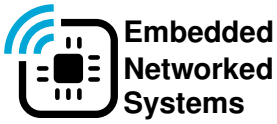
Delft University of Technology
Master's Thesis in Embedded Systems

Command Recognition on Intermittently-Powered Devices

Patrick Schilder



TU Delft Delft University of Technology

Command Recognition on Intermittently-Powered Devices

Master's Thesis in Embedded Systems

Embedded and Networked Systems Section
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands

Patrick Schilder
p.t.schilder@student.tudelft.nl

30th April 2019

Author
Patrick Schilder (p.t.schilder@student.tudelft.nl)
Title
Command Recognition on Intermittently-Powered Devices
MSc presentation
9th May 2019

Graduation Committee

Prof. dr. K.G. Langendoen (chair)	Delft University of Technology
Dr. ir. J.S.S.M. Wong	Delft University of Technology
M.A. Zuniga, PhD.	Delft University of Technology
A. Majid, MSc.	Delft University of Technology

Abstract

The Internet of Things (IoT) is expected to include billions of tiny devices that collect, process, and communicate sensory data. As of now, batteries power these devices. Batteries, however, are large, expensive, and short-lived – even the rechargeable ones wear out in a few years. Therefore, they are not a sustainable powering solution. Tiny battery-less devices promise a maintenance-free and environment-friendly alternative. They operate by harvesting energy from the environment. Ambient power, however, is marginal and unpredictable. This causes tiny energy-harvesting devices to operate *intermittently*, violating the requirements of many real-world applications.

This work presents the Coalesced Intermittent Command Recognizer (CICR), a group of intermittently-powered sensors that together perform a real-world application: command recognition. To achieve this, we first developed an event-based command-recognition algorithm tailored towards battery-less sensors, taking into account the challenges of intermittent execution. We then used this algorithm on multiple intermittent sensors to make use of their collective availability. To experience continuous operation of the CICR – at all times – at least one of the sensors must be on. In worst-case conditions (little, intermittent power) the random nature of the ambient power source (e.g. solar) randomizes the awake times. However, as the energy conditions increase, sensors react collectively on the same word – depleting their energy buffer all at the same time and missing subsequent words. To counter this behavior we use a probabilistic algorithm to postpone sensor reactions.

We implemented a CICR consisting of 8 battery-less recognizers on real hardware and constructed a solar testbed for evaluation. Single-word commands are recognized 50% of the time at a light intensity of 500 lux and over 90% of the time at $\geq$800 lux. For multiple-word commands, our probabilistic approach effectively distributes the recognizers over the words of the command. Many recognizers, however, then have insufficient energy left in their buffer to finish recording, which drastically reduces the capture rate. This can be mitigated by a hardware modification that would allow recognizers to start recharging as soon as the the amount of energy becomes too low for recording.

Preface

Computers have always held my interest. By studying Embedded Systems at the Delft University of Technology I could extend that interest and learn how computers connect to the physical world we live in. Battery-less IoT sensors, however, are a whole step further into the future. They form a great challenge in exchange for a sustainable solution. Writing my thesis about this relatively new topic at the Embedded & Networked Systems group at the TU Delft offered a taste of the future of technology.

Of course, I could not have completed this project without the help of others. First of all I want to thank my daily supervisor, Amjad Majid, for the frequent discussions, brain-storming sessions, and other help. Further I want to thank my supervisors Przemysław Pawełczak and Koen Langendoen for support and useful feedback, Ioannis for giving advice on designing a custom PCB, and other master and PhD students for the time spent together at the faculty.

I also want to thank my family for always supporting me, and my friends, especially Kavya, Rebecca, Patrícia, Álvaro, David, Madhu, and Carlo, for memorable moments during my thesis, and during my master in general.

Patrick Schilder

Delft, The Netherlands
30th April 2019

Contents

Chapter 1

Introduction

The Internet of Things (IoT) is expected to include billions (or trillions) of tiny devices that collect, process, and communicate sensory data. These devices will improve healthcare [8], water infrastructure [59], and smart buildings [1, 14], to name a few.

As of now, batteries power these devices. Batteries, however, are large, expensive, and short-lived – even the rechargeable ones wear out in a few years. They impose regular maintenance of devices that are otherwise functional. Therefore, they are not a sustainable powering solution.

Tiny **battery-less** devices promise a maintenance-free and environment-friendly alternative. They operate by harvesting energy from ambient sources such as light [40, 41], vibration [22], and radio frequency waves [21]. Tiny energy harvesters, however, can only scavenge very limited power from such energy sources [36]. Therefore, the execution is *intermittent*: it is triggered when a threshold in the energy buffer (e.g. super-capacitor) is reached, and terminated when the energy buffer has been depleted (Figure 1.1). Because of intermittent execution data processing, sensing, and communication are often disrupted, clocks are reset, and volatile memory is lost. Recent advances in checkpointing [3, 54], data consistency [11, 38], timekeeping [17, 26], energy management [24], testing [23], and debugging [13] address some of the key challenges of intermittent execution.

Existing intermittent applications include temperature monitoring [12], voice presence detection [68] and robot actuation [68]. So far, intermittent applications have been often implemented as toy applications, i.e. proof of concepts instead of functional applications. This thesis contributes to the intermittent-sensing community with an implementation of a functional speech recognition algorithm for intermittently-powered devices.

Speech recognition has long been studied, and currently most of the algorithms employ hidden Markov models (HMMs) for feature matching [18, 29, 45]. Recently, also deep neural networks are employed for acoustic modeling of state-of-the-art speech recognition systems [27]. The acoustic and

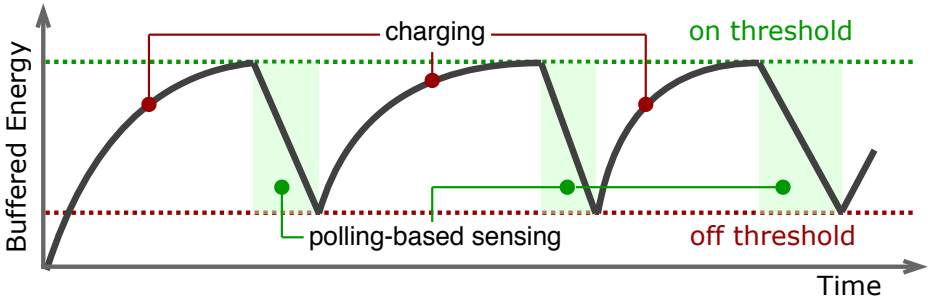


Figure 1.1: The execution of an intermittent system is triggered when an energy threshold in the buffer is reached, and terminated when the energy buffer has been depleted.

language models used for speech recognition typically use tens to hundreds of megabytes of storage with significant computation required for large vocabulary search [16].

Due to their limited resources, battery-less devices can only run a much simpler speech recognition algorithm. One that is still able to recognize a limited set of words (i.e. 10-20), requiring a significant smaller amount of computation, and therefore less energy.

In many useful cases a simple form of speech recognition – *command recognition* – suffices to communicate the necessary information from a user to a device. The use of battery-less devices for human-device interaction could offer a green and maintenance-free alternative, especially in places where a limited vocabulary size is needed, e.g. data entry applications in smart homes [44].

Despite significant progress achieved in the intermittent domain, *the system availability problem* has not been addressed. A monitoring sensor that has a very low probability to be available when an external event occurs is not worth deploying. A sensor that is capable of capturing only very short events has a limited number of potential applications. For example, a voice-controlled light-switch capable of only accepting short (single-word) commands has its limitations. Using "on" to turn on the lights might turn on other devices as well. Using "lights" does not allow the specification of "on" or "off". Consequently, intermittent sensors have not gained widespread adoption.

This thesis touches the paradox of continuous sensing on intermittent devices. It studies the power cycles of energy-harvesting command recognizers and makes a key observation about the relationship between those power cycles. Energy-harvesting recognizers driven by the same ambient energy source (e.g. light) do *not* show correlated on/off (sense/charge) cycles. Building on top of this observation, we introduce *coalesced intermittent*

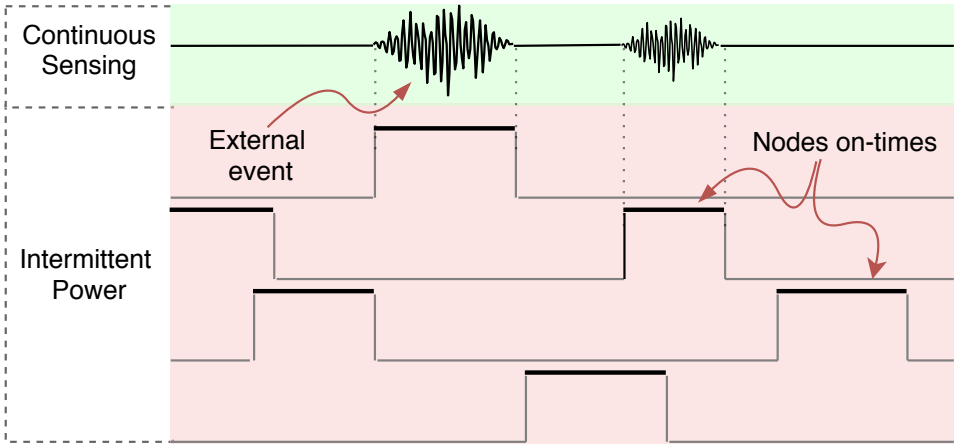


Figure 1.2: A Coalesced Intermittent Command Recognizer is a group of intermittently-powered nodes that sense continuously despite intermittent power supply.

command recognizer (CICR). The CICR is defined as the abstraction of a group of energy-aware intermittent word recognizers providing the collective sense of continuous availability. Figure 1.2 illustrates the CICR concept; a number of solar-powered nodes equipped with a microphone, recording voice commands in a smart home setting. Recording and processing a word depletes the super-capacitor that powers a node, leaving it unresponsive until the subsequent recharging completes. Multiple nodes with (partially overlapping) on/off cycles spread in time can provide continuous service despite the inherent intermittency.

In contrast to periodic (one-shot) sensing applications, event-based applications (like our command recognizer) may induce implicit synchronization (multiple nodes detecting the same command) that compromises the availability required by the application (all nodes recharge after the first word, missing any subsequent word). To guarantee continuous availability, a CICR may need to introduce artificial randomness.

1.1 Problem Statement

A lot of research has been performed on intermittent systems already, however most applications are focusing on measuring slow-changing variables, due to the limitations that are introduced by intermittent sensing. It would be beneficial to be able to use small, battery-less devices for continuous sensing.

Often only certain parts of a signal are interesting. In order to respond to the interesting events, these events themselves can be used to trigger sensing. For example a big vibration could trigger an earthquake detector,

a vibration or sound could trigger a presence detector, or a sound could trigger a command-recognition system.

However, handling bursts of events is challenging on intermittent systems: one device is not sufficient, as it may be unavailable or will need to recharge quickly, and using multiple devices requires specialized algorithms. Keeping this in mind and focusing on command recognition, the main research objective is formulated as follows:

Develop, implement, and evaluate a command-recognition algorithm for intermittently-powered devices.

During evaluation properties of interest are system availability and recognition accuracy.

