MASTER'S THESIS 2024

Segmenting a power consumption profile for approximating battery behavior

Patrik Gyllvin, Maria Svensson

ISSN 1650-2884
LU-CS-EX: 2024-24

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2024-24

Segmenting a power consumption profile for approximating battery behavior

Segmentering av en strömförbrukningsprofil som approximativ batteri-last

Patrik Gyllvin, Maria Svensson

Segmenting a power consumption profile for approximating battery behavior

Patrik Gyllvin
pa6514gy-s@student.lu.se

Maria Svensson
ma4383sv-s@student.lu.se

June 5, 2024

Master's thesis work carried out at Qoitech AB.

Supervisors: Jonas Skeppstedt, jonas.skeppstedt@cs.lth.se
Björn Rosqvist, bjorn.rosqvist@qoitech.com
Andreas Olausson, andreas.olausson@qoitech.com

Examiner: Flavius Gruian, flavius.gruian@cs.lth.se

Abstract

This thesis aims to analyze how a power consumption profile for a battery-powered device can be segmented into a number of different sized steps while keeping the most important characteristics for preserving the behavior of the battery under a load of said profile. This approximation can be used to emulate the device. It attempts to investigate what those important characteristics are and how they can be preserved using proposed algorithms. Peaks in the profile in particular were deemed of significant meaning because of how batteries react to high peak loads. Our algorithms aim to recreate peaks as closely as possible given the constraints of the segmentation, and our analysis is focused on the effect these have.

All of the algorithms tested came within 1.7 % of the total energy of the original power profile at the voltage cutoff. Results show that for the battery tested, peak reconstruction mostly matters in order to reach the voltage cutoff at the expected time. This is due to the battery becoming more peak sensitive when discharged. It is also important that the energy of the profile, i.e. the integral of the power consumption, is retained by the algorithms to make an accurate battery life estimation.

Keywords: non-uniform, segmentation, peak detection, peak reconstruction, peak sensitivity, battery

Acknowledgements

We would like to extend our gratitude to Qoitech for their dedication and support. Special thanks to Björn Rosqvist for his invaluable expertise on batteries, and always being there to lend a helping hand. Additional thanks to Andreas Olausson for his detailed technical advice and active involvement in our brainstorming sessions.

We are also very grateful to Jonas Skeppstedt at Lunds Tekniska Högskola for his help with this thesis. He is an ever-present provider of feedback and encouragement. He has been valuable not only to this thesis, but has also given multiple insightful courses that have been enriching and beneficial for our future careers.

Contents

Chapter 1
Introduction

> *The Internet of Things has the potential to change the world, just as the Internet did. Maybe even more so.*
>
> —Kevin Ashton, That 'Internet of Things' Thing [1].

Internet of Things, IoT in short, was coined by Kevin Ashton in 1999 [1] and has since grown into a well-known term in the IT community. As of 2022 there were more than 14 billion connected IoT devices, and was forecasted to grow another 16% to 16.7 billion year 2023 [12]. That is approximately twice the world population [4]. IoT devices are of great importance in many industries to improve convenience and make certain processes more efficient. One example is the presence of IoT devices in agriculture, commonly referred to as Smart Agriculture [5], for instance smart sensors out in the fields measuring soil moisture, temperature and humidity. These could be stationed in remote locations, and great numbers. The devices are often powered by a battery, rendering battery selection a process of high significance, since the battery life could have a huge impact on device longevity and cost. The cost does not only include the price of the actual battery. It could also render great expenses considering human labor to replace all the batteries out in the field since there might be many sensors covering a very big area.

1.1 Motivation

Today there exist evaluations on different battery types for different IoT applications [6]. However, there is a lack of tools to actually try different types of batteries for your IoT device, which are easy and accessible not only to electrical and hardware engineers, but to firmware and software developers as well. This is the void we intend to fill by means of this study. The objective is to imitate the behavior of an IoT device in terms of power consumption and identify how different features of the curve affect how energy is consumed from a physical battery. Furthermore, the aim is to streamline the process of finding a suitable battery for

the device, while only using a prerecorded power consumption profile without needing the actual device.

1.2 Execution

Due to hardware limitations in the target setup, there is a limit of 1000 data points available for simulating the power consumption of an IoT device, as described in more detail in Section 2.2.2. Consequently, the power consumption curve of the device has to be segmented, i.e. split or grouped into a number of parts or segments. Since some parts of the curve contain more information than others, and because they have different effects on the battery, the most relevant data points to keep must be carefully selected while not exceeding the limitation in the number of data points. As a consequence of removing data in some places from the input while preserving it in others, the output from the algorithm is a non-uniform piecewise constant step function of the power consumption curve. An example of what this could look like is displayed in Figure 1.1. Different algorithms for segmenting the power consumption curve are explored with the goal of preserving how the battery is affected by the load. An algorithm is evaluated with metrics regarding the terminal voltage of the battery and energy consumption over time.

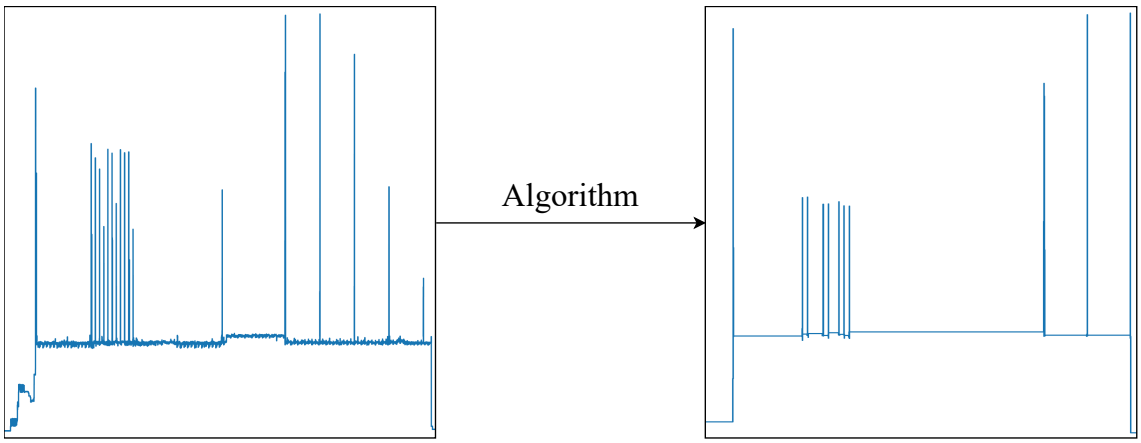


Figure 1.1: Expected behavior of the algorithm, taking input data through segmentation while keeping some of the characteristics.

1.3 Research questions

- RQ1: What would an algorithm generating a non-uniform piecewise constant approximation of a power consumption profile look like?
- RQ2: How should the error of the resulting discharge curve be calculated in order to evaluate whether an approximation is satisfactory?

1.4 Division of work

The authors of this thesis have closely collaborated on the work while dividing some parts equally between the two. While some parts of the report were the greater reponsibility of one author, all parts were checked, corrected and commented on by both. To enable the work done in the thesis, some tooling had to be developed or extended for our needs. This work was also performed collaborately using ideas and suggestions from both authors, with the responsibilities of different tools being divided between the authors based on area of expertise.

