# Distributed Batteryless Intermittent Sensors

Amjad Yousef Majid
a.y.majid@tudelft.nl
Delft University of Technology

Patrick Schilder
p.t.schilder@student.tudelft.nl
Delft University of Technology

Koen Langendoen
k.g.langendoen@tudelft.nl
Delft University of Technology

Stephan Wong
j.s.s.m.wong@tudelft.nl
Delft University of Technology

## ABSTRACT

Small sensors are desired. As battery size shrinks, its stored energy reduces too. This imposes regular maintenance of devices otherwise functional. Sensors have to forgo batteries and rely on perpetual energy sources, such as light, to enable long-term affordable sensing. Battery-less sensors, however, operate intermittently, when ambient energy is available. Intermittent operation prevents wide adaptation of battery-less sensors as it does not meet the requirements of many real world applications.

To bridge this gap we propose the concept of a distributed intermittent system—the abstraction of a group of intermittently powered devices (or nodes). We hypothesize that as the number of intermittent devices increase their collective up time approaches continuous time. However, if power cycles of intermittently powered devices are correlated, then they may tend to cluster and the benefit of adding nodes demolish, unless special techniques are applied.

This paper investigates distributed intermittent systems, proposes techniques to control the overall on-time of distributed systems, and demonstrates the first distributed intermittent system, namely, a distributed intermittent microphone.

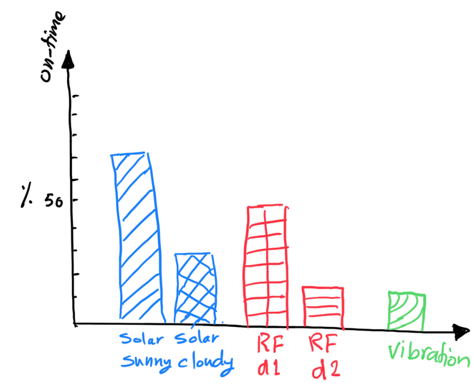
## CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

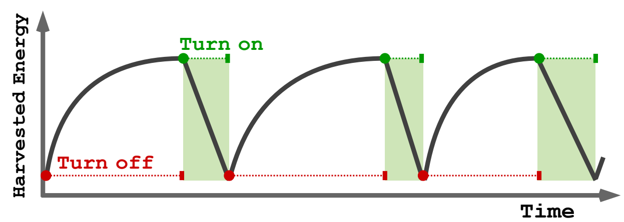
## KEYWORDS

Embedded systems, Distributed intermittent computing

**ACM Reference Format:**
Amjad Yousef Majid, Patrick Schilder, Koen Langendoen, and Stephan Wong. 2019. Distributed Batteryless Intermittent Sensors. In *Proceedings of ACM Conference (Conference'19).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn




**(a) The on-time percentage of an intermittent device powered by different energy sources relative to its on/off cycle length**




**(b) The power cycle of an intermittently powered device.**

## 1 INTRODUCTION

Intermittently powered devices use their environment as an energy source instead of batteries. Therefore, they promise a small, cheap, and maintenance-free version of the current Internet of Things (IoT) edge devices. Driven by this vision, recent years have paid significant attention to intermittent systems [2, 3, 8, 13, 29]. However, their inherent sporadic operation patterns (Figure 1a) have prevented researchers from demonstrating real word applications, for example [2, 7] present activity recognition applications without driving a real sensor to capture external signals.


*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

This paper responds to the challenge of intermittent power supply by introducing the concept of *distributed intermittent systems*. A distributed intermittent system is defined as the abstraction of a group of tiny intermittently powered devices (or nodes). It increases the temporal and spatial availability of an intermittent system and enables resource distribution such as large number of words templates for spoken words recognition systems. The on-time of the distributed intermittent system should approach continuous time as the number of intermittent devices increases. However, the on-time and off-time of distributed intermittent systems depend on the environment and the load. As such, we do not expect, for example, a linear relationship between the number of nodes and the overall on-time.