1.2 Contributions

The key contributions of this thesis are twofold:

- Development and implementation of a command-recognition algorithm that is able to run on intermittently-powered devices. We used isolated-word recognition together with a probabilistic desynchronizing algorithm to distribute individual command recognizers over a four-word command.
- Evaluation of a coalesced intermittent command recognizer consisting of eight intermittent nodes. We show that the CICR recognizes over 90% of single-word commands with $\geq$800 lux, but that distributing the nodes over a four-word command often leaves the nodes with insufficient energy left in their buffer by the time they start recording.

1.3 Thesis Outline

The layout of this thesis is as follows: The background of this thesis topic and related work will be discussed in Chapter 2. System design and implementation will be described in Chapter 3. Results will be presented in Chapter 4. Finally the conclusions will be presented in Chapter 5 as well as directions for future work.

Chapter 2

Background and related work

This chapter provides background information on energy harvesting (Section 2.1), intermittent execution (Section 2.2), and speech recognition (Section 2.3).

2.1 Energy harvesting

Many studies have proposed techniques for harvesting kinetic energy [22], solar energy [40, 41], thermal energy [4] and energy from radio frequency (RF) waves [21].

Recently specialized small-form-factor harvesting circuits have been developed [61], making it possible to use some of the natural energy resources in IoT devices. Since then, many battery-less energy-harvesting platforms have been proposed. Some of them rely on dedicated external energy sources such as WISP, a general wireless identification and sensing platform [58, 71, 69]; WISPcam, an RF-powered camera [46]; and the battery-free cellphone [60]. Others, harvest from ambient sources such as the ambient backscatter tag [36], and the solar-powered tag [39]. Other platforms that facilitate the development of batteryless energy-harvesting systems have also been proposed. For instance, Flicker [25], a prototyping platform for battery-less devices; EDB [10] an energy-interference-free debugger for intermittent devices; and Capybara [12], a re-configurable energy storage architecture for energy-harvesting devices.

Ambient energy, however, is volatile and scarce. For example, harvestable RF 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 of illumination surface [37, 55]. For comparison, an intermittent command recognizer uses up to 850 µW (Section 3.4).

Tiny solar panels will not be able to provide enough energy to power the command recognizer continuously, thus it will have to operate intermittently.

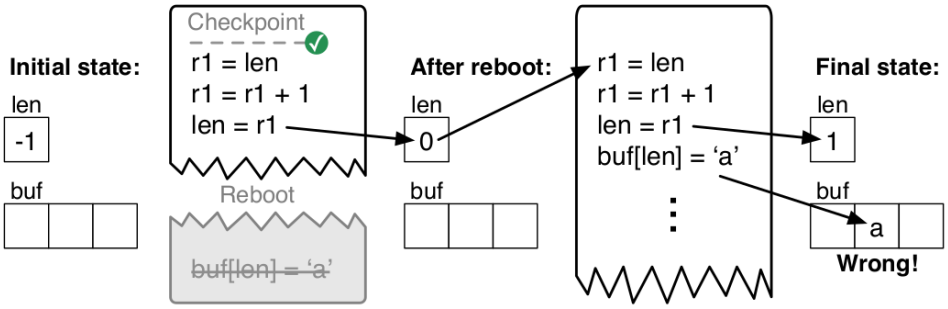


Figure 2.1: Example of a failure that can occur when there is a Write After Read (WAR) dependency during intermittent execution. The variable 'len', which is non-volatile (does hold its value during a reset) is incremented more often than it should. Figure taken from [38].

2.2 Intermittent execution

Intermittent systems are regarded as the successor of energy-aware systems. Dewdrop [7] is an energy-aware runtime for (computational) RFID's such as WISP. Dewdrop goes into low-power mode until sufficient energy for a given task is accumulated. QuarkOS [70] divides a given task (i.e. sending a message) into small segments and sleeps after finishing a segment for charging energy. However, these systems are not disruption tolerant.

On an intermittent system, the execution of a program is composed of periods of sequential execution interrupted by reboots that can occur at any moment in time. During a reboot all volatile memory is cleared, and control returns to the entry point of the application. Only data in non-volatile memory (e.g. Ferroelectric RAM) is retained. Power-failure-tolerant systems [53] use checkpointing of the volatile state into non-volatile memory to ensure forward progress during the frequent resets (up to multiple times per second). Once execution reaches a certain point in the code, a checkpoint is made, from where execution is continued after a reset.

For bigger programs it takes a lot of effort to place checkpoints manually. Ratchet [64] uses compiler analysis to eliminate the need of programmer intervention and hardware support. HarvOS [6] uses both compiler and hardware support to optimize checkpoint placement and energy consumption.

Unfortunately, frequent checkpointing causes overhead. Hibernus [3] measures the voltage level in the energy buffer to reduce the number of checkpoints. Measuring the voltage level, however, still causes some overhead.

Using checkpoints does not automatically ensure data consistency. It may happen that code is executed multiple times because of resets, causing a data value to be altered more often than intended. DINO [52] shows that

in addition to the volatile memory, the non-volatile memory of the processor must also be protected to ensure correct executions. For example, Figure 2.1 shows how non-volatile variable with a Write After Read (WAR) dependency leads to wrong results.

Programming models are available that guarantee data consistency during intermittent execution [6, 38, 64]. To do that, sometimes multiple copies of data need to be made. Some of the models optimize intermittent execution by reducing the amount of data needed to be saved into non-volatile memory to protect applications against power interruptions [11]. Intermittent execution models, however, also enforce certain coding schemes, which are generally cumbersome to use.

2.3 Speech recognition

While there are many speech-recognition algorithms, none work on intermittent devices. This section briefly explains speech types and speech recognition steps, and gives an overview on the history of speech recognition.

2.3.1 Types of speech

Speech recognition algorithms can focus on different types of speech, namely: *spontaneous speech, continuous speech, connected word,* and *isolated word* [20]. The recognition of different types of speech all have their advantages and disadvantages. Systems with *continuous* or *spontaneous speech* recognition are the closest to supporting natural speech, but are the most difficult to create because they need special methods to detect word boundaries [20]. 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 the spoken word and accepts only single words.

2.3.2 Speech-recognition process

Speech recognition usually consists of several steps (Figure 2.2). The basic steps are mentioned briefly here, while a more detailed description of the implemented algorithm is given in Chapter 3.

First the speech has to be recorded. 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 [5]).

After that the digital signal is divided into blocks of usually 10-30 ms called frames [15, 16, 20]. In the rest of the code, speech is processed on a frame-by-frame base. This reduction of dimensionality is possible because,

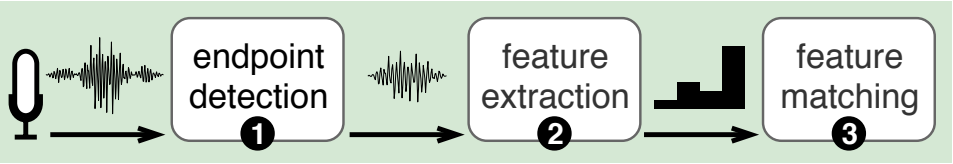


Figure 2.2: Typical steps in a template-based speech-recognition algorithm: (1) An endpoint detection algorithm extracts the part of the audio signal that corresponds to the input word, (2) during feature extraction the energy spectrum of the trimmed signal is calculated, and (3) features are matched against templates from the local database to find the most similar word.

while audio properties are varying with time, the properties can be considered constant on a small time scale.

If the position of the word in the recording is unknown, it needs to be determined by an endpoint detection algorithm [49, 67].

Then, for each frame a feature vector is extracted, which contains the essential acoustic information.

Finally the sound features from the recording are matched against features known to the recognizer. In some statistical-based speech recognition models, feature matching is split into (i) calculating the probabilities of all the possible sounds that could have been pronounced and (ii) matching those sounds to a dictionary of words [18].

2.3.3 History of speech recognition

To give an idea about existing algorithms, we give a brief summary of the advances in speech recognition over the years.

Early systems for automatic speech recognition started appearing in the 1950's and 1960's [33]. They were able to recognize small vocabularies (order of 10-100 words) of isolated words, based on simple acoustic-phonetic properties of speech. The key technologies that were developed in this decade were filter-bank analyses and elementary time-normalization methods, that were based on the ability to reliably detect speech starts and ends [42].

The speech recognition algorithms from the 1970's were able to recognize medium vocabularies (order of 100-1000 words) using simple template-based pattern recognition methods. The key technologies that were developed during this period were pattern-recognition models, Linear Predictive Coding methods for spectral representation, pattern clustering methods for speaker-independent recognizers, and dynamic programming methods for time aligning a pair of speech utterances – known as Dynamic Time Warping (DTW) – including algorithms for connected word recognition [66].

Speech recognition research in the 1980s was characterized by a shift in methodology from the more intuitive template-based approach towards a

more rigorous statistical modeling framework [18]. Today, most speech recognition systems are based on the statistical framework developed in the 1980s [29, 45], with significant improvements having been added in the 1990's. The key technologies introduced during this period were the hidden Markov model (HMM) [50, 51] and the N-gram stochastic language model, which together enabled powerful new methods for handling virtually any continuous speech recognition problem efficiently and with high performance.

In the 1990's large vocabulary systems were built with unconstrained language models, and constrained task syntax models for continuous speech recognition and understanding [33]. Various techniques were investigated to increase the robustness of speech recognition systems against the mismatch between training and testing conditions, caused by background noises, voice individuality, microphones, transmission channels, room reverberation, and so-called disfluencies such as partial words, hesitation, and repairs. In this period speech recognition started to be used within telephone networks to automate operator services [18].

Finally, from 2000 onwards, very large vocabulary systems have been introduced with full semantic models, integrated with text-to-speech synthesis systems. Research has focused on spontaneous speech recognition and further increasing robustness, using techniques like flexible acoustic modeling, sentence boundary detection, pronunciation modeling, adaptation of acoustic as well as language models, and automatic speech summarization [19]. Speech recognition has further entered the market. Mobile devices with internet access have caused a movement towards distributed speech recognition, where the computationally-intensive parts of speech recognition are offloaded to the cloud [2].

Recently, deep neural networks are being employed for acoustic modeling of state-of-the-art speech recognition systems that, however, still often are combined with a HMM [27].

2.3.4 Low-power speech recognition

Developing speech recognition for intermittent devices first of all means developing speech recognition for ultra-low-power devices. A lot of research on low-power speech recognition has been done. Most solutions use special-purpose hardware [9, 43, 47] or are implemented on FPGAs [35]. Others focus on speech recognition for handheld devices [16, 30], which still require significantly higher resources than our targeted hardware (Chapter 3). Open-source speech recognition toolkits such as PocketSphinx [30] and Embedded Julius [34] also focus on handheld devices like smartphones rather than ultra-low-power devices. They use a HMM in combination with large language models and therefore are not suitable for intermittently-powered devices.

The problem of implementing speech recognition on less powerful hardware was already faced in the early 90's. The objective at that time was to bring speech recognition to personal computers in a time where speech recognition systems often required sophisticated special-purpose hardware to obtain reasonable processing times. A speaker-dependent, isolated-word speech-recognition algorithm was implemented for a personal computer using a relatively simple signal processing technique [28]. FFT was used for feature extraction while DTW was used to determine the most likely entry from a dictionary of previously stored word templates. Inspiration was taken from this approach during development of the speech recognition algorithm described in this thesis.

More recent work that does not require special-purpose hardware includes a low-power speech recognition algorithm using neural networks [5]. The algorithm used in that paper does not divide the recording into multiple frames, but uses only one time frame instead, which limits the system to recognize only five vowels instead of whole words. The detection of a vowel takes 347ms on hardware similar to our target hardware. The algorithm does not support intermittent execution, however.

Chapter 3

Design and implementation

We designed a Coalesced Intermittent Command Recognizer (CICR) consisting of multiple intermittently-powered command recognizers (nodes). Each node is capable of extracting and compressing voice features, as well as performing simple word recognition. By combining results from multiple nodes in a central base station, multiple-word commands can be processed, and further actions can be taken (Figure 3.1).

The design aims for devices with a small form factor (a couple of square centimeters), which can be embedded in furniture, wallpaper, or ceiling and placed in an office or home environment.

The requirement of a small form factor also holds for the energy harvesting equipment: A small solar panel or antenna provide the device with some tens to hundreds of µW, depending on the exact size and type, and the energy environment. This calls for an energy-aware design that takes into account low power usage as well as power outages and charging times (intermittent execution).

3.1 Hardware

We selected the ultra-low-power microcontroller MSP430FR5994 [62] from Texas Instruments for all sensing and processing. This microcontroller has a 16-bit RISC processor running at 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. We chose this microcontroller because of the low cost (couple of dollars), and because of the low power consumption and the availability of non-volatile memory in the form of FRAM. FRAM retains data during a power reset, while it is faster and less power hungry than other types of persistent storage like an SD-card.

For recording we used the PMM-3738-VM1010-R [48] piezoelectric MEMS microphone, which features Wake-on-Sound and ZeroPower-listening tech-

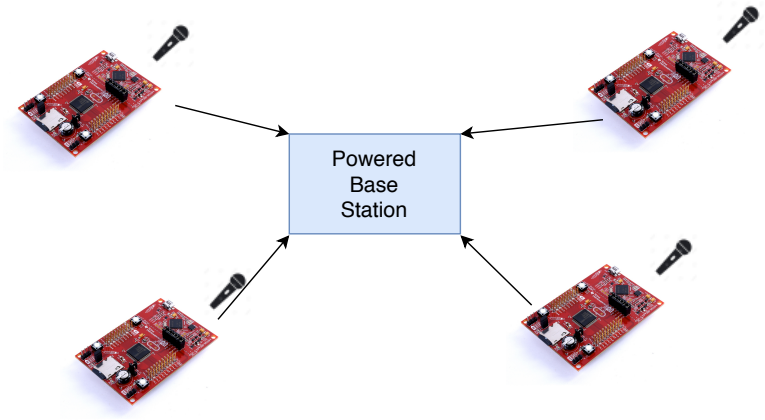


Figure 3.1: Results of the coalesced intermittent command recognizer are combined at a central base station, where multiple-word commands can be processed, and further actions can be taken.

nologies [65], allowing both the microcontroller and the microphone to sleep in a low-power mode until a sound is detected. The microphone chip is mounted on a custom PCB (Appendix A) and has a wired connection to the microcontroller.

The microcontroller and microphone are powered intermittently by a Texas Instruments BQ25570 [61] solar-power harvester connected to an IXYS SLMD121H04L [31] solar panel (6 cm^2) and a super-capacitor of 470 µF.

3.2 Intermittent-node design

The low-power design calls for low-power hardware, which in turn calls for a very simple command-recognition algorithm. We implemented a speaker-dependent, template-based, isolated-word algorithm (implementation details in Section 3.3). These properties of the algorithm have a great impact on the rest of the design.

As the algorithm is speaker dependent, it requires a short training period for each user. This suggests the use in an environment with a small, fixed group of users, such as a smart home or smart office.

An isolated-word algorithm can process one word at the time and requires a short period of silence both before and after a word. This disqualifies it for recognition of continuous speech, but lends it well for command recognition. In fact shorter recording times are preferred on intermittently-powered systems, as a smaller capacitor can be used, decreasing the charging time, and therefore making the device more reactive.

There is a lower recording limit, however. If a recording length were used that is much shorter than a word, a single node would be unable to

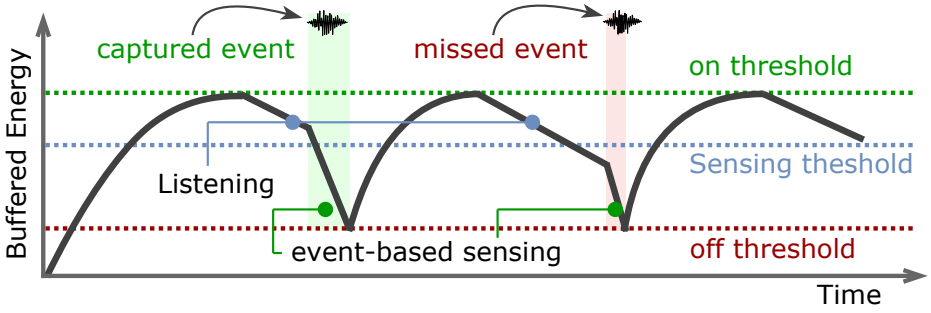


Figure 3.2: A command recognizer node first charges its energy buffer, then stays in low-power mode until it detects a word and starts recording it.

recognize a word on its own, since it would have recorded only a part of a word, and would have to recharge before it can record again. Therefore it would not have enough information to recognize a word. Theoretically different nodes could record different parts of a word and afterwards a word could be reconstructed by putting the pieces together. In reality, accurate timing information to put the sound pieces together is not available on battery-less devices. Additionally, combining several sub-word sound pieces would introduce extensive communication between the nodes — a significant energy investment.

To avoid the need for precise timing and extensive communication, we used a recording length long enough to recognize a word. It is important for the recording to happen uninterrupted: a recording interrupted by a power reset effectively exists of two shorter recordings. Once a node has finished recording, power resets do not affect the recognition result of the just recorded word anymore, since the recoding is safely stored in FRAM.

Multiple-word commands will always have to be recognized by multiple nodes, as a single intermittent node must recharge its energy buffer after recording.

A representative sound input for a command recognition system consists of bursts of 1-4 words closely after each other, forming a command, and long periods of silence between the commands. As words can be seen as events, we chose to design an event-driven algorithm, which will await events (words), sleeping in a low-power mode, and start acting once it detects a word (Figure 3.2).

In an environment where commands are not spoken often, the availability of a CICR at the time someone starts speaking can be modeled using the duty cycles that intermittent nodes have when in low-power mode – as that is the state they will be in until that moment. The following equation models the system availability when a single node is added to the CICR:

$$a_{sys}(N) = a_{sys}(N-1) + a_{node} * (1 - a_{sys}(N-1)), \tag{3.1}$$

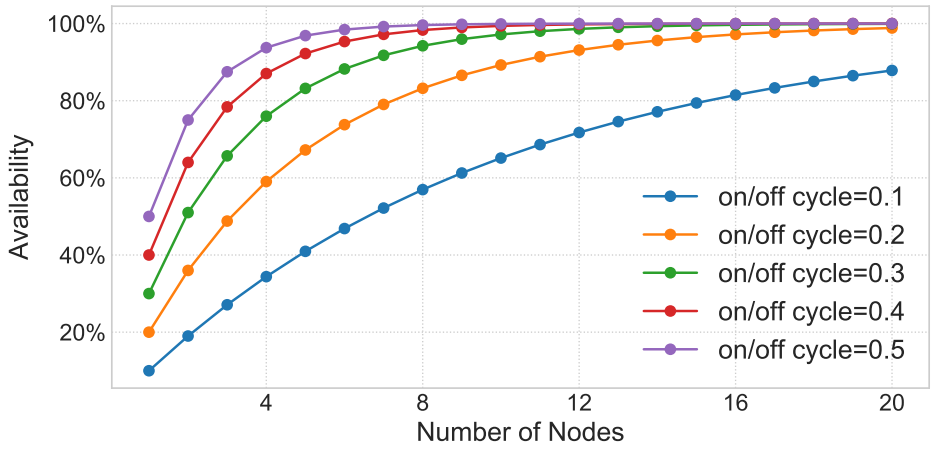


Figure 3.3: Modeled Coalesced Intermittent Command Recognizer availability percentage for different numbers of nodes and different duty cycles, using Equation (3.1)

where a_{sys} is the system availability, a_{node} is the availability of the node that is being added to the system, and N is the number of added nodes so far ($a_{sys}(0) = 0$). Figure 3.3 presents the modeled CICR availability percentage for different numbers of nodes and different duty cycles.

3.2.1 Power States

A CICR can experience a wide range of ambient power intensities. For example, a CICR may harvest no energy at night, modest energy from artificial light, and abundant energy from direct sunlight. Generally, we can identify four different powering states:

- *Targeted power state*—These are the powering conditions that the CICR is designed for. In these conditions, it should work intermittently and have sufficiently randomized power cycles to uniformly distribute its intermittent nodes' on-times and meet the desired availability percentage (Figure 3.3). In general, the targeted powering conditions should be near worst-case 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 the CICR will be under-powered or even comes to complete and long power down (for example in darkness). In general, for under-targeted energy conditions, the CICR behavior can be considered as undefined.

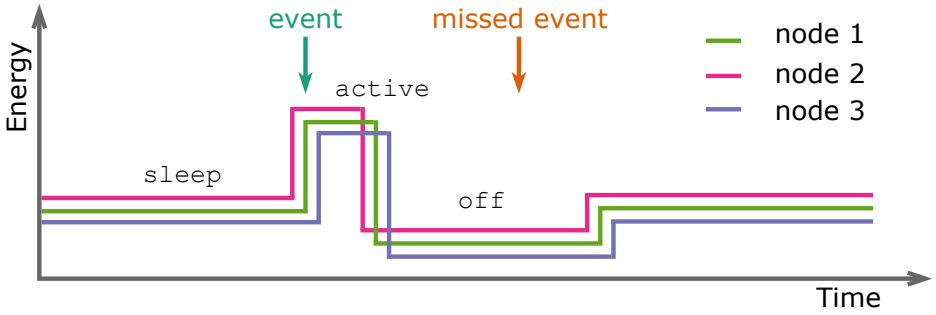


Figure 3.4: Coalesced Intermittent Command Recognizer is in a hibernating power state when the energy harvesting rate approximates the energy consumption rate at the sleeping (low-power) mode. In this state, the intermittent nodes lose the randomization between their power cycles. Thus, all the nodes record the first word and power down shortly afterwards, missing the subsequent ones. Consequently, the CICR senses intermittently and fails to take advantage of its redundant nodes.

- *Hibernating power state*—When the intermittent nodes of a CICR sleep in low-power mode waiting for a word to wake them up. If the energy conditions are significantly higher than the targeted conditions, the nodes may not die and sustain their sleeping power consumption. This will cause them to synchronize their wake-ups on the first incoming word and their powering down as the word recording process depletes their energy buffers quickly. Consequently, the CICR may miss the next incoming words (especially if the words arrive in the form of a long command) causing it to sense intermittently in stead of continuously (Figure 3.4).

- *Continuous power state*—Under direct mid-noon sun even a tiny solar panel can continuously power a sensor. In such conditions, the CICR 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 the base station, N identical messages will be sent, which wastes a lot of energy.

3.2.2 Desynchronization

The inefficiencies highlighted in the Hibernating and Continuous power states can be mitigated by desynchronizing (enforcing randomization) on the response of the intermittent nodes (Figure 3.5): when a node detects word it only starts recording with a certain probability.

For this, we generated random numbers from the least significant bits of the last recorded sound values. Depending on the random number, a node

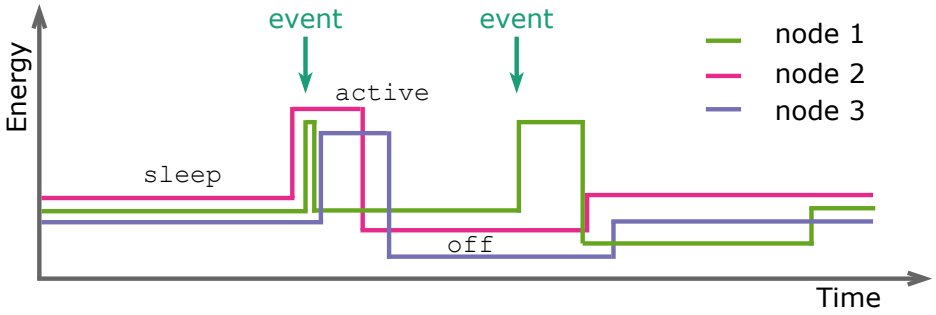


Figure 3.5: Randomized response helps in mitigating the hibernating power state problem. Also, it reduces the number of duplicated captured events when the CICR is overpowered.

will either (i) start recording as usually, or (ii) turn off the microphone, sleep for an additional 700 ms and then wait for the next voice event. At the next event a new random number is drawn to make the same decision again. The additional period of sleep (in this case 700 ms) is based on the used testing set of words (Table 4.1) and is needed to ensure that the node will not wake up from the same word, but instead will wait for the next word.

The downside of using randomness for desynchronization is that it introduces a probability that all available nodes will postpone their recording and a word will not be captured, even though some nodes were available. Furthermore, it may happen that a node that initially had enough energy to record, will not have enough energy anymore after postponing recording a couple of times.

3.3 Command recognition implementation

This section describes the used steps for performing command recognition. Figure 3.6 gives an overview of where the steps fit in the command recognition process.

In our implementation the code is always executed in a fixed order and is non-preemptive. This means that once recording is completed a node has to finish processing before it can record again.

3.3.1 Recording

We use a sample rate of 8 kHz for recording to cover the frequency range of the human voice. By studying the characteristics of the targeted vocabulary – in particular the minimum effective recording length (Section 4.1.1) – we selected a fixed recording length of 285 ms. Because we use a fixed recording length and exploit the *wake-on-sound* microphone feature, the

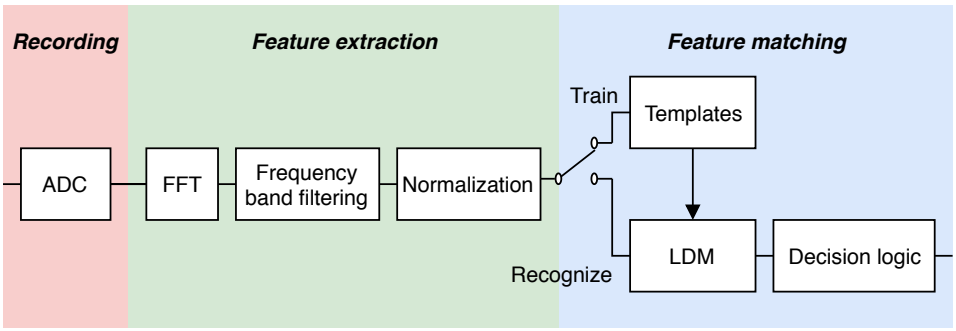


Figure 3.6: System block diagram.

word is always positioned at the beginning of the recording ($t = 0$) and endpoint detection is not needed, greatly improving the processing time and system efficiency from the energy perspective.

Once a recording has finished, framing and data processing begin. We use non-overlapping frames of 256 samples ($\approx$ 33 milliseconds). This size is beneficial for doing a Fast Fourier Transform and short enough for the voice-features to be considered constant inside one frame (see Section 2.3).

3.3.2 Feature Extraction

Next, features of the recorded voice are extracted for comparison against previously stored templates. Feature extraction takes place on a frame-by-frame basis and is based on the energy and spectral characteristics of each frame. The spectral characteristics of a frame are determined by dividing the frequency range of interest into several spectral bands. We use 12 spectral bands as in [28] (Table 3.1). Because most of the energy in human speech is contained in lower frequencies, the width of the bands is smaller for the lower frequencies and gradually increases as the frequency gets higher.

For each frame in a word, a 256-point FFT is performed. The results of the FFT are used to determine the energy in each of the 12 spectral bands. This process essentially produces a 12-element vector for each frame of the speech input. Each element of the vector represents the energy in one of the 12 spectral bands. These vectors are the basis for comparison against the previously stored templates, once they are normalized.

Normalization

Once the features are extracted from the speech input, the frames are normalized. This process helps to eliminate errors that could result from differences between the amplitude of the speech input and the previously stored template for that word. To normalize a feature vector, for each of the spectral bands the binary logarithm is taken of the energy in that band and

Table 3.1: Spectral bands used for feature extraction.

Band	Frequency range (Hz)
1	100-300
2	250-450
3	400-600
4	550-750
5	700-900
6	900-1200
7	1200-1500
8	1500-1800
9	1800-2300
10	2300-2800
11	2800-3400
12	3400-4000

the average energy logarithm of the frame is subtracted from that. This is shown in the following equation:

$$f_i = \log(\hat{f}_i) - \frac{\sum_{i=1}^{S} \log(\hat{f}_i)}{S}, \tag{3.2}$$

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

3.3.3 Feature Matching

The system identifies the recorded word by comparing the normalized features of the input signal against the features of all the templates contained in its dictionary.

As was previously discussed, these features are contained in vectors, and the comparison of these vectors is made on a frame-by-frame basis. The similarity between two frames is determined by computing the distance between them, which can be obtained by computing the squared Euclidean distance between their normalized vectors as shown in the following equation:

$$\mathrm{d}[F_T, F_R] = \sum_{i=1}^{S} (f_{T,i} - f_{R,i})^2, \tag{3.3}$$

where d is the the distance between two frames, F_T is the a template frame, F_R is the a recorded frame, $f_{T,i}$ is the normalized output of the i^{th} spectral

band of a template frame, $f_{R,i}$ is the normalized output of the i^{th} spectral band of a recorded frame and S is the number of spectral bands.

It should be emphasized that feature matching algorithms may choose different frame pairs to compare for determining word similarity. We implemented two feature matching methods: Linear Distance Matching and Dynamic Time Warping.

Linear Distance Matching

In Linear Distance Matching (LDM) the frames within the feature vectors of two words are compared successively, not accounting for differences in pronunciation speed. LDM can only compare feature vectors of equal length. The total distance between two words is calculated as follows:

$$\mathrm{d}[W_T, W_R] = \sum_{i=1}^{L} \mathrm{d}[F_{T,i}, F_{R,i}], \tag{3.4}$$

where d is the distance, W_T is a template word, W_R is a recorded word, $F_{T,i}$ is the i^{th} template frame, $F_{R,i}$ is the i^{th} recorded frame, and L is the recording length measured in frames.

Dynamic Time Warping

Dynamic time warping (DTW) is an algorithm for measuring similarity between two time sequences of numbers (representing words, in our case), which may vary in speed. Two words can be "warped" non-linearly by stretching or shrinking them along their time axis, as long as time-ordering is preserved and the boundary conditions are met: the starting and ending points of the warping path must be the first and last points of the words [57]. The process is depicted in Figure 3.7. The optimal warping path is the path with minimal global distance, and is calculated as follows:

For two words of length m and n, an accumulated cost matrix D is constructed, filling up the matrix from $i = 1, j = 1$ till $i = m, j = n$:

$$\mathrm{D}(i,j) = \mathrm{d}(i,j) + \min[\mathrm{D}(i-1,j-1), \mathrm{D}(i-1,j), \mathrm{D}(i,j-1)], \tag{3.5}$$

where $\mathrm{D}(i,j)$ is the minimal distance between two words up till frames i and j respectively, and d is the distance between two frames, as in Equation (3.3). When all values of the matrix are calculated, $\mathrm{D}(m,n)$ holds the optimal distance between two words:

$$\mathrm{d}[W_T, W_R] = \mathrm{D}(m,n). \tag{3.6}$$

We compared the performance of both LDM and DTW (Section 4.1.2). In the scope of our project (short recording time of fixed length) both methods achieve a comparable recognition accuracy, while DTW needs significantly more computation power. Therefore, we chose to use LDM for our project.

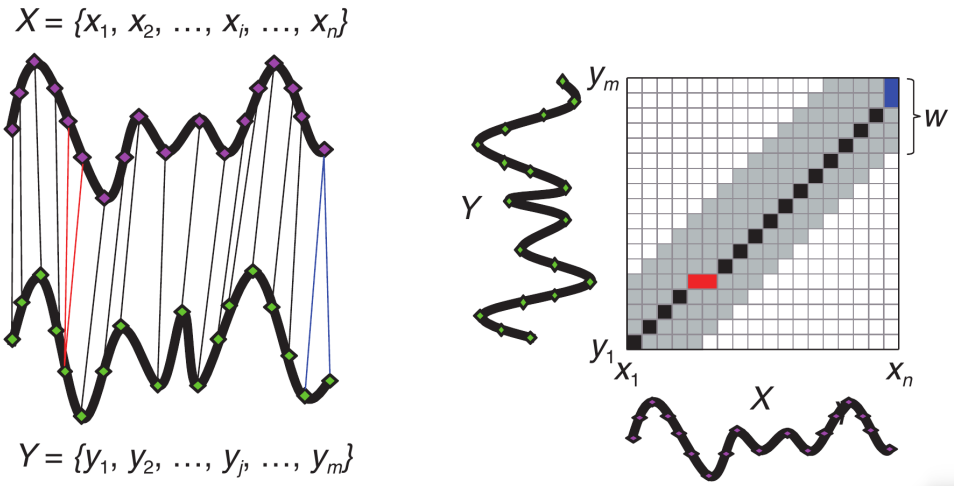


Figure 3.7: Example of a time warping process between time sequences X and Y. The warping path is shown as a black line in the array on the right. A perfectly diagonal line means no time-warping is applied. Figure from [63].

Word recognition

Once a word has been compared to all templates, the template with the smallest distance to the recorded word is considered the recognized word, unless the smallest distance exceeds a threshold (see Section 4.1.3). This means that the recorded word is not similar to any of the word templates and hence the recognizer concludes the word is unknown.

At this point, the recognized word is communicated via UART for debugging. In the future a wireless transmitter can be used to send the result to a central server, or directly to an actuator device in case of single-word commands.

3.3.4 Power-failure proofing

In order to make the code progress despite power failures, we manually inserted checkpoints into the code. The checkpoints are placed after completing each step of the process of *recording, feature extraction, feature matching*. We placed additional checkpoints inside *feature extraction* and *feature matching*, see Table 3.2. In total we use 20 checkpoints. We ensure that the execution of the code in between each two checkpoints requires less energy than what the energy buffer can provide with a single charge.

During every calculation, results that are needed past the checkpoint are stored directly into non-volatile memory. A checkpoint at the end of the calculation indicates that the calculation has finished, and that the results are stored successfully. If a power reset occurs later on, the execution will return to the last reached checkpoint.

Table 3.2: Checkpoints

Stage	Checkpoints(#)
Recording	1
Future extraction	9 (each frame)
Future matching	10 (each word)
Total	20

Table 3.3: 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

By manually placing the checkpoints, we could specify exactly which data is needed after the checkpoint and needs to be stored in non-volatile memory (results), and which data does not need to be retained during a power reset (temporary variables).

The data-flow is implemented in such a way that the input and output of a process are stored in different locations in memory, avoiding values to be overwritten if a function is restarted multiple times due to power resets (Write-After-Read dependency). This way data consistency is preserved during intermittent execution.

3.4 Code profiling

The entire command recognition software was written in the `C` programming language. The program consists of 973 lines of code in total, excluding the Texas Instrument DSP library, from which the FFT function was used. See Table 3.3 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 3.4.

The power usage of a node differs according to its 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 measured with a Monsoon power monitor and shown in Table 3.5.

Table 3.4: Profiling results for the recording and processing of 9 frames using different feature matching methods.

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

Table 3.5: Power usage with standard deviation.

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

Chapter 4

Results

First we show several experiments we did to optimize the command recognition algorithm. Next we evaluate the CICR behavior in different energy conditions and with different lengths of commands.

4.1 Design optimization

4.1.1 Minimum effective recording length

The recording length has a big influence on the system performance. Since a recording has to be done in one go, the recording length directly determines the minimum capacitor size. The recording length also affects the amount of processing that needs to be done, and the memory usage, since the length of the word templates in the dictionary needs to be (at least) as long as the recording length.

Therefore, the first experiment is targeting the minimum recording length without including a significant accuracy loss.

Table 4.1: Words in the testing set together with their duration.

Word	Duration (std. dev.) [ms]
Cancel	593 (33)
Clear	561 (28)
Edit	446 (33)
Go	441 (35)
Load	520 (31)
Off	388 (14)
On	486 (27)
Pause	596 (28)
Resume	662 (34)
Stop	533 (28)

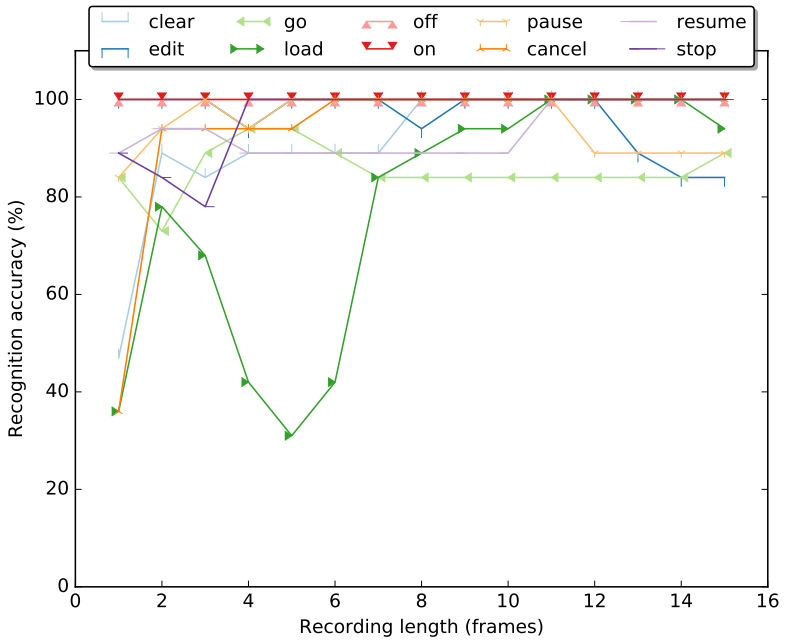


Figure 4.1: Recognition accuracy versus the recording length (in frames). Each data point is the result of 19 runs. The Linear Distance Matching algorithm was used for feature matching.

For this experiment we used a single microcontroller running on continuous power. Each word taken from Table 4.1 was recorded on a PC 20 times. We used the first recording for training and the rest of the recordings for conducting the experiment.

In Figure 4.1 the word recognition accuracy is shown when the 19 remaining recordings of each word were played back from a bluetooth speaker [32]. We see that for some words the recording length that is needed to distinguish them properly, is longer than for other words.

We conclude that recording beyond *nine frames* (285 ms) does not significantly increase the recognition accuracy for the used set of words. Therefore, we use a recording length of nine frames for the rest of the experiments. This recording length is very specific for the combination of words in our testing set and will vary for different testing sets, although on every testing set a similar experiment can be conducted to minimize the recording length. The average recognition accuracy for the words from our test set with a recording length of nine frames is 97%.

4.1.2 Comparison of feature matching methods

We compared the accuracy of the Linear Distance Matching and Dynamic Time Warping for our chosen set of words and a recording length (285 ms).

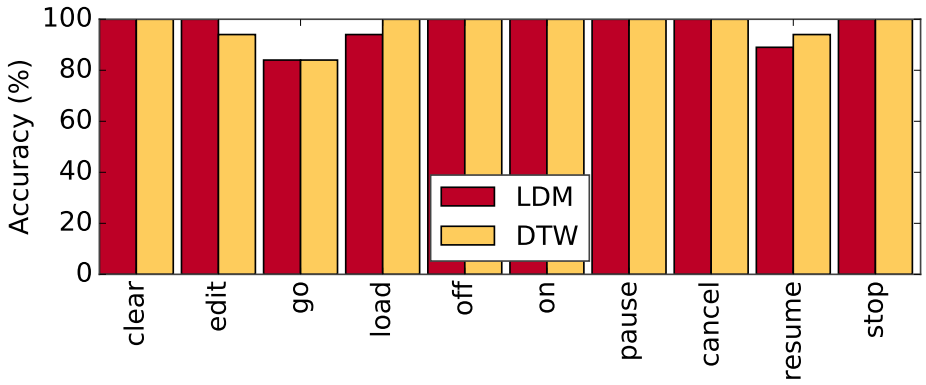


Figure 4.2: Recognition accuracy for LDM and DTW when using nine frames (285 ms) as the recording length.

Figure 4.2 shows that both methods provide similar recognition accuracy. However, the profiling results (Table 3.4) show that DTW has a longer processing time. Therefore we use LDM in future experiments.

4.1.3 Determining the recognition threshold

Our template-based command-recognition algorithm bases its result on the template in its dictionary that is most similar to the recorded word. However, when a word is recorded that is not in its dictionary, it must be able to tell it apart. To achieve this, we choose a recognition threshold. The lower the distance, the bigger the similarity between words (see also Section 3.3), therefore known words (that can be matches against a template) are expected to have a lower distance than unknown words.

First, to assess the ability of the algorithm to tell different words apart, we plotted the distances between words in a confusion matrix (Figure 4.3). On the diagonal each word is matched against itself and the 19 other recordings of the same word. In the rest of the matrix each word is matched against other words. The highest distance on the diagonal is 403. We use this value as the recognition threshold.

4.1.4 Recognition of late recordings

Each time when a node comes on-line after recharging, there is a chance that a word is being spoken at that time. A node will detect that there is a voice speaking and will start recording. It can not know, however, *when* someone started speaking and will assume it is recording from the beginning of the word. This causes a misalignment between the recording and the template. The effects of the misalignment on the recognition accuracy are

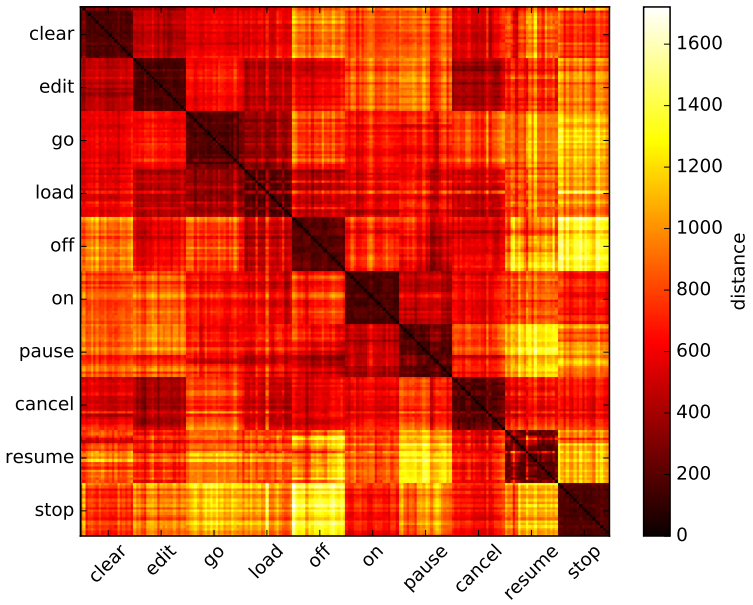


Figure 4.3: Confusion matrix, showing the distance between the words in our testing set. From every word there are twenty recordings.

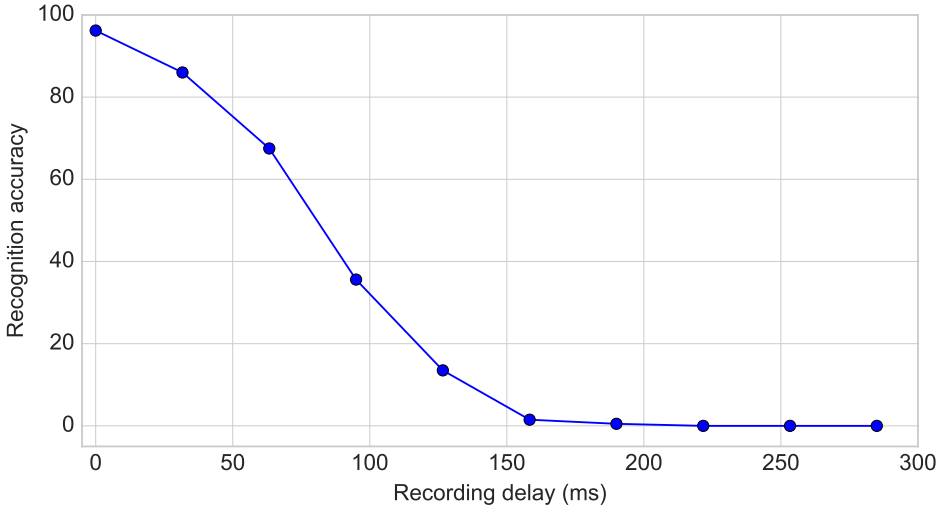


Figure 4.4: Effect of a late start of recording on the recognition accuracy.

shown in Figure 4.4. We see that with a small delay in recording, words can still be recognized, but with an increasing delay the recognition accuracy quickly decreases. This means the node used its precious energy without any contribution, and will possibly miss following words when it is recharging its energy buffer. However, we measure that late recordings only happen occasionally (Figure 4.9). Therefore, countermeasures are not needed.

4.2 Evaluation

In this section we focus on how well the CICR is able to react on spoken words (Table 4.1) and longer commands made of these words. We recorded patterns of single words as well as multiple-word commands, and varied the periods of silence in between them.

4.2.1 Experimental setup

In order to get reproducible results, we designed a testbed with steady and controllable light intensity. To achieve this, we blocked uncontrollable light sources with a box of $60 \times 40 \times 40\,\text{cm}$. 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 8 intermittent command recognizer nodes (the hardware is described in Section 3.1). We used a Bluetooth speaker [32] to replay a certain record. The data was collected using a logic analyzer [56] and processed on a laptop. The whole experimental setup is shown in Figure 4.5.

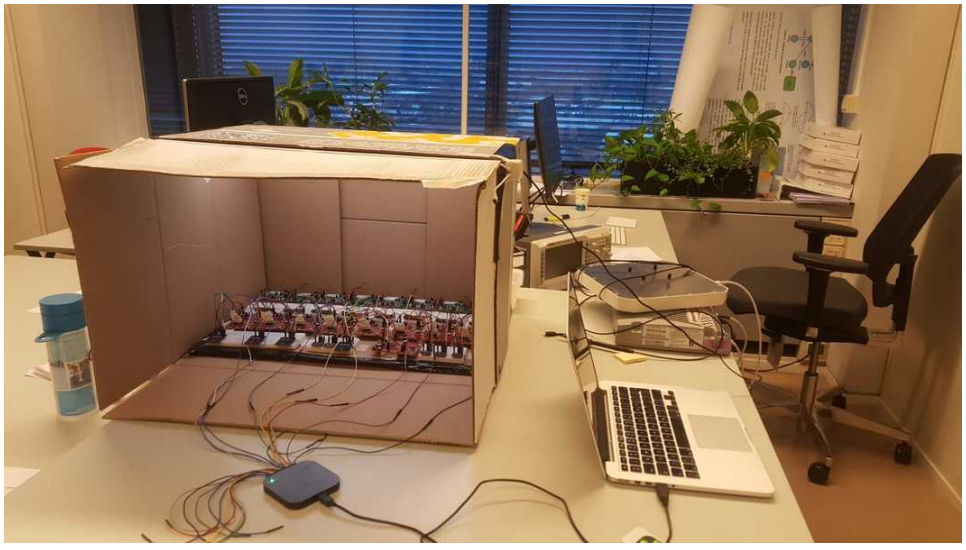


Figure 4.5: Solar testbed where uncontrollable light sources can be blocked out and a LED strip provides 15 different light intensities.

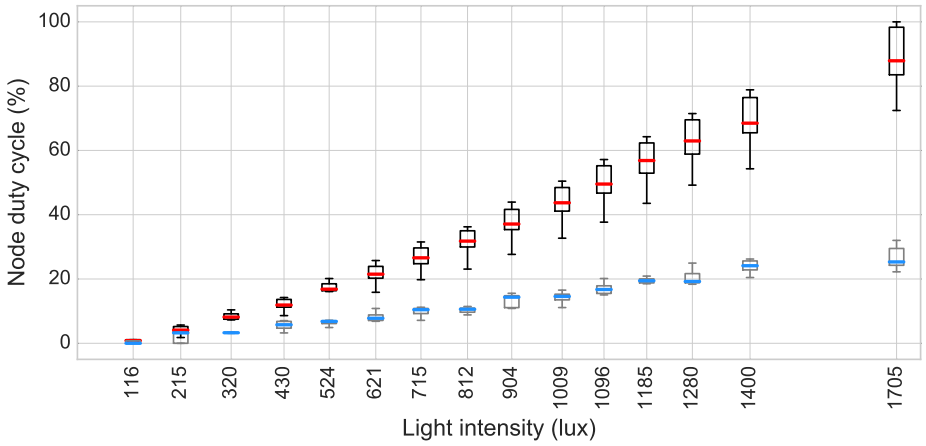


Figure 4.6: Duty cycle of an intermittently powered node for different light intensities when the node is sleeping (red) and when processing (blue).

4.2.2 Single-node duty cycle

Figure 4.6 shows the duty cycle of an intermittently-powered node when in sleep mode and when processing, for different light conditions. Due to a higher power consumption, the duty cycle is lower when a node is processing. The CICR is designed for a light intensity of about 500 lux, where it has a duty cycle of 10-20%. In these conditions, it should work intermittently and have sufficiently randomized power cycles to uniformly distribute its intermittent nodes' on-times.

4.2.3 CICR availability

In an environment where commands are not spoken often, the availability of a CICR at the time someone starts speaking is determined the duty cycles that nodes have in sleep mode – as that is the state they will be in until that moment. Figure 4.7 shows the system availability for different numbers of intermittent nodes while in sleep mode. These results show how a CICR consisting of multiple nodes can achieve a much higher availability than its individual intermittent nodes, and confirms the availability model; the dashed line matches a 15% duty cycle based on the availability model in Equation (3.1).

However, when words follow each other with little time in between, as is the case in multiple-word commands, nodes spend part of their time having a lower duty cycle, resulting in a lower – and harder to predict – CICR availability.

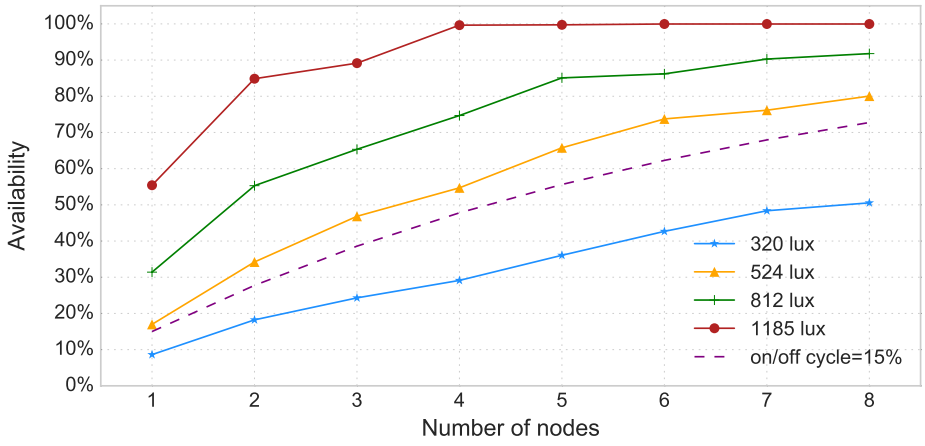


Figure 4.7: System availability for different numbers of intermittent nodes, measured under different light conditions. The dashed line matches a system availability of 15% based on the model in Equation (3.1)

4.2.4 Single word detection

First we evaluate the CICR behavior in an environment with regular words. In this experiment we varied the light intensity from 500 lux to 1700 lux and played a word repeatedly every one, two, four, and six seconds. The lowest inter-word arrival time is one second because this is slightly longer than the 885 ms that our algorithm takes when run on a continuously-powered command recognizer (see Table 3.4). This means a single continuously-powered command recognizer is able to record and process all the words at this rate.

Figure 4.8 shows the percentage of the total detected words and the uniquely detected ones. We see a positive correlation between light intensity and the average number of nodes detecting a word, caused by the increased duty cycles.

In particular, we see that the number of duplicately-detected words rises dramatically when light intensity increases, demonstrating the overpowering problem. Moreover, increasing the inter-word arrival time also increases the number of duplicated words. The reason for this phenomenon is that when the time between words increases, a bigger part of the nodes has finished processing and recharging by the time the next word comes. This brings them in the same state (low-power mode), and this reduces the inherent randomization of the intermittent nodes and leads them to the *hibernating power state* (Section 3.2.1).

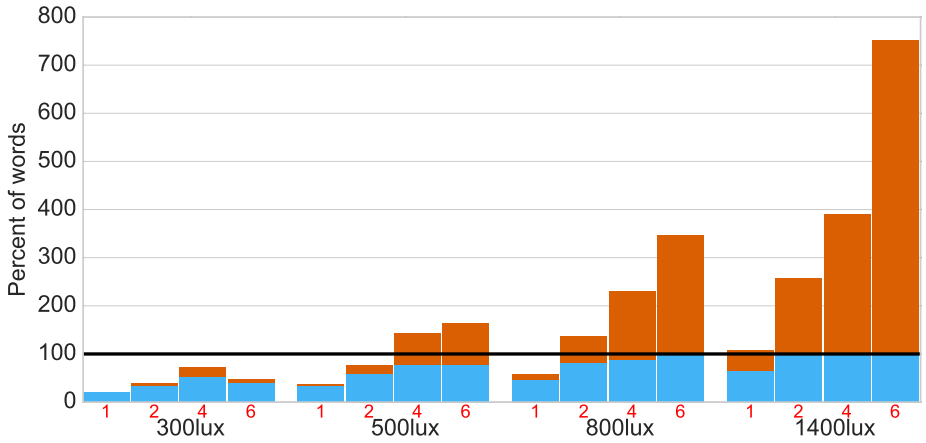


Figure 4.8: The total number of word detections (orange) and unique detected words (blue) by a coalesced intermittent command recognizer with 8 intermittent nodes. The total number of played words is 240. The red numbers indicate the word arrival interval. In general, we see that when the light intensity increases, the number of detected words rises too.

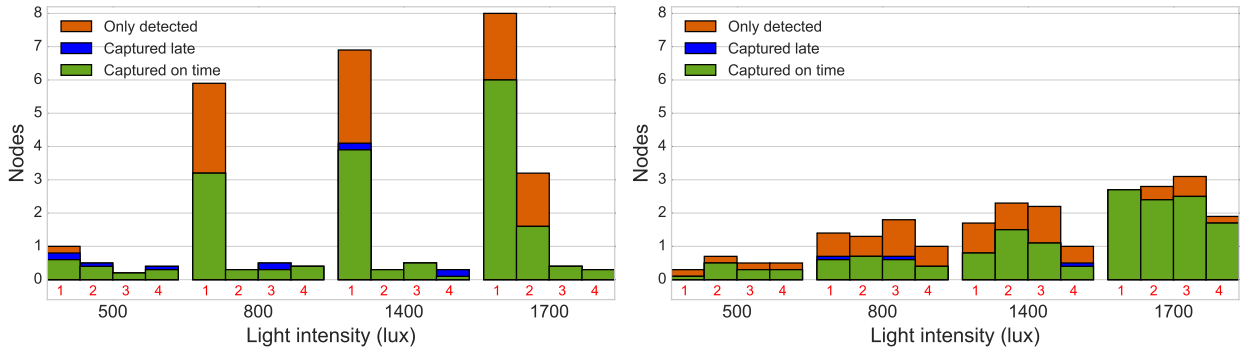
4.2.5 Long commands – detection and capture

Having seen how nodes react to regularly repeating words, we will next study the reaction on multiple-word commands separated by periods of silence.

When a node detects a sound and starts recording we count this as a **detection**. However, words are relatively long events and therefore some of their recordings do not complete due to insufficient harvested energy. When a node successfully finishes recording we say the node **captured** the word. If a node finishes charging and starts recording *during* a word, it will finish recording, but the recording will contain wrong information. We call this a **late capture**. The effects of a late capture is studied in Section 4.1.4.

Figure 4.9a shows the detection and capture rates for words inside a command. A command of four words, with one second between the start of individual words, was played repeatedly with 20 seconds of silence between the commands. Each command was repeated 10 times and for four different light intensities.

For light intensities of 800 lux and greater, we observe that the intermittent nodes react to the first word of a command and power down shortly after, missing other words in the command. This results validate our theory about the side effect of the *hibernating power state* (Section 3.2.1). 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 mitigate these problems.



(a) Plain.

(b) Explicit randomization

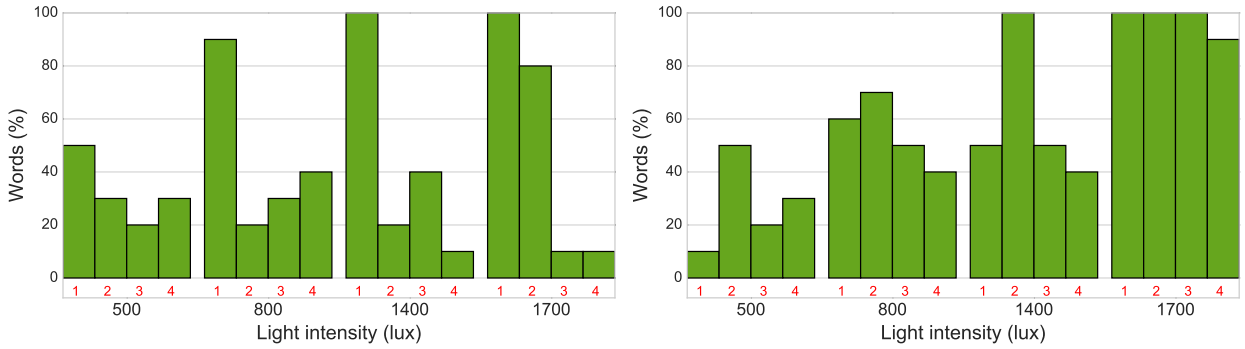
Figure 4.9: Average number of nodes reacting on a word in a four-word command. When a node detects a sound and starts recording we count this as a detection. However, words are relatively long events and therefore some of the recordings do not complete due to insufficient harvested energy (labeled 'only detected'). When a node successfully finishes recording we say the node captured the word. If a node finishes charging and starts recording *during* a word, it will finish recording, but the recording will contain wrong information. We call this a late capture.

Word detection rate with explicit randomization

To randomize during commands (which length is known in advance), a node reacts with a certain probability on a word. This probability is different for each word the node encounters after the last recharge. In order to spread the nodes over the words, the probabilities need to increase for subsequent words, since some nodes have reacted already on previous words, and therefore the number of nodes still available is smaller after each word.

We can calculate the probabilities to distribute the nodes evenly under the assumption that the nodes have a duty cycle of 1 while sleeping (*Hibernating power state*), and the nodes will only react once during a command: Since we expect four words, 25% of all nodes should react on the first word. The nodes that are left should be divided over the remaining three words, hence the (remaining) nodes should react on the second word with a probability of 33%. After that there are two words left, hence the remaining nodes should react with a probability of 50%. For the last word all the remaining nodes should react.

However, we can not use these probabilities when nodes have a lower duty cycle: when a command arrives, some nodes might be charging. Furthermore, nodes that were charging at the start of the command may become active later on. This affects the probabilities that should be used for spreading the nodes evenly. We experimentally determined the following probabilities to reasonably spread the nodes over the words: We programmed the nodes



(a) Plain.

(b) Explicit randomization.

Figure 4.10: Percentage of words inside a four-word command that are captured by the CICR.

to react with a probability of 40% on the first word, with 50% on the second word, 70% on the third word and 100% on the fourth word.

Figure 4.9b shows that a command recognizer with randomized response spreads its resources – as compared to Figure 4.9a.

4.2.6 CICR word coverage

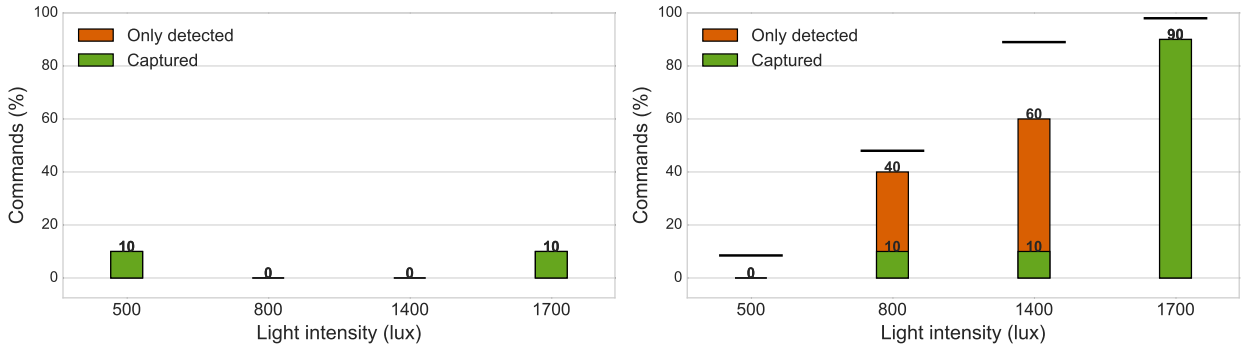
For a coalesced system as a whole, to capture a word, it is sufficient that at least one of the nodes captures the word. Therefore we analyze the previous experiments with four-word commands again, both plain and with explicit randomization, to see how often a word was captured by at least one node.

Figure 4.10 shows that explicit randomization helps to increase the capture percentage of second, third and fourth words in a command, and that in total more words are captured with explicit randomization for light intensities of 800 lux and greater.

For a light intensity of 500 lux, however, the additional randomization decreases the total number of captured words in a command. This happens because at this light intensity there is no additional benefit from explicit randomization – ambient randomness already spreads the nodes quite well – while postponing a recording comes with the risk of a node depleting its energy buffer in the meantime.

4.2.7 CICR command coverage

In case of having commands longer than one word, it is important for the system to capture all of the words of a command: an incomplete command is meaningless. Figure 4.11 shows the percentage of commands where all the four words were detected (orange) and the percentage where all the



(a) Plain.

(b) Explicit randomization.

Figure 4.11: Effect of explicit randomization on four-word command detection (orange) and capture (green). The horizontal lines in Figure (b) represent the theoretical upper bound computed from the CICR availability at a given light intensity and an ideal spreading of two nodes per event.

four words were captured (green). This last percentage effectively is the availability of the CICR for commands consisting of four words.

The horizontal lines in Figure 4.11b indicate an upper limit for the command detection rate for an ideal spreading of two nodes per word at a given light intensity. The limit is calculated by (i) taking a single node's sleeping duty cycle at a given light intensity from Figure 4.6, (ii) calculating the CICR availability while sleeping (silent environment) with Equation (3.1) for each word in the command – taking into account how many nodes are left unoccupied; 8 for the first word, 6 for the second, etc. – and (iii) multiplying those availabilities.

We see that without randomization hardly any words are captured or even detected. Explicit randomization greatly improves the percentage of detected commands for light intensities of 800 lux and above. This means that nodes are successfully *distributed* over the four-word command. However, we see that for light intensities of 800 and 1400 lux nodes almost never have enough energy to finish recording, causing the percentage of captured commands to stay low. We propose a solution for this problem as a direction for future work (Section 5.2).

Chapter 5

Conclusions and Future Work

5.1 Conclusions

We developed a command-recognition algorithm for intermittently-powered devices and built a prototype of a battery-less coalesced intermittent command recognizer. From the evaluation we conclude that it is possible to run a simple command-recognition algorithm on ultra-low-power hardware used for battery-less sensors. Our algorithm achieves a recognition accuracy of 97% for a ten-word vocabulary. Furthermore we conclude that multiple battery-less command recognizers distribute themselves uniformly in time for low light conditions.

An important finding is that favorable energy conditions may cause sleeping intermittent nodes to synchronize their power cycles on the arrival of the first word. Consequently, they react to the same word, start recharging at the same time, and miss the next word. To counter this unwanted behavior we use a probabilistic algorithm to postpone node reactions. We show that a coalesced intermittent command recognizer using this algorithm is able to distribute its nodes over a four-word command, but often the nodes have insufficient energy left in their buffer by the time they start recording. We propose a solution for this problem in the next section.

5.2 Future Work

Intermittent sensors are a relatively new research topic. Till now they have been considered as devices that work separately, used to sense slow-changing properties. Although this work presents the first Coalesced Intermittent Command Recognizer, there is still much research that can be done to improve it. We propose the following topics that we feel will improve the CICR performance or extend its functionality.

- **Speech recognition on coalesced intermittent devices** In this thesis we have shown the feasibility of speech recognition on intermittent power. However, our implementation targets the simplest type of speech – isolated words – and is speaker dependent. A next attempt may target a more complicated type of speech, understand multiple speakers, and support a larger number of words than the 10 words chosen for this study.

- **Buffer energy estimation** A node's availability could further be improved by eliminating unfinished recordings. This can be achieved by (i) enabling a node to estimate how much energy there is left in its energy buffer, and – once there is not enough energy left to record – (ii) triggering an early power-off and recharge. The buffer energy estimation can be achieved by either directly measuring the voltage over the capacitor or by estimating the energy environment (e.g. based on previous on-time) and timing how long the node has been on since the last recharge.

- **Preemptive execution** Our current implementation first needs to finish processing before it can record again. By making the algorithm preemptive, a node will be able to record a new word even when it is still processing the previous word, and finish the processing later. If the processing of a word is preempted too often, however, the word that was being processed in the first place may become outdated and loose its relevance. Also, recording new words before the old ones are processed will obviously have a bigger memory footprint.

- **Node distinction** Future designs could make use of the fact that a coalesced system consists of multiple nodes spaced across a room. Spacial information of individual nodes could be used additionally to the commands. This way a coalesced system will know e.g. where an additional light should go on. Also, not all the nodes need to necessarily have the same functionality. For example, a part of the nodes could support a different word vocabulary to extend the total system word vocabulary.

Bibliography

[1] Omid Ardakanian, Arka Bhattacharya, and David Culler. Non-Intrusive Techniques for Establishing Occupancy Related Energy Savings in Commercial Buildings. In *Proc. BuildSys*, Palo Alto, CA, USA, nov 2016. ACM.

[2] Mehdi Assefi, Guangchi Liu, Mike P Wittie, and Clemente Izurieta. An Experimental Evaluation of Apple Siri and Google Speech Recognition. *Proccedings of the 2015 ISCA SEDE*, pages 1–6, 2015.

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

[4] Saurav Bandyopadhyay and Anantha P. Chandrakasan. Platform Architecture for Solar, Thermal, and Vibration Energy Combining With MPPT and Single Inductor. *IEEE Journal of Solid-State Circuits*, 47(9):2199–2215, sep 2012.

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

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

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

[8] Gregory Chen, Hassan Ghaed, Razi M Haque, Michael Wieckowski, Yejoong Kim, Gyouho Kim, David Fick, Daeyeon Kim, Mingoo Seok, Kensall Wise, David Blaauw, and Dennis Sylvester. A Cubic-Millimeter Energy-Autonomous Wireless Intraocular Pressure Monitor. In *Proc. ISSCC*, San Francisco, CA, USA, 2011. IEEE.

[9] Octavian Cheng, Waleed Abdulla, and Zoran Salcic. Hardware-software codesign of automatic speech recognition system for embedded real-time applications. *IEEE Transactions on Industrial Electronics*, 58(3):850–859, 2011.

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

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

[12] Alexei Colin, Emily Ruppel, and Brandon Lucia. A Reconfigurable Energy Storage Architecture for Energy-harvesting Devices. *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS '18*, pages 767–781, 2018.

[13] Alexei Colin, Alanson P. Sample, and Brandon Lucia. Energy-interference-free system and toolchain support for energy-harvesting devices. In *2015 International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*, pages 35–36. IEEE, oct 2015.

[14] Samuel DeBruin, Bradford Campbell, and Prabal Dutta. Monjolo: An Energy-harvesting Energy Meter Architecture. In *Proc. SenSys*, Rome, Italy, nov 2013. ACM.

[15] Brian Delaney, Nikil Jayant, Mat Hans, Tajana Simunic, and Andrea Acquaviva. 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*, volume 1, pages I—-793. IEEE, 2002.

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

[17] Carlo Delle Donne. Wake-up alignment for batteryless sensors with zero-energy timekeeping. master's thesis, 2018.

[18] Sadaoki Furui. *Speech Technology*. Springer, 2010.

[19] S. Furui, T. Kikuchi, Y. Shinnaka, and C. Hori. Speech-to-Text and Speech-to-Speech Summarization of Spontaneous Speech. *IEEE Transactions on Speech and Audio Processing*, 12(4):401–408, jul 2004.

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

[21] Shyamnath Gollakota, Matthew Reynolds, Joshua Smith, and David Wetherall. The Emergence of {RF}-Powered Computing. *Computer*, 47(1):32–39, 2014.

[22] Maria Gorlatova, John Sarik, Guy Grebla, Mina Cong, Ioannis Kymissis, and Gil Zussman. Movers and Shakers: Kinetic Energy Harvesting for the Internet of Things. In *Proc. SIGMETRICS*, Austin, TX, USA, 2014. ACM.

[23] Josiah Hester, Timothy Scott, and Jacob Sorber. Ekho. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems - SenSys '14*, pages 1–15, New York, New York, USA, 2014. ACM Press.

[24] Josiah Hester, Lanny Sitanayah, and Jacob Sorber. Tragedy of the Coulombs. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems - SenSys '15*, pages 5–16, New York, New York, USA, 2015. ACM Press.

[25] Josiah Hester and Jacob Sorber. Flicker: Rapid Prototyping for the Batteryless Internet-of-Things. In *Proc. SenSys*, pages 1–13, Delft, The Netherlands, nov 2017. ACM.

[26] Josiah Hester, Nicole Tobias, Amir Rahmati, Lanny Sitanayah, Daniel Holcomb, Kevin Fu, Wayne P. Burleson, and Jacob Sorber. Persistent Clocks for Batteryless Sensing Devices. *ACM Transactions on Embedded Computing Systems*, 15(4):1–28, aug 2016.

[27] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara

Sainath, and Brian Kingsbury. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine*, 29(6):82–97, nov 2012.

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

[29] Xuedong. Huang, Alejandro. Acero, and Hsiao-Wuen. Hon. *Spoken language processing : a guide to theory, algorithm, and system development.* Prentice Hall PTR, 2001.

[30] D. Huggins-Daines, M. Kumar, A. Chan, A. W. Black, M. Ravishankar, and A. I. Rudnicky. PocketSphinx: A free, real-time continuous speech recognition system for hand-held devices. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 185–188, 2006.

[31] IXYS Corporation. IXOLAR™ high efficiency SLMD121H04L solar module, 2018.

[32] JBL. JBL Go+ bluetooth speaker, 2019.

[33] B H Juang and Lawrence R Rabiner. Automatic Speech Recognition - A Brief History of the Technology Development. *Elsevier Encyclopedia of Language and Linguistics*, pages 1–24, 2005.

[34] Hiroaki Kokubo, Nobuo Hataoka, Akinobu Lee, Tatsuya Kawahara, and Kiyohiro Shikano. Embedded julius: Continuous speech recognition software for microprocessor. *2006 IEEE 8th Workshop on Multimedia Signal Processing, MMSP 2006*, pages 378–381, 2007.

[35] Minjae Lee, Kyuyeon Hwang, Jinhwan Park, Sungwook Choi, Sungho Shin, and Wonyong Sung. FPGA-based low-power speech recognition with recurrent neural networks. *IEEE Workshop on Signal Processing Systems, SiPS: Design and Implementation*, pages 230–235, 2016.

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

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

[38] Brandon Lucia, Benjamin Ransford, Brandon Lucia, and Benjamin Ransford. A simpler, safer programming and execution model for intermittent systems. *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation - PLDI 2015*, 50(6):575–585, 2015.

[39] Amjad Yousef Majid, Michel Jansen, Guillermo Ortas Delgado, Kasım Sinan Yıldırım, and Przemysław Pawełczak. Multi-hop backscatter tag-to-tag networks. *arXiv preprint arXiv:1901.10274*, 2019.

[40] Robert Margolies, Maria Gorlatova, John Sarik, Gerald Stanje, Jianxun Zhu adn Paul Miller, Marcin Szczodrak, Baradwaj Vigraham, Luca Carloni, Peter Kinget, Ioannis Kymissis, and Gil Zussman. Energy-Harvesting Active Networked Tags ({EnHANTs}): Prototyping and Experimentation. 11(4):62:1—-62:27, nov 2015.

[41] Robert Margolies, Guy Grebla, Tingjun Chen, Dan Rubenstein, and Gil Zussman. Panda: Neighbor Discovery on a Power Harvesting Budget. In *Proc. INFOCOM*, San Francisco, CA, USA, apr 2016. IEEE.

[42] T B Martin, A L Nelson, and H J Zadell. Speech recognition by feature-abstraction techniques, 1964.

[43] Binu Mathew, Al Davis, and Zhen Fang. A Low-Power Accelerator for the Sphinx 3 Speech Recognition System. *Proceedings of the International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, page 210, 2003.

[44] Ian McLoughlin and Hamid Reza Sharifzadeh. Speech recognition for smart homes. In *Speech Recognition*. InTech, 2008.

[45] Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88, jan 2002.

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

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

[48] PUI audio, vesper. Pmm-3738-vm1010-r: A zeropower listening piezoelectric mems microphone, 2019.

[49] He Qiang and Zhang Youwei. Prefiltering and Endpoint Detection. *Proceedings of ICSP*, (i):749–752, 1998.

[50] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[51] Lawrence R Rabiner, Biing-Hwang Juang, and Janet C Rutledge. *Fundamentals of speech recognition*, volume 14. PTR Prentice Hall Englewood Cliffs, 1993.

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

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

[54] Benjamin Ransford, Jacob Sorber, Kevin Fu, Benjamin Ransford, Jacob Sorber, Kevin Fu, Benjamin Ransford, Jacob Sorber, Kevin Fu, Benjamin Ransford, Jacob Sorber, and Kevin Fu. Mementos. In *Proceedings of the sixteenth international conference on Architectural support for programming languages and operating systems - ASPLOS '11*, volume 39, page 159, New York, New York, USA, 2011. ACM Press.

[55] V S Rao. Ambient-Energy Powered Multi-Hop Internet of Things. 2017.

[56] Saleae. Logic 16 analyzer. `http://www.saleae.com`, 2017. Last accessed: Jul. 28, 2017.

[57] Pavel Senin. Dynamic time warping algorithm review. *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA*, 855:1–23, 2008.

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

[59] Ivan Stoianov, Lama Nachman, Sam Madden, and Timur Tokmouline. PIPENET: A Sireless Sensor Network for Pipeline Monitoring. In *Proc. IPSN*, Cambridge, MA, USA, apr 2007. ACM/IEEE.

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

[61] Texas Instruments. Ultra low power management IC, boost charger nanopowered buck converter evaluation module, 2018.

[62] Texas Instruments. MSP430FR5994 16 MHz ultra-low-power microcontroller product page, 2019.

[63] MacHiko Toyoda and Yasushi Sakurai. Subsequence matching in data streams. *NTT Technical Review*, 11(1), 2013.

[64] Joel Van Der Woude and Matthew Hicks. Intermittent Computation Without Hardware Support or Programmer Intervention. In *OSDI*, pages 17–32, Savannah, GA, USA, nov 2016. ACM.

[65] Vesper. ZeroPower Listening FAQ.

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

[67] J. G. Wilpon and L. R. Rabiner. Application of hidden Markov models to automatic speech endpoint detection. *Computer Speech and Language*, 2(3-4):321–341, 1987.

[68] Kasm Sinan Yldrm, Amjad Yousef Majid, Dimitris Patoukas, Koen Schaper, Przemyslaw Pawelczak, and Josiah Hester. InK. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems - SenSys '18*, pages 41–53, New York, New York, USA, 2018. ACM Press.

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

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

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

Appendix A

Microphone PCB design

Due to the small dimensions of the microphone chip, we had to design a custom printed circuit board in order to connect it to the microcontroller. Design and result are shown in Figure A.1.

The sound comes in trough a hole in the bottom of the chip, therefore the PCB also contains a hole for the sound input.

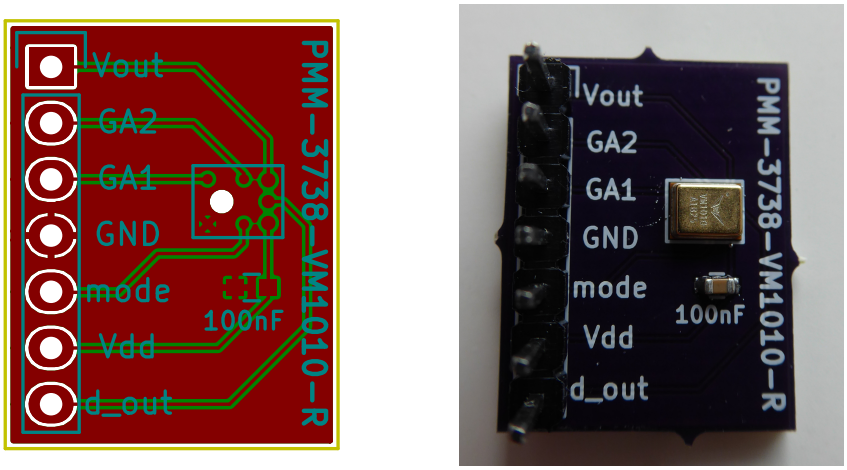


Figure A.1: Footprint and PCB for the microphone.