 209–220.

[6] Michael Buettner, Ben Greenstein, and David Wetherall. 2011. Dewdrop: an Energy-aware Runtime for Computational RFID. In *Proc. NSDI*. USENIX, Boston, MA, USA, 197–210.

[7] Alexei Colin, Graham Harvey, Brandon Lucia, and Alanson P Sample. 2016. An energy-interference-free hardware-software debugger for intermittent energy-harvesting systems. *ACM SIGPLAN Notices* 51, 4 (2016), 577–589.

[8] Alexei Colin and Brandon Lucia. 2016. Chain: tasks and channels for reliable intermittent programs. *ACM SIGPLAN Notices* 51, 10 (2016), 514–530.

[9] Alexei Colin, Emily Ruppel, and Brandon Lucia. 2018. A reconfigurable energy storage architecture for energy-harvesting devices. In *ACM SIGPLAN Notices*, Vol. 53. ACM, 767–781.

[10] Brian Delaney, Nikil Jayant, Mat Hans, Tajana Simunic, and Andrea Acquaviva. 2002. A low-power, fixed-point, front-end feature extraction for a distributed speech recognition system. In *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, Vol. 1. IEEE, I—-793.

[11] Brian Delaney, N Jayant, and T Simunic. 2005. Energy-aware distributed speech recognition for wireless mobile devices. *IEEE design & test of computers* 22, 1 (2005), 39–49.

[12] Santosh K Gaikwad, Bharti W Gawali, and Pravin Yannawar. 2010. A review on speech recognition technique. *International Journal of Computer Applications* 10, 3 (2010), 16–24.

[13] 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*. ACM, 199–213.

[14] Josiah Hester and Jacob Sorber. 2017. Flicker: Rapid Prototyping for the Battery-less Internet-of-Things. In *Proc. SenSys*. ACM, Delft, The Netherlands.

[15] 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*. ACM, 17.

[16] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine* 29, 6 (2012), 82–97.

[17] Greg Hopper and Reza Adhami. 1992. An FFT-based Speech Recognition System. *Journal of the Franklin Institute* 329, 3 (1992), 555–562.

[18] IXYS Corporation. 2018. IXOLAR™ High Efficiency SLMD121H04L Solar Module. http://ixapps.ixys.com/DataSheet/SLMD121H04L_Nov16.pdf

[19] JBL. 2019. JBL Go+ bluetooth speaker. https://www.jbl.com/

[20] Frederick Jelinek. 1997. *Statistical methods for speech recognition*. MIT press.

[21] Vincent Liu, Aaron Parks, Vamsi Talla, Shyamnath Gollakota, David Wetherall, and Joshua R Smith. 2013. Ambient backscatter: wireless communication out of thin air. In *ACM SIGCOMM Computer Communication Review*, Vol. 43. ACM, 39–50.

[22] Brandon Lucia, Vignesh Balaji, Alexei Colin, Kiwan Maeng, and Emily Ruppel. 2017. Intermittent Computing: Challenges and Opportunities. In *LIPIcs-Leibniz International Proceedings in Informatics*, Vol. 71. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[23] Brandon Lucia and Benjamin Ransford. 2015. A simpler, safer programming and execution model for intermittent systems. *ACM SIGPLAN Notices* 50, 6 (2015), 575–585.

[24] Amjad Yousef Majid, Michel Jansen, Guillermo Ortas Delgado, Kasım Sinan Yıldırım, and Przemysław Pawełczak. 2019. Multi-hop Backscatter Tag-to-Tag Networks. *arXiv preprint arXiv:1901.10274* (2019).

[25] Monsoon Solutions Inc. 2018. High Voltage Power Monitor. https://www.msoon.com/hvpm-product-documentation

[26] Saman Naderiparizi, Aaron N. Parks, Zerina Kapetanovic, Benjamin Ransford, and Joshua R. Smith. 2015. WISPCam: A Battery-Free RFID Camera. In *Proc. RFID*. IEEE, San Diego, CA, USA, 166–173.

[27] Michael Price, James Glass, and Anantha P Chandrakasan. 2018. A Low-Power Speech Recognizer and Voice Activity Detector Using Deep Neural Networks. *IEEE Journal of Solid-State Circuits* 53, 1 (2018), 66–75.

[28] Michael Price, James R Glass, and Anantha P Chandrakasan. 2015. A 6 mW, 5, 000-Word Real-Time Speech Recognizer Using WFST Models. *J. Solid-State Circuits* 50, 1 (2015), 102–112.

[29] pui audio, vesper. 2019. PMM-3738-VM1010-R: A ZeroPower Listening piezoelectric MEMS microphone. http://www.puiaudio.com/pdf/PMM-3738-VM1010-R.pdf

[30] Benjamin Ransford and Brandon Lucia. 2014. Nonvolatile memory is a broken time machine. In *Proc. MSPC*. ACM, Edinburgh, United Kingdom.

[31] Benjamin Ransford, Jacob Sorber, and Kevin Fu. 2012. Mementos: System support for long-running computation on RFID-scale devices. In *Proc. ASPLOS*. ACM, Newport Beach, CA, USA, 159–170.

[32] VS Rao. 2017. Ambient-Energy Powered Multi-Hop Internet of Things. (2017).

[33] Saleae. 2017. Logic 16 Analyzer. http://www.saleae.com. Last accessed: Jul. 28, 2017.

[34] Joshua R. Smith, Alanson P. Sample, Pauline S. Powledge, Sumit Roy, and Alexander Mamishev. 2006. A Wirelessly-Powered Platform for Sensing and Computation. In *Proc. UbiComp*. ACM, Orange County, CA, USA.

[35] Vamsi Talla, Bryce Kellogg, Shyamnath Gollakota, and Joshua R Smith. 2017. Battery-free cellphone. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 2 (2017), 25.

[36] Texas Instruments. 2018. Ultra Low Power Management IC, Boost Charger Nanopowered Buck Converter Evaluation Module. http://www.ti.com/tool/BQ25570EVM-206

[37] Texas Instruments. 2019. MSP430FR5994 16 MHz Ultra-Low-Power Microcontroller Product Page. http://www.ti.com/product/MSP430FR5994

[38] Joel Van Der Woude and Matthew Hicks. 2016. Intermittent Computation without Hardware Support or Programmer Intervention.. In *OSDI*. 17–32.

[39] Joel Van Der Woude and Matthew Hicks. 2016. Intermittent Computation Without Hardware Support or Programmer Intervention. In *Proc. OSDI*. ACM, 17–32.

[40] T. K. Vintsyuk. 1972. Speech discrimination by dynamic programming. *Cybernetics* 4, 1 (1972), 52–57.

[41] 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*. ACM, 41–53.

[42] Hong Zhang, Jeremy Gummeson, Benjamin Ransford, and Kevin Fu. 2011. *Moo: A Batteryless Computational RFID and Sensing Platform*. Technical Report UM-CS-2011-020. UMass Amherst.

[43] Pengyu Zhang, Deepak Ganesan, and Boyan Lu. 2013. Quarkos: Pushing the operating limits of micro-powered sensors. In *Proceedings of the 14th USENIX conference on Hot Topics in Operating Systems*. USENIX Association, 7–7.

[44] Yi Zhao, Joshua R Smith, and Alanson Sample. 2015. NFC-WISP: A sensing and computationally enhanced near-field RFID platform. In *RFID (RFID), 2015 IEEE International Conference on*. IEEE, 174–181.