Controlling the on/off cycle of intermittent devices enables adapting them to many real world applications. For example, once a certain on/off cycle is preserved, an intermittent wake-up receiver can be implemented; intermittent acoustic monitoring system for monitoring engines modules—the sound produced by a deformed gear tooth—can be made. Moreover, with the advances in passive communication (such as passive light [], and backscatter tag-to-tag [] communication) battery-free miniaturized sensors can form self-powered wireless sensor network to, for instance, create smart wallpaper and revolutionize smart buildings.

This paper pushes the boundaries of intermittent systems by: [update the contributions and challenges]

- introducing *distributed intermittent systems* to control the duration of the up time of intermittent sensors and increase their responsiveness,
- investigating the relation between distributed intermittent systems power cycle and their environment,
- demonstrating the *world's first* distributed intermittent system: a distributed microphone.

# 2 BACKGROUND

## 2.1 Energy harvesting

Ambient energy is volatile, and scarce. For example, radio waves harvestable power varies from nW-scale when harvesting ambient RF energy to µW-scale when harvesting a dedicated RF signal; and solar power ranges from tens of µW to tens of mW when it is harvested by a solar panels of a few cm$^2$ illumination surface [13, 21].

A tiny energy-harvesting device slowly charges its energy buffer (e.g. capacitor) while the device is off. Once the buffer is full, the device begins operating and depleting its energy reservoir—since power consumption is much higher than harvested power—until it powers down. This charging-discharging cycle repeats indefinitely (Figure 1b).

## 2.2 Intermittent computing

Intermittent execution models [1, 2, 14, 27] enable applications to progress despite frequent power failure. They decompose an application into several small code pieces and save the progress state of the application 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, as the applications that assume continuous power supply, instead they resume from the last saved progress state of execution.

## 2.3 Spoken words recognition

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* [6].

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 word boundaries [6]. 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.

Speech recognition consists of several steps. The basic steps are mentioned briefly here: *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 [**?** ]). *Framing*—after that the digitized signal is divided into blocks of usually 10-30 ms [4–6] 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.

# 3 BATTERYLESS DISTRIBUTED SENSING

## 3.1 Events classification

Given an intermittent device with a certain on-time duration ($d_{on}$), we can classify external events from the device perspective as follows:

- *Short events*—The event duration is shorter than the on-time of an intermittent device ($d_{on} < t_{on}$). This event can be (i) repetitive, i.e. the acoustic wave caused by a deformed gear tooth; or (ii) non-repetitive, i.e. single word command for a voice assistant.
- *Long events*—The event duration is much longer than the on-time of an intermittent device ($d_{on} > t_{on}$). These events can be (i) simple, such that capturing a small fraction of the event is sufficient to get all the information, i.e. vibration; or (ii) complex, such that capturing the entire event is required for correct interpretation, i.e. a command of several words to a voice assistant system.

## 3.2 Intermittent devices classifications

Intermittent devices can be classified based on the relation between their on/off cycles and the occurrence of external events:

- An *Event oblivious* intermittent device starts running once its energy buffer is full, regardless of the occurrence of an external event. In other words, the dispersion of the on-times of this type of intermittent devices is unaffected by the occurrence of external events.
- An *Event aware* intermittent device enters sleep mode once its buffer is full and upon the occurrence of an external event starts software execution. The power cycles of this type of

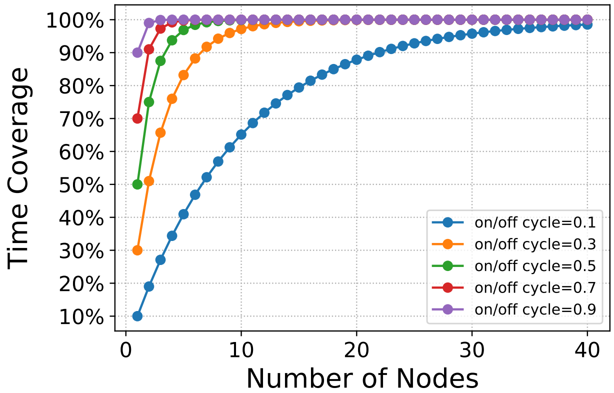



**Figure 2: The on-time of a distributed intermittent system.**

intermittent devices tend to synchronize as their up times is triggered by an event.

Events aware intermittent devices tend to have a longer effective on-time: the time needed to capture an event and save the data in memory.

The on-time evolution of a event-oblivious distributed intermittent system, as more nodes join, depends on the special diversity of its individual nodes.

### 3.3 Spatially diverse intermittent devices

Check— If the nodes are spatially diverse such that their energy harvesting rates are statistically different, then we can assume that the power cycles of the nodes are independent and uniformly distributed over the overall distributed system’s power cycle—When all the nodes power up and shutdown again[1]. When the power cycles are uniformly distributed, adding a node increases the average on-time as follows,

$$\delta t = \frac{t_{off}}{t_{sp}} * n_{on} \tag{1}$$

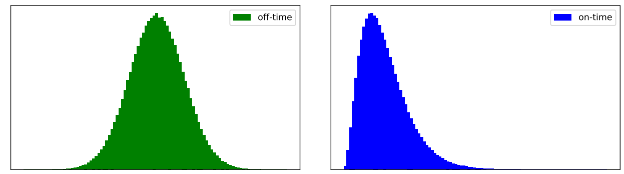
where $t_{off}$ is the off-time of the distributed system, $t_{sp}$ is the period of the distributed system’s power cycle, $n_{on}$ is a node on-time, and $\delta t$ is the time gain of the distributed intermittent system.

As Figure 2 shows adding nodes to intermittent distributed systems increases its overall on-time, and consequently, its responsiveness. It is also clear that the benefit of an additional node is inversely proportional to the distributed system’s on-time. Equation 1, however, holds only when nodes wake-ups approximate a uniform distribution. Check— When the nodes are in close proximity, we cannot model their power cycles as a random variable drawn from a uniform distribution because their energy charging rates are correlated.

### 3.4 Spatially invariant intermittent devices

To understand how the on-time of a distributed intermittent system evolves when devices’ power-ups are correlated, we need to investigate the Check— variance of the off-time and on-time intervals.

[1] [maybe it is better to define it as the mean of the individual power cycles]




**(a) off-time length distribution** **(b) on-time length distribution**

**Figure 3: The distributions of the operational stages of a distributed intermittent system.**

The *off-time* of an intermittent device depends only on the environment: high charging rate results in short charging time and vice versa. Hence, we may model the off-time as a random variable drawn from a normal distribution (Figure 3a). The *on-time*, however, depends on the buffered energy—the harvested energy while the device is off—; the harvested energy while the device is on, which only prolongs the execution; and the load of the device. Therefore, if we assume the load is constant, then we can model the on-time as a random variable drawn from a gamma distribution can be more appropriate (Figure 3b). Another important factor is *the relation* between the off-time and the on-time. Short off-time indicates a high charging rate. A high harvesting rate results in a non-negligible amount of the harvested-while-executing energy. This energy lengthens the on-time. Therefore, we can conclude that there is an inverse relationship between the on-time and the off-time. Check— By considering these three factors (on-time, off-time, and their relation), we can model the variation in intermittent devices’ power cycles as a normal distribution. As a result, when charging rates are correlated, the power-ups of intermittent nodes will tend to cluster around the mean instead of spreading over the entire power cycle of the distributed intermittent system.

[relocate the following paragraph] To flatten the normal distribution we need to add another random variable that is drawn from a uniform-like distribution. To achieve that, we further randomize the length of the on-time of intermittent nodes by injecting delays—putting the nodes into sleep mode—upon devices’ power-ups (Figure 4). The maximum delay that can be added is bounded by the buffer size and the minimum energy consumption of the load. The length of this delay represents the spreading factor by which we spread the original distribution of the nodes’ wake-ups.

[Update equation 1 to include the spreading factor]

## 4 BATTERYLESS DISTRIBUTED MICROPHONE

BFAlexa is a prototype of a distributed battery-free audio assistant agent. BFAlexa is envisioned to be embedded in, for example, wallpapers and furniture coverage to turn these objects to smart and controllable ones. The distributed nature of the BFAlexa system enables it to exploit the inherent randomness of intermittent devices to offer a reliable interaction.

BFAlexa is a battery-less distributed microphone. It operates intermittently, when a threshold amount of energy accumulates.

Audio is recorded for a fixed time period. After that, if the audio recording has finished successfully, the recording is divided into

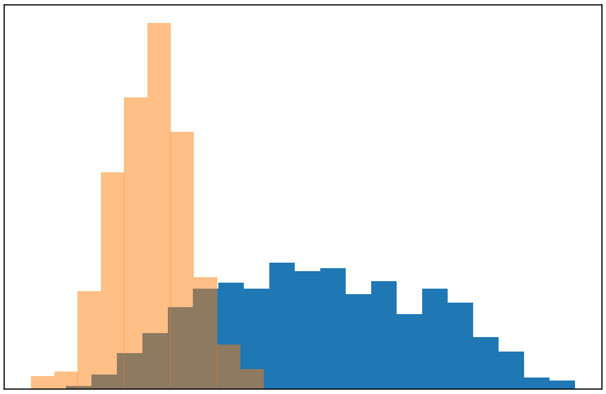


**Figure 4: Spreading normally distributed intermittent devices over a larger range by injecting delays with a uniformly distributed lengths.**

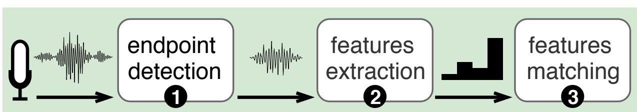



**Figure 5: (1) endpoint detection algorithms extract the part of the signal corresponds to the input word, (2) feature extraction calculates the energy spectrum of the trimmed signal, and (3) feature matching searches the local database for the closest word**

smaller parts called *frames* and speech recognition is performed using an algorithm consisting of three steps: (1) endpoint detection (2) feature extraction (3) feature matching. This steps are also depicted in figure 5.

During *feature extraction* on frames containing the word Fast Fourier Transform is applied to obtain the energy spectrum. The energy spectrum is then sorted into a feature vector, where each vector value holds the total energy of a specific frequency range. Finally each vector value is normalized over the total vector energy.

After that, *feature matching* is performed, where the features of the recorded word are compared against a local database to find the most similar word in the database. There exist many feature matching methods, yet only few are able to run under the constraints of an intermittent system. Two methods have been implemented and tested. The first is Dynamic Time Warping (DTW), which is able to compare two sequences of data, even when they vary in speed. This way if a word is spoken slower or faster, the algorithm can compensate for that. The second is a method that linearly compares two feature vectors. This requires less computations than DTW.

## 4.1 Hardware description

we used MSP430RF5994 [26] an ultra-low-power microcontroller to drive the PMM-3738-VM1010-R microphone that features zero power listening and wake-on-sound technology [18]. A BQ25570EVM-206 [25] solar power harvester connected to a pair of SLMD121H04L [10] solar panels intermittently powers a BFAlexa sensor. For some experiments simulated intermittent power is used. For debugging, we used saleae logic analyzer [22].

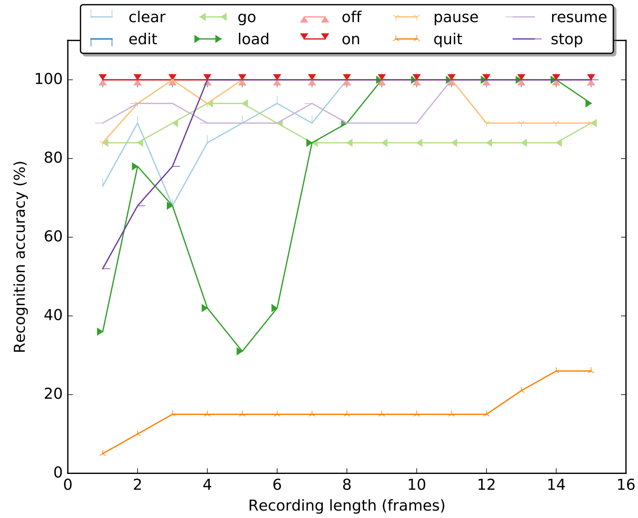



**Figure 6: Recognition accuracy versus the recording length (in frames), while using multiple recordings per word for testing. Used feature matching method: linear.**

## 4.2 Implementation

During recording a sample rate was used of 7812 Hz, which covers the human voice frequency range. The used frame size was 256 samples. This size was beneficial for doing a Fast Fourier Transform and corresponds to approximately 33 milliseconds of speech.

For normalization an integer log function was applied on every value, divided by the average log. The whole algorithm is implemented using only integer / fixed point arithmetic.

In the linear comparing method the feature vectors of two words are compared successively, not accounting for differences in the speed of pronunciation. If two words vary in length, the last frames cannot always be compared and instead a penalty is applied linear to the length difference.

In between the different steps (see figure 5) checkpoints in non-volatile memory are used to assure progress while running on intermittent power. In some cases additional checkpoint were used inside the steps.

# 5 EVALUATION

**Table 1: Testing set**

| | | | | |
|---|---|---|---|---|
| on | off | stop | clear | load |
| go | pause | resume | edit | quit |

## 5.1 Effective recording length

Since our target hardware has extremely limited resources, the first experiment targets the minimum effective recoding length without significant accuracy loss.

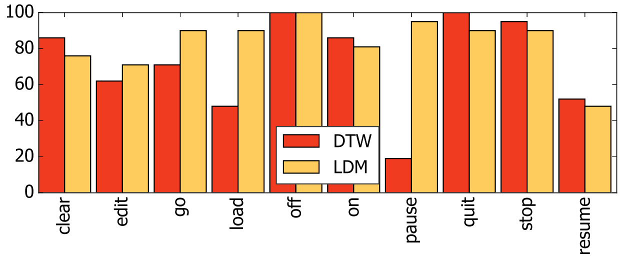



**Figure 7: Recognition accuracy for linear matching and DTW when using 9 frames as the recording length.**

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

| Section | Linear (ms) | DTW (ms) |
|---|---|---|
| Recording | 285.9 | 285.9 |
| Feature extraction | 501.9 | 501.9 |
| Feature matching | 99.4 | 1251 |
| Total | 887.2 | |

**Table 3: The mean and standard deviation of the parameters that were used to simulate intermittent execution. Here $t_{on}$ is the time the device is on while recording or processing, $t_{off}$ is the time the device is charging and $t_{sleep}$ is the time the device is sleeping while waiting on sound input.**

| Parameter | $\mu$ (ms) | $\sigma$ (ms) |
|---|---|---|
| $t_{on}$ | 590 | 17.7 |
| $t_{off}$ | 5310 | 154 |
| $t_{sleep}$ | 22420 | 652 |

For this experiment a single microcontroller running on continuous power is used. Each word from Table 1 was recorded on a PC 20 times. The features—normalized FFT-based values—of the first recording are saved in the microcontroller's persistent memory as a signature to perform the feature matching, during the testing. the rest of the recording are used for conducting the experiment.

Figure 6 shows words recognition accuracy when the 19 recordings of each word are played back from the PC speaker. We can concluded that recording beyond *nine frames* do not increase the recognition accuracy; therefore, nine frames recording length is chosen for the rest of the experiments.

### 5.2 Comparison of feature matching methods

We implemented two algorithms for voice features matching, Dynamic Time Warping (DTW) and Linear Distance Matching (LDM). Due to the microphone wake-on-sound feature and the fixed recording length, DTW did not outperform LDM as Figure 7 shows. Moreover, DTW takes more time to process that data as Table 2 shows. Therefore, LDM algorithm is used in future experiments.

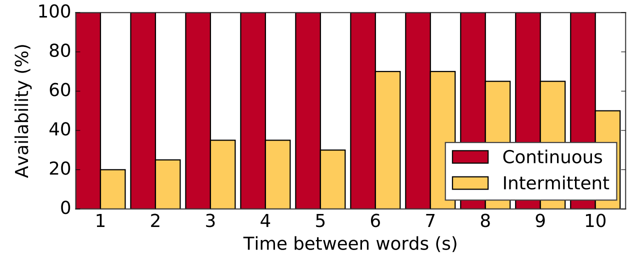



**Figure 8: The effect of the time between consecutive words on the availability: the percentage of words that is processed by the command recognizer. [Add correct recognition as stacked bar?]**

### 5.3 Intermittent microphone availability

This experiment shows the effect of intermittent power supply on a microphone availability [and recognition accuracy] for different command (or word) repetition speed. Each word (from Table 1) was played back 20 times with intra-words-playing time ranges from 1 to 10 seconds.

For the experiment we chose to simulate the intermittent power supply, based on a real measurement, to ensure environment consistency across different words. In particular, we measured the $t_{on}$; used an on/off ratio of 10% to simulate harsh harvesting conditions [20**? ?**]; and calculated the $t_{sleep}$ based on the micro controller specification [**?** ] (Table 3).

The obvious observation from Figure 8 is that intermittency has a great impact on the command recognizer availability. However, a less obvious, but important, observation is that the correlation between the on/off cycle and the command repetition speed has also great impact as the bars labeled with 5 and 6 from Figure 8 indicate.

## 6 RELATED WORK

### 6.1 Energy-harvesting

Many batteryless energy-harvesting platforms have been proposed, for example, Wireless Identification and Sensing Platform (WISP) [23] and its variants such as NFC-WISP [31], WISPcam [15] and, NeuralWISP [28], batteryless phone [24], ambient backscatter tag [12] Moo [30]

### 6.2 Intermittent execution

Intermittent systems regarded as the successor of energy-aware systems. Dewdrop [] is an energy-aware runtime for (Computational) RFIDs such as WISP. Dewdrop goes into low power mode until sufficient energy for a given task is accumulated. QuarkOS [] divides the given task (i.e. sending a message) into small segments and sleep after finishing a segment for charging energy. However, these systems are not disruption tolerance.

The first power failure tolerant systems use the idea of volatile progress state checkpoiting into persistent memory [20]. DINO [19], however, shows that in addition to the volatile memory, the nonvolatile memory of the processor must also be protected to ensure correct executions. Hibernus [] measure the voltage level in the energy buffer to reduce the number of checkpoints. Ratchet [] uses compiler analysis to eliminate the need of programmer intervention or hardware support. HarvOS [] uses both compiler and hardware support to optimize checkpoint placement and energy consumption.

Task-based systems optimize intermittent execution by reducing the amount of data needed to be saved into non-volatile memory to protect applications against power interruptions [? ]. However,

### 6.3 Speech recognition

The speech recognition problem has been tackled from many angles and has experienced many great breakthroughs. For example, Dynamic time warping (DTW) algorithm enables matching voice signals with different speed (or time) []. Approaches based on Hidden Markov Models showed much better performance than DTW-based ones [11]. Hence, they became the standard techniques for general purpose speech recognition until artificial intelligent algorithms [9], however, outperform them.

Many specialized hardware architectures for speech recognition have been proposed to, for instance, reduce energy consumption [16, 17].

## 7 DISCUSSION AND FUTURE WORK

## 8 CONCLUSION

## REFERENCES


[1] 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.

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

[3] 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.* ACM, 116–127.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] Matthew Hicks. 2017. Clank: Architectural support for intermittent computation. In *Computer Architecture (ISCA), 2017 ACM/IEEE 44th Annual International Symposium on.* IEEE, 228–240.

[9] 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.

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

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

[12] 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.

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

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

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

[20] 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.

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

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

[23] 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.

[24] 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.

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

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

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

[28] Daniel J Yeager, Jeremy Holleman, Richa Prasad, Joshua R Smith, and Brian P Otis. 2009. Neuralwisp: A wirelessly powered neural interface with 1-m range. *IEEE Transactions on Biomedical Circuits and Systems* 3, 6 (2009), 379–387.

[29] 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.

[30] 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.

[31] 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.