In developing the tooling, Maria took lead in extending some tooling of the company to suit our study in running battery scripts. She also implemented new tools for analyzing our algorithms. Patrik extended company tools to facilitate exporting and analyzing it faster, in addition to creating new tools for collecting the data for our study. The creation of our algorithms within the framework we had created was then a simple process, where both were involved in contributing with their ideas for how the algorithms should work while implementing them collaborately.

For the report, Maria had more responsibility for the Implementation, Hardware limitations and consequences and the Results sections. Patrik had responsibility over the Background and Discussion sections. Other parts of the report were written as a joint collaboration.

Chapter 2
Background

2.1 Theory

2.1.1 Batteries

A battery is a device with the ability to convert chemical energy to electric energy [11]. Batteries are often described by their capacity in terms of the number of ampere-hours they can deliver. They are also rated using their nominal voltage, which is the typical voltage during operation [11]. The nominal voltage is dependent on the chemistry used, as well as other properties of the battery. The capacity of the battery is related to the amount of energy stored in the battery by its nominal voltage. Energy is measured in watt-hours and is calculated as $E = V \cdot C$, where V is the nominal voltage and C is the capacity of the battery.

Simple battery model

The simplest model of a battery consists of a voltage source, open circuit voltage (OCV), and an equivalent series internal resistance (R_i) that models some of the properties of the battery as one resistance [3], shown in Figure 2.1. This follows from Thevenin's theorem, in which a linear circuit of any complexity can be modeled as a circuit of one voltage source and a series of resistances and is considered equivalent. The terminal voltage (V_t) measured on an open circuit, where no current is flowing through the battery, is the open circuit voltage ($V_t = OCV$). In a closed circuit, there is a current I flowing through the battery. When a current flows through a resistance, there is a voltage drop over that resistance given by Ohm's law ($V = R \cdot I$). Thus when measured over a closed circuit, the terminal voltage is reduced by the voltage drop over the internal resistance and becomes $V_t = OCV - R_i \cdot I$, where I is the current flowing through the circuit. As the battery is discharged, its internal resistance increases while the OCV decreases, and as a result the terminal voltage drops [11].

These effects essentially mean that a graph of the voltage and current over time will look

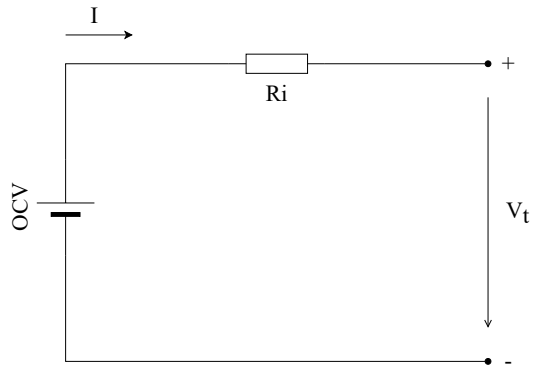


Figure 2.1: Simple battery model.

very similar but mirrored as seen in Figure 2.2.

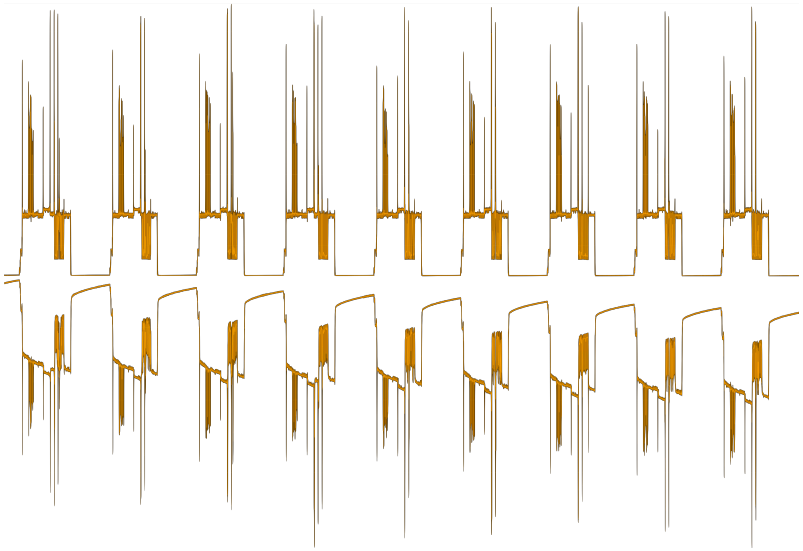


Figure 2.2: Measured current load on a battery (top) and the terminal voltage of the battery(bottom).

As this simple battery model helps explain the voltage drops during peaks and over time, it is sufficient enough to use in our case to show the effect peaks may have on battery performance. There are however some other properties such as the battery recovering, i.e. the terminal voltage increasing slightly again after a high load has been removed, that are not taken into account by this model [3].

Battery chemistries

Current peaks will momentarily affect the voltage as it quickly dips down and as the battery is discharged, the internal resistance increases further decreasing the voltage. The overall effect of peaks on a battery varies depending on the chemistry used [7]. Some battery chemistries are more sensitive to pulses than others and they react differently. For example, a button cell battery might get damaged internally by a short current pulse of high amplitude while a lithium polymer pouch cell battery simply has to recover after the pulse. It is clear, how-

ever, that short high current pulses affect the battery voltage greater than a slightly lower average continuous load [7], which is why the effect of peaks has been examined in our study. The datasheet for a certain battery oftentimes has information regarding its max discharge current when under continuous or momentary load.

Batteries with a Lithium-ion (Li-ion) chemistry are a common choice when developing IoT devices. It has advantages compared to other alternatives such as high efficiency while charging and discharging, low self discharge and high specific energy and power, which is stored energy per unit of mass and delivered power per unit of mass, respectively [9]. Another alternative for IoT devices is Lithium-ion Polymer batteries, LiPo in short, which have recently grown in popularity. They have higher energy density than traditional Li-ion batteries [11]. Furthermore, it has a flexible form factor, meaning it does not share the constrain of traditional Li-ion batteries in terms of shape and size, since they often have constructions with steel or aluminum [11]. This makes it suitable for applications requiring high portability in terms of light weight or smaller size of the battery.

2.1.2 DC-DC converters

In order to accuratly imitate a device, some characteristics of its consumption need to be considered. Loads from an IoT device can behave differently depending on the hardware, either as constant current or constant power. The hardware component with the biggest impact on battery powered devices is the DC-DC converter. When trying to imitate a device, knowledge of its behavior as a load is important to achieve the most accurate representation.

Battery-powered devices often contain at least one DC-DC converter to regulate the voltage from the battery to a level that is required by the components internally [8]. As the battery gets depleted, its terminal voltage decreases as mentioned in Section 2.1.1. The components of devices, however, require a steady voltage which may be much lower than that of the battery when fully charged. A DC-DC converter reduces the voltage from its input to its output to get it in line with the requirements of the device.

There are many different kinds of DC-DC converters, one of which is the buck converter. Buck converters are high-efficiency DC-DC converters that conserve power from their input to output by increasing voltage while reducing current [19]. Conserving the power makes buck converters highly efficient compared to other DC-DC converters for example linear voltage regulators such as low-dropout regulators (LDOs), which dissipate the excess power as heat [20]. For buck converters, current consumption increases as the battery terminal voltage decreases. In essence, buck converters conserve power on a constant level, while LDOs maintain the current consumption of the devices [8].

2.2 Otii Ace Pro

In conducting this study we will use a tool called Otii Ace Pro [15], below referred to as Ace. It is an instrument that can source or sink voltage and current while measuring simultaneously, and make a recording of the measurement over time. Since power is the product of voltage and current, and power over time is energy, Ace can also measure the energy consumption of some device under test (DUT) over time.

Figure 2.3: Otii Ace Pro by Qoitech.

2.2.1 Overview and limitations

According to the device specifications [15], seen in Table 2.1, Ace can measure up to 5 amperes with an accuracy of ±(0.05% + 25 nA) and a resolution of 0.4 nA. The voltage measurement has an output voltage read-back resolution of 3.5 µV and an accuracy of ±(0.01% + 1 mV). Ace can create cycles of load with constant current or constant power over set periods of time with an accuracy of less than 0.1 mA and a minimum step length of 1 ms. It has support for device-to-device communication using the UART protocol through the RX and TX expansion port pins, as seen in Figure 2.3. When used as a measurement tool, Ace has an internal sample rate of 250 ksps, outputting at 1 sps through 50 ksps after downsampling in real time.

Table 2.1: Specifications for Otii Ace Pro.

General	Specification
Internal sample rate	250 ksps
Current measurement	
Accuracy (0-5 A)	±(0.05% + 25 nA)
Resolution	0.4 nA
Voltage measurement	
Accuracy	±(0.01% + 1 mV)
Resolution	3.5 µV

2.2.2 Battery scripting

Ace is able to execute scripts consisting of a maximum of 1000 discharge steps of constant current or constant power. Each step has an exit condition specifying the condition at which the script should continue to the next step and forms part of an outer loop that runs for a set number of iterations or until a global exit condition is met. The exit conditions may be a number of iterations, a cutoff value or a duration. Duration exit conditions have a resolution of 1 ms and the set current has an accuracy of less than 0.1 mA. Since the hardware is not

infinitely fast the requested current or power sink might take some time to be achieved, depending on what is currently set and what is requested. When battery scripting is performed with constant power steps, it is in fact the current that is regulated according to the power equation $P = V \cdot I$ where V is the voltage and I the current. Ace measures the voltage and regulates the current continuously by calculating $I = \frac{P_{requested}}{V_{measured}}$.

2.2.3 4-wire sensing

Ace has support for 4-wire sensing, also known as four-terminal sensing or Kelvin sensing. This enables measuring both current and voltage simultaneously using two force wires and two sense wires. Doing so makes the voltage drop across the sense leads negligible as they carry a very small amount of current. 4-wire measurements are done on Ace by putting it in 4-wire mode and utilizing the Sense+ and Sense- expansion port pins for the sense wires, and the main ports for the force wires. All ports are visualized in Figure 2.3.

2.3 Particle Argon

This study is done using a consumption curve from a device called Argon from Particle, seen in Figure 2.4. The Particle Argon is a development board with WiFi capabilities, built using the Nordic Semiconductor nRF52840 system-on-chip [14]. This board is designed to be powered by LiPo batteries and for this reason contains a Torex XC9258A DC-DC buck converter for regulating the voltage to a level of 3.3 V. As a consequence of this, the Argon will most probably behave more as a constant power load than a constant current. For this reason, when imitating the Particle Argon using the Ace battery scripting, constant power should be used.

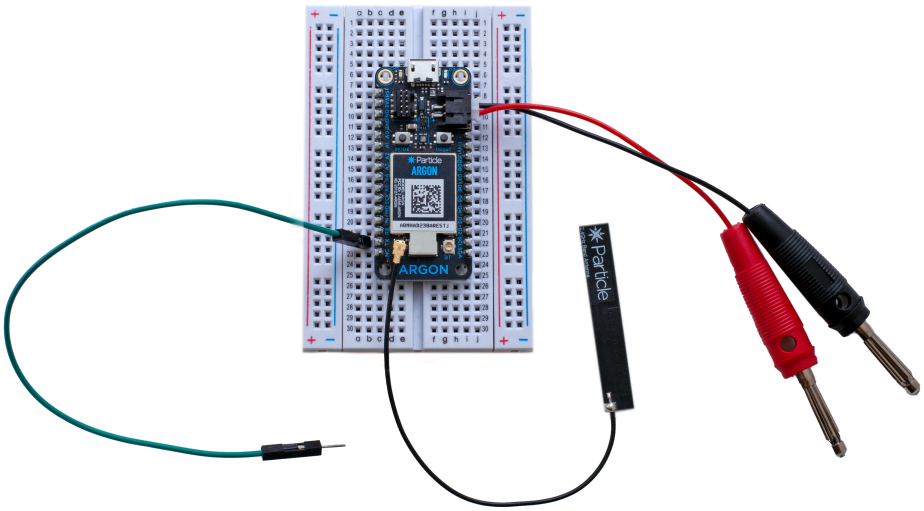


Figure 2.4: The development board Argon from Particle.

2.4 Related work

De Santis et al. [16] utilize a non-uniform piecewise constant approximation to accurately describe the arrival rate of patients at an emergency department. The incentive for the non-uniform segmentation is that the arrival rate is highly varying in the night, and more stable in the daytime, which requires a more precise estimate in the night. The best segmentation is found by solving a black-box optimization problem. The arrival rate is assumed to follow a Poisson process, therefore the optimization is constrained by a statistical test to verify this assumption. The problem is related to our research questions in terms of needing non-uniform segmentation to have higher accuracy in segments holding some important information. However, there lies a big difference in the nature of the data. Their problem is statistical in nature, where the patient arrival rates follow a Poisson distribution, i.e. the probability that a certain number of patients arrive within a time frame and that the arrivals are independent of one another. For our study on power consumption, the model will be based on the physical properties of a power consumption load on a battery.

Bergerhoff et al. [2] formulate a similar problem on non-uniform sampling of a signal to a fixed number of segments. In their description of the problem, segments are created using the mean values of the signal over set intervals, which are then optimized with regards to minimizing the global MSE. The optimization is done using a particle swarm optimization (PSO) technique which is a metaheuristic that iteratively tries to find the global minimum of the non-convex MSE function. Their approach is very interesting since they show that for the global MSE to be minimized, some parts may have a greater error than others, and vice versa. That being said, since our goal is to minimize some error after executing the series of segments on a battery, the exact MSE between our segmentation and the original data is not of much interest. This in conjunction with the fact that our approach is much based on exploring significant physical properties has led us to not use the described method.

2.5 Contribution

The contributions of this thesis include the proposed algorithm High peak resolution described in Section 4.5.2 for producing a non-uniform segmentation. Additionally, we provide insights into how said algorithm can be evaluated compared to some benchmarks, with regards to several aspects found to be of importance for the intended application. Furthermore, the thesis offers perspectives on how different characteristics of a power consumption curve might affect the behavior of a battery under the load of said power consumption.

Chapter 3
Method

In this section we will present a brief overview of the method, which will then be explained in a deeper context in Section 4.

The goal of our study is to find an algorithm that approximates this power consumption on the left such that the approximation can be used as load on the battery instead. Essentially we wish to replace the device, while retaining the behavior of the battery. The device we try to imitate in this study is the Particle Argon that is powered by a battery. Due to the hardware components of the Argon, we can anticipate that its power consumption will not change throughout the discharge. Our method of conducting this study consists of a few steps: First, we connect our Particle Argon to a fully charged battery. We have used a small rechargable Lithium Polymer battery. The active and sleep cycle is repeated until the battery is discharged, reaching a voltage of 3.17 V. During this process, we record Argon's power consumption and the voltage over the battery using otti ace. All recordings are made with a samplerate of 1000 sps From this recording, we select 5 cycles in the middle of the discharge where the battery is the most stable. These will act as input to the algorithm, one cycle at at time. This means that each cycle can utilize 200 steps from the total of 1000. The output from the algorithm is a step function. The step functions from all 5 cycles are combined into one. Next, we start over with a fully charged battery connected to the Otii Ace. The combined step function is used to discharge the battery until the same cutoff voltage level of 3.17 V. The same battery has been used throughtout all of our consecutive test runs.

Chapter 4
Implementation

4.1 Resources

4.1.1 Battery

For the study, a Lithium-ion Polymer 190 mAh battery is used [13]. The nominal voltage is 3.7 V. It is charged using a CCCV (Constant Current Constant Voltage) method, common for Lithium-ion batteries [9]. The charging process begins with constant current charging at a current of one times the capacity of the battery, i.e. 1 C or 190 mA. The voltage level will increase until the voltage of the battery is at the charging voltage, namely 4.2 V. At this voltage level, the charging will change to constant voltage. In this phase the current will decrease, carried out until the current reaches 0.1 C or 19 mA. The charge process is followed by an hour of resting the battery, allowing it to stabilize. Subsequently, the battery is considered fully charged. Given that the battery used in our study has a max discharge rate of 2 C or 380 mA [13] it is probable that it is fairly peak sensitive which is why we have focused on a method that emphasizes and tries to recreate peaks as much as possible.

4.1.2 Particle Argon

The DUT used in this study is the Particle Argon. The Argon is flashed with a firmware that cycles the following steps:

- Wake up.
- Turn on the WiFi module.
- Attempt to connect to a WiFi network with faulty credentials.
- Interrupt attempt after 9 seconds.

- Turn off the WiFi module.
- Enter a low-power sleep mode for 10 seconds.

Each of these steps produces a log message using UART communication to sync the activity with the voltage and power consumption curves.

4.2 Profiling setup

Figure 4.1 visualizes the setup of the profiling of the battery and the DUT. The DUT is connected to the Ace which in this setup works as a multimeter. Ace is measuring the power on the circuit and the voltage over the battery using 4-wire as described in Section 2.2.3. All measurements are downsampled to 1000 sps. The DUT is powered by the battery. UART log data is transferred from the DUT and received by Ace on the RX expansion port with millisecond accuracy. With the battery fully charged a synced recording of log, voltage and power data is made until the battery voltage drops below 3.17 V which is past the voltage cutoff of the DUT where the voltage is no longer sufficient to power the device.

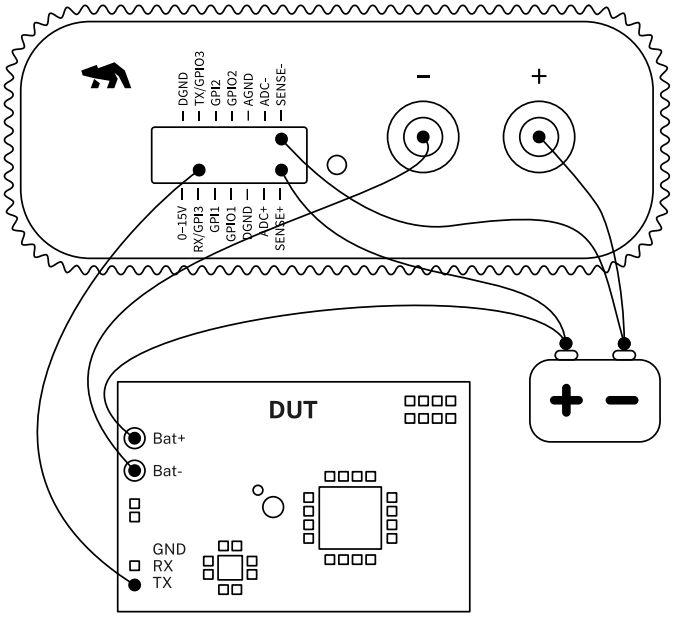


Figure 4.1: Profiling setup with Otii Ace and DUT connected to a battery.

4.3 Segmenting the consumption profile

Once the profile is recorded, the cycles are identified using the UART log messages produced by the DUT. One cycle is considered as the interval between the first log message of the cycle (inclusive) and the first log message of the next cycle (exclusive). A set number of five consecutive cycles are cycled to load the battery from fully charged to a cutoff of 3.17 V. The

algorithm is executed on one cycle at a time, meaning each cycle is to be segmented into a maximum of 200 steps, considering the maximum step size of 1000 . The five periods in the consumption curve are selected based on used energy, where around half of the energy is consumed. For example, if the total energy consumption for the DUT is 600 mWh, the cycle closest to where the energy consumption has reached 300 mWh is selected as the first cycle. This is where the battery used is in the vicinity of its nominal voltage of 3.7 V.

The algorithm outputs data in the form of Figure 5.1 which is then translated to a script that can be interpreted by the Otii Ace, with script steps as described in Figure 4.2b. In addition to the figure representing the script steps, the script has a global exit condition of 3.17 V cutoff voltage.

```
Timestamp,Value
225.323,0.07022
225.448,0.07670
225.573,0.09894
```

(a) CSV-format as input and output of the algorithm.

```
Step 1:
Constant current: 70.2mA
Exit condition: Duration 125ms

Step 2:
Constant current: 76.7mA
Exit condition: Duration 125ms

Step 3:
Constant current: 98.9mA
Exit condition: Duration 125ms
```

(b) Corresponding battery script.

Figure 4.2: Format of input and output of algorithm and how the output translates to a battery script, using the difference between two timestamps as the duration.

4.4 Peak detection

In the conduct of this study peak detection is used. A peak is a point in a signal which is a local maxima, i.e. its value is higher than its two neighboring values. This difference in values can be used as a threshold to find the peaks. Another way to describe peaks is by their prominence. The prominence is found by looking at valleys surrounding a peak. Figure 4.3 is a visual representation of how to calculate the prominence. From the peak of which the prominence shall be calculated, a line is drawn in both directions until encountering the slope of a higher peak, in the figure referred to as peaks A and B respectively. The areas below this line are referred to as the left and right valleys of the peak. By finding the lowest point of the left valley (point c) and the right valley (point d), the left and right bases of the peak are found, of which the higher one corresponds to the lowest contour line of the peak. The distance from the lowest contour line to the peak is the prominence.

We have used the peak detection function `find_peaks` from the signal package of the Scipy Python library which accepts a threshold and prominence as its parameters [17]. The algorithm produces a sorted list in ascending order of indices into the sampled data to where

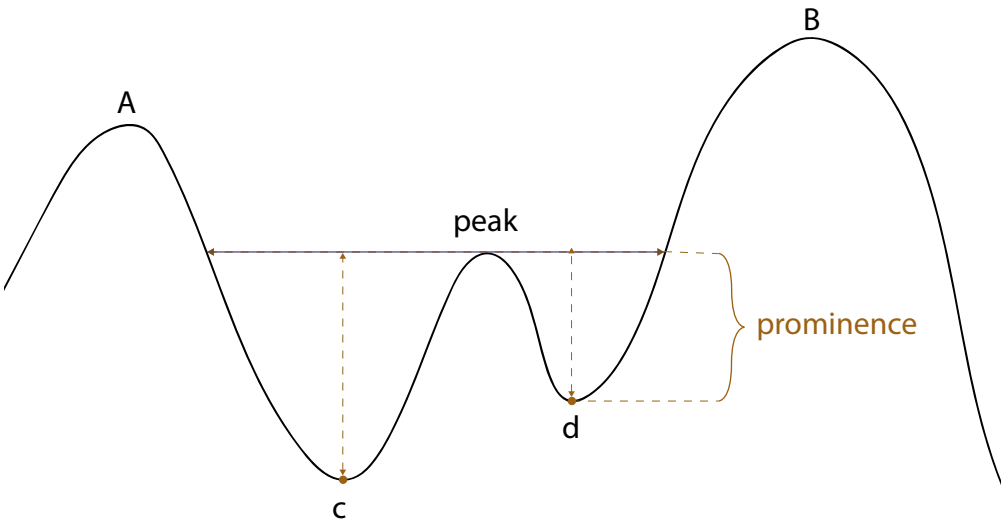


Figure 4.3: The prominence of a peak.

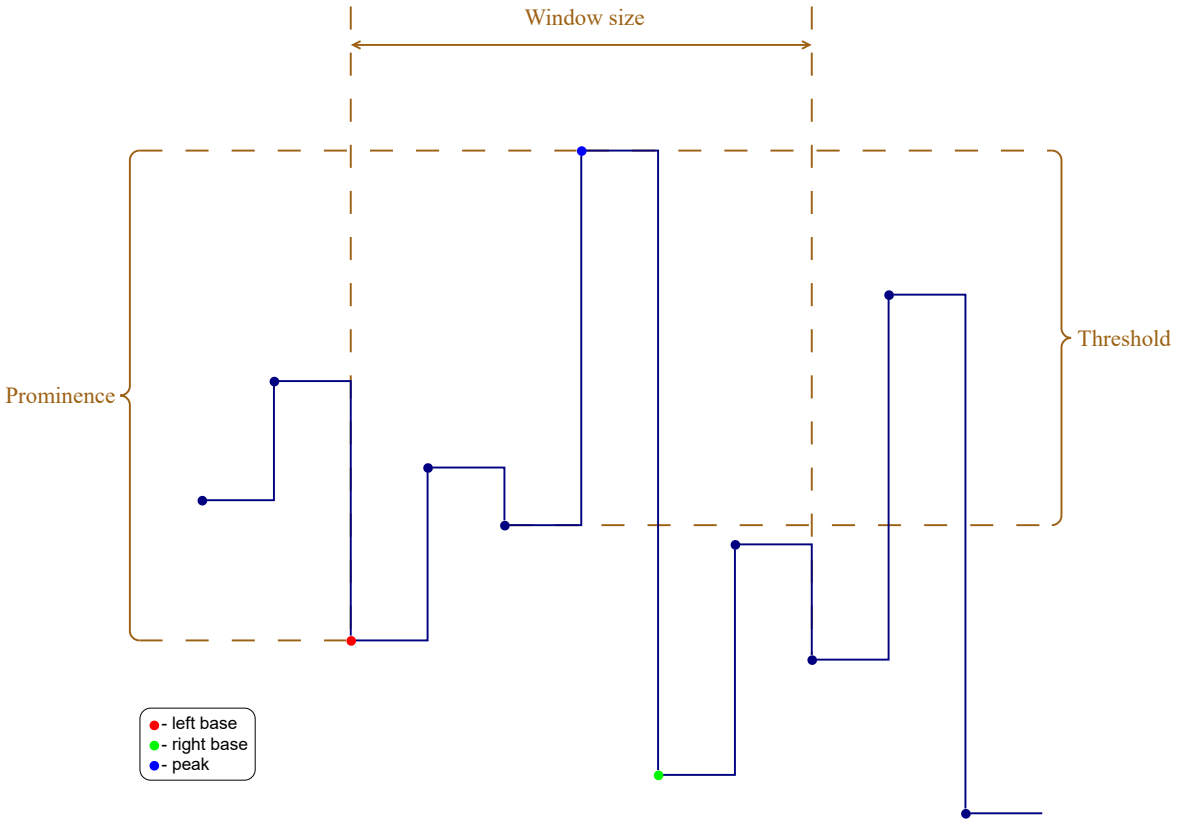


Figure 4.4: Example of a peak with its left base (inclusive) and right base (exclusive), showing the difference between prominence with a window size and threshold.

the peaks are located. The peaks are filtered using the minimum threshold and prominence if given as a parameter. When having a minimum prominence one could also filter the peaks further by setting a window size which will limit the search for a prominence within a certain

window of the samples. Figure 4.4 displays the relation between a peak, its threshold and its prominence according to some window size in samples. The function produces additional data depending on which parameters are given. If the algorithm is given a threshold, the thresholds to the left and right of the peaks are given, and if prominence is used, both the prominence and the indices of the left and right bases are calculated. The left base is inclusive and the right base sample is exclusive, which means that two adjacent peaks with the same right and left bases do not overlap.

The calculations are performed by first finding all the local maxima in the sample data. According to the SciPy documentation [17], a local maximum is defined as *any sample whose two direct neighbors have a smaller amplitude.* When the local maxima have been identified, they are filtered according to the parameters given to the algorithm. Prominence is calculated by the `peak_prominence` function when calling `find_peaks` using the following method [18]:

1. Find the size of the left and right valleys according to

$$\min(\lfloor window\_size \div 2 \rfloor, d)$$

   where d is the distance to the previous and next peak, respectively.

2. Find the lowest points within the valleys to the left and right from the peak; these are the bases.

3. The lowest contour line is the base at which the signal has the higher value.

4.5 Algorithms

The algorithms share a fundamental characteristic, the integral of the input power consumption data is the same as that of the output. This consistency is of importance to ensure that energy consumption remains unaltered across the five periods, since $E = P \cdot t$, i.e. power over time translates to energy. The input data is in CSV format where each row has columns "Timestamp" and "Value". Each row, in combination with the subsequent row, is the representation of a sample. The sample value starts at the timestamp of the same row and holds until the timestamp of the next row. The output data is in the same CSV format, and is to be translated into a battery script according to Figure 4.2. Here, each row is the start of a battery script step, and the duration is the time between the current row and the next. This time is translated to a duration exit condition of the step. Used as a baseline for the exploration of algorithms, one of the simplest algorithms is downsampling the data into even chunks. This has been the reference throughout the study to evaluate whether a new algorithm performs better than this baseline.

4.5.1 Downsampling

Simple average downsampling of the cycle is performed at different frequencies, i.e. taking the average of a set number of neighboring samples. Firstly, each cycle is downsampled into two steps aiming to represent the active period and the sleep period of the DUT. Ergo, each cycle is represented by one discharge step with high power, and one with low power, rendering a total of ten steps for the five cycles. Secondly, a minimum downsampling based on

the limitations of the Ace is performed, meaning each cycle is to be downsampled to 8 sps. For the Argon used in this study, with a cycle length of roughly 22 seconds, this means that each cycle is represented with approximately 180 steps, and each set of five cycles will use 900 steps, which is close to the limit of 1000 for battery scripting with the Ace.

4.5.2 High peak resolution

Given that peaks in the power consumption of the Argon Particle can appear within the duration of a single sample, equivalent to 1 ms, downsampling will attenuate these peaks, since each step in the minimum downsampling method will be 125 ms. To replicate the characteristics of a peak more accurately we need to make a non-uniform segmentation.

The key idea of this algorithm is to increase the resolution around the peaks, in comparison to the downsampling algorithm, to better represent them. Peaks are detected as described in Section 4.4 with a prominence of three times the standard deviation of the power with a window size of 40 samples. The peaks are characterized using the left and right base of the peak, with the peak somewhere in between. The algorithm is described with pseudo code in Algorithm 1. The principle is that all samples between the peaks will be represented using only one step with a value that is the average of all samples. In comparison, the peaks will be represented using the highest resolution possible considering the step limit of the Ace battery scripting.

Before starting the algorithm, the peaks are combined using the `combine_peaks` procedure, so that overlapping peaks are handled as one set of samples, and downsampled together. There are two cases of overlaps, either the right base of a peak is overlapping with a left base of the next peak, or the peak can be encapsulated by a bigger peak, as depicted in Figure 4.5. Henceforth, the number of steps used to represent the area between the peaks will be one less than the number of peaks, as in line 3, leaving the rest of the 200 steps available to try to replicate the peaks (line 5). Using this max amount of steps, the maximum resolution, or minimum downsampling sample rate of the peaks can be calculated (line 7). Hence there exists a trade-off between the amount of peaks and the resolution of the peaks, which is a natural cause of the step limit.

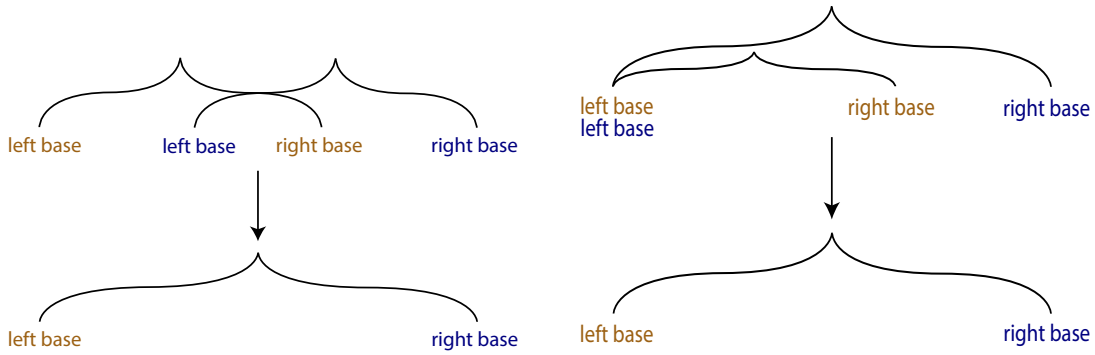
The next step is to iterate over the peaks and create the non-uniform segmentation. In the first iteration, the values from the start of the original samples (`prev_time`, line 8) until the left base of the first peak (exclusive) are grouped, and a mean is calculated. This value is inserted at the start time. Next, we process the samples representing the peak, namely with timestamps bigger than or equal to the left base and smaller than the right base. The samples are downsampled to the target peak sample rate and added to the result. `prev_time` is set to the right base of the peak, which is the timestamp of the first sample after the peak. Subsequently, this is repeated for all peaks, where `before_peak` is the samples from the right base of the previous peak, until the left base of the current peak. Once all peaks are iterated, the samples after the last peak shall be handled in the same way as the samples in between peaks (lines 16-17).

4.5.3 One peak

In order to see if the maximum peak height of a cycle is the only thing of importance we do one additional experimental algorithm. It will identify the max value of one active-sleep

Algorithm 1 High peak resolution.

```
1:  procedure HIGH_PEAK_RESOLUTION(samples, peaks)
2:      peaks ← combine_peaks(peaks)
3:      steps_between_peaks ← len(peaks) + 1
4:      total_peak_length ← ∑_{l,r∈peaks}(r − l)
5:      steps_for_peaks ← 200 − steps_between_peaks
6:      sample_rates ← {f | f ∈ Z, 1 ≤ f ≤ 1000, 1000 mod f = 0}
7:      peak_sample_rate ← max(sr | sr ∈ sample_rates, sr · total_peak_length ≤
    steps_for_peaks)
8:      prev_time ← min(t | t, _v ∈ downsampled)
9:      result ← ∅
10:     for left, right ∈ peaks do
11:         before_peak ← {t, v | t, v ∈ samples, prev_time ≤ t < left}
12:         result[prev_time] ← mean(before_peak_values)
13:         during_peak ← {t, v | t, v ∈ samples, left ≤ t < right}
14:         result append samples from downsample(during_peak, peak_sample_rate)
15:         prev_time ← right
16:     after_last_peak ← {t, v | t, v ∈ samples, prev_time ≤ t}
17:     result[prev_time] ← mean(after_last_peak_values)
18:     return result
19:
20: procedure COMBINE_PEAKS(samples, peaks)
21:     result ← [peaks[0]]
22:     for left, right ∈ peaks do
23:         prev_left, prev_right ← result[−1]
24:         overlap ← left ≤ right
25:         if overlap then
26:             result[−1] = (prev_left, max(right, prev_right))
27:         else
28:             result append (left, right)
29:     return result
```



(a) The left peak and the right peak have overlapping bases. The left peak is replaced by the new representation.

(b) The left peak is encapsulated by the right peak. The right base of the left peak is replaced by the right base of the right peak.

Figure 4.5: The peaks, represented by their left and right base, are combined if they are overlapping in one of the two ways.

cycle, and hold this value for 10 ms, ideally creating a semi-long peak. The rest of the period holds a lower constant power value. Consider the following example: the cycle has the highest documented value at 1.5 W, and the integral of the original data is 4 Ws, with a cycle length of 20 seconds. The cycle will then be replaced by two steps.

- 1.5 W for 10 ms.
- ~200 mWs for 19.9 seconds.

The length for the second step is calculated by $l = T - 10$ ms where T is the cycle length in seconds. The value is calculated by $\frac{A_{total} - A_{peak}}{l}$ where A_{peak} is the peak value times the peak length, i.e $1.5\ \text{W} \cdot 10\ \text{ms} = 15$ mWs and A_{total} is the integral of the original data, namely 4 Ws.

Chapter 5
Results

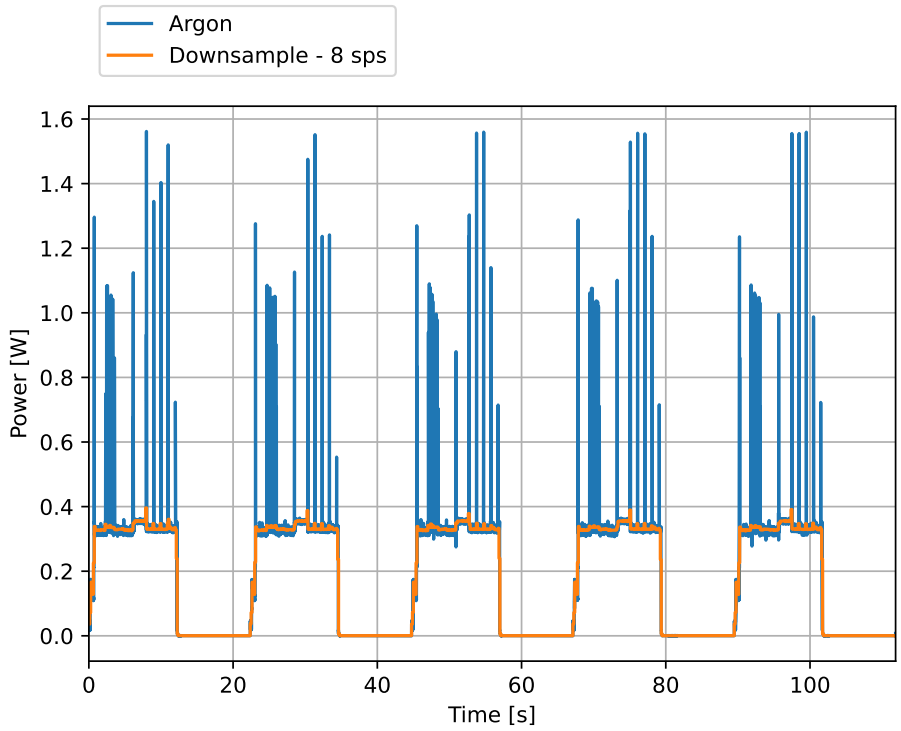
5.1 Algorithm output

Figure 5.1a-5.1d offer visual representations of the algorithm outputs in orange in comparison to the input in blue. It is a depiction of how specific characteristics are retained, or overlooked, by the algorithm. The high peak resolution algorithm aims towards recreating peaks more accurately compared to simple average downsampling which attenuates peaks. The difference can be analysed by comparing Figures 5.1b and 5.1a. The peaks are heavily attenuated after downsampling, while the high peak resolution algorithm does a much better job of recreating the peaks. However, it still attenuates the peaks somewhat. This issue arises from having an insufficient amount of steps to recreate the peaks at a maximum resolution with regard to the maximum step limitation, as well as the number of peaks detected by the peak detection algorithm. To achieve a higher resolution of the peaks with the current hardware limitations, the filtering of peaks needs to be more meticulous by imposing stricter prominence requirements, since there is a trade-off in number of peaks and the resolution of them, as discussed in Section 4.5.2.

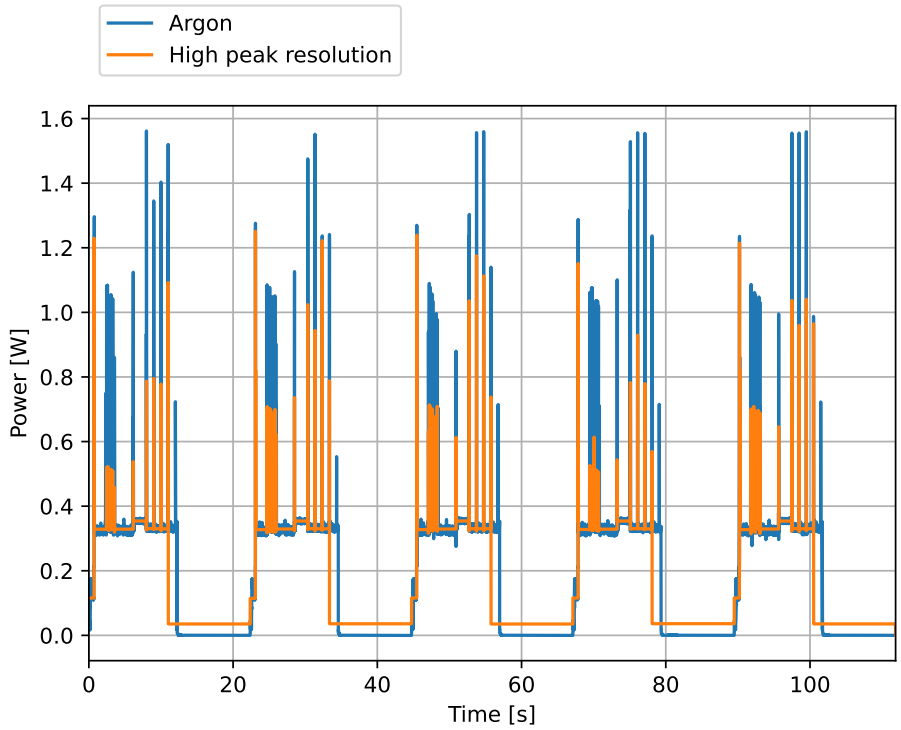
5.2 Particle Argon

Results regarding the profiling conducted on Particle Argon are presented in this section. Figure 5.2 illustrates the power consumption over time. The energy consumed by each active-sleep cycle is shown in Figure 5.3. It illuminates how the energy consumption changes as the connected battery is discharged. Variations in energy consumption per cycle may arise from different power consumption or cycle lengths. The variation in cycle lengths is explored in Figure 5.4. The mean cycle length is 22.366 s with a standard deviation of approximately 0.023 s.

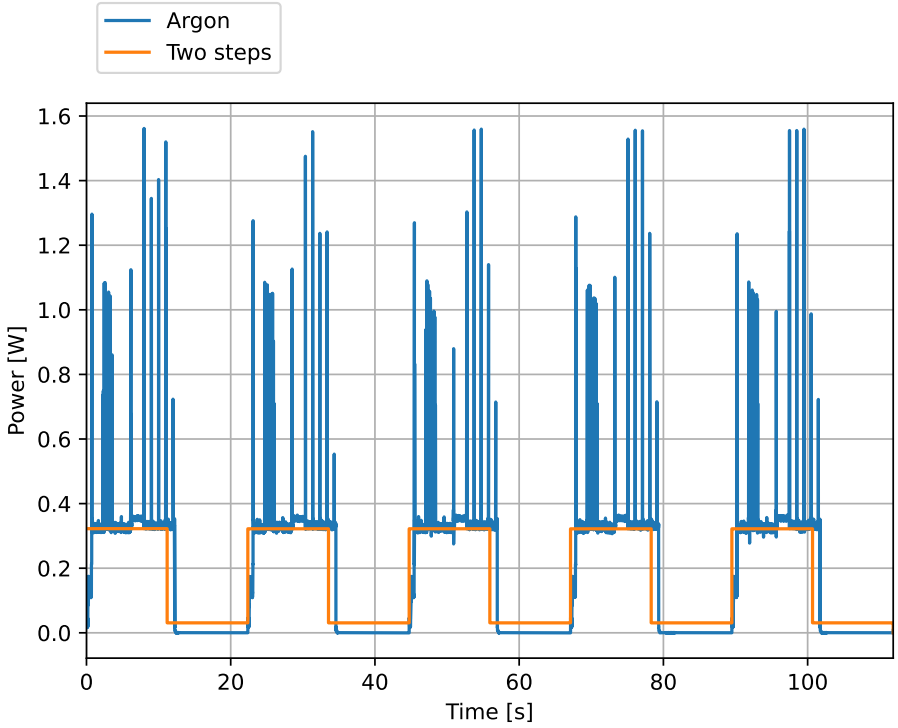
As mentioned in Section 2.3 the expected behavior when using a Particle Argon powered



(a) Downsample - 8 sps: attenuating the peaks.



(b) High peak resolution: preserving some peak characteristics.

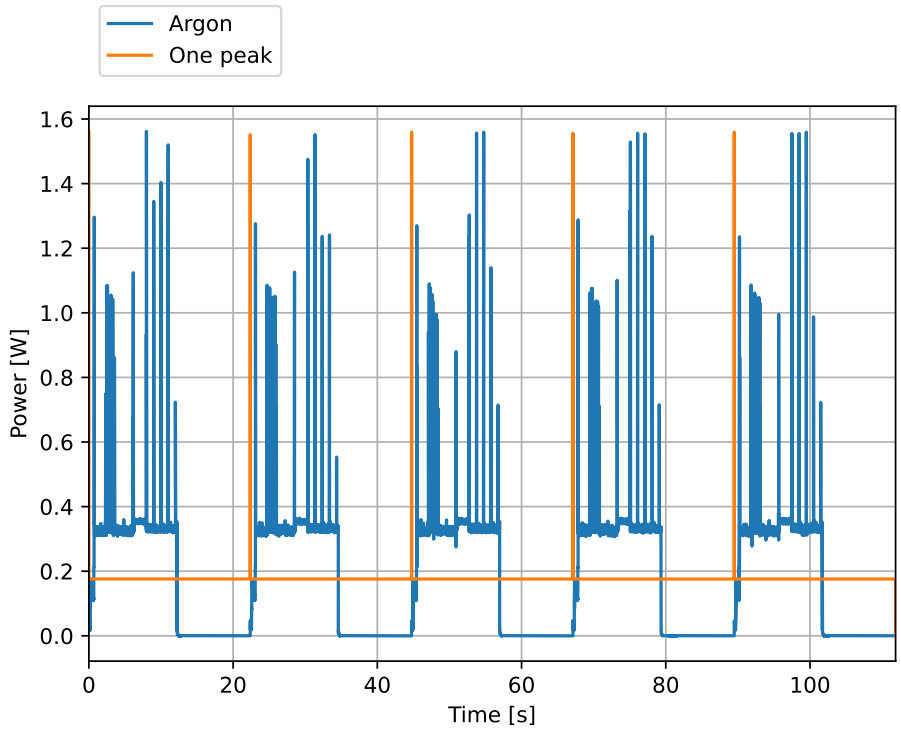


(c) Two steps: high power followed by low power.

by a battery is that it consumes a constant amount of power. Figure 5.2 can affirm that this is an accurate assumption for the beginning of the discharge. At the later phase of the discharge we distinguish a drop in power consumption, which may cause some errors at the end of the discharge due to the constant power assumption. This is likely caused by the big drop in voltage in the end phase of discharging a Li-ion battery. To take this into account when evaluating the algorithms, we have split our results into two parts considering voltage: how the algorithm performs until the end of the discharge cycle, at 3.17 V, and before the big voltage drop, which we set at 3.55 V. Before this drop, we can also observe a slight decrease in energy consumption per cycle at the beginning of Figure 5.3, before plateauing near where half the energy has been consumed. Because we pick our period of five cycles near this plateau, the slightly higher energy consumption per cycle of the Argon before this can affect the results and will be taken further into consideration when evaluating the algorithms.

5.3 Voltage level

Two different metrics are used for measuring how the segmentation impacts battery voltage level; measured voltage over time and change in average voltage over time. Firstly, Figure 5.5 is a plot of samples from the measurement of voltage for the period of five cycles, along with the mean squared error (MSE) of all data points compared to the Argon, rounded to 6 decimal points. Secondly, Figure 5.6 is an illustration of the change in voltage from one period of five cycles to the next one. The data points of the plot are the average voltage over one such period, connected by linear interpolation. In addition to these plots, Figure 5.7



(d) One peak: imitate the highest value for one peak only.

Figure 5.1: Input and output from algorithms in blue and orange respectively.

shows a min-max plot of all data points of the voltage during a full discharge for the Argon. The distance between the minimum and maximum increases as the voltage gets lower.

The voltage curve shown in Figure 5.6 displays information regarding how the different algorithms perform in matching the reduction of voltage as the battery is discharged. There is no significant difference across the algorithms, rendering peak preservation superfluous in terms of matching the voltage profile when discharging the battery in question. The small differences that can be observed in the curve could depend on various factors. The most relevant one is the initial state of charge of the battery, seen as a difference in voltage level in the beginning, which can be compensated for by looking at the slope. The slight discrepancy near the end is most likely caused by the fact that the power consumed by the Argon decreases towards the end, resulting in a more shallow decrease in voltage.

5.4 Energy consumption

Energy consumption can be examined from different perspectives, which will be depicted in this section. Table 5.1 presents energy consumption across different algorithms at various points in time. Conversely, Table 5.2 gives insights into the consumed energy at certain voltage cutoffs. That is, when the minimum voltage measured reaches a set value, the energy consumption up until this point in time is noted.

Energy consumption over time aligns closely between all algorithms, as seen in Table 5.1,

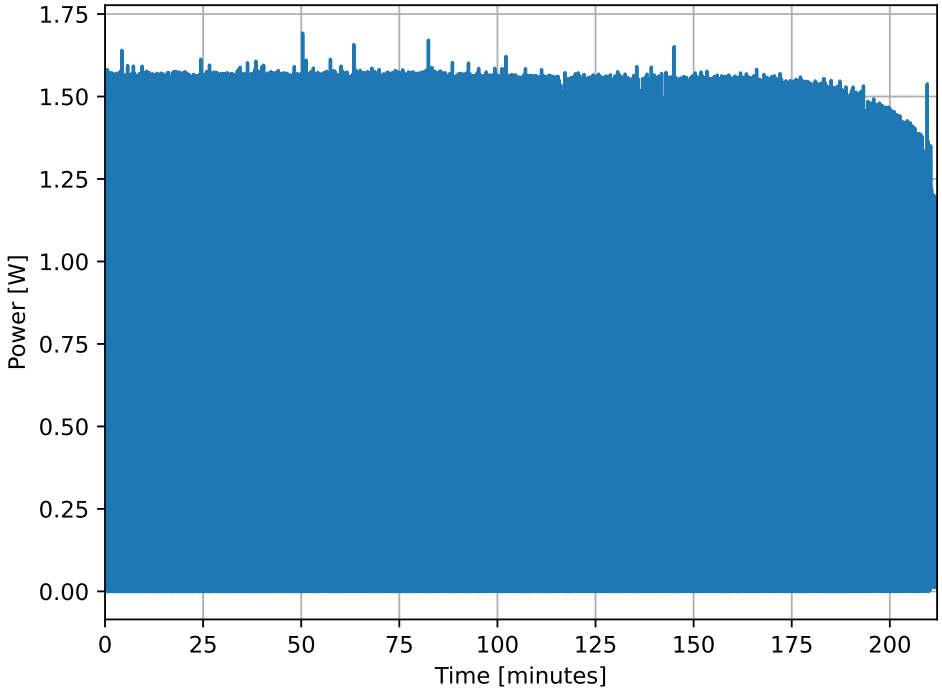


Figure 5.2: Power consumption over time for particle Argon. It appears dense due to big variations in power rapidly in each cycle, over an extended period of time

which is expected because of the conservation of energy characteristic of the studied algorithms. There is however a small difference between the algorithms and the Argon. This difference is increasing in the beginning from around 0.40 mWh after 10 minutes to about 1.5-2.0 mWh after 100 minutes. Given that the behavior is the same for all algorithms, it is probable that this is due to what we saw with the energy per cycle of the Argon decreasing a little before reaching the plateau. After this, the difference almost stays constant and even slightly reducing. Over time this difference is negligible, going from 12.2-18.9 ‰ after 10 minutes to 2.7-3.3 ‰ at the end. The paramount observation is that there is little to no difference in energy consumption over time regardless of algorithm choice. The only difference of significance is that between algorithms and the Argon, caused by making the assumption that Argon consumes constant power which has been proven to be an approximation.

Using Table 5.2 we analyze the total energy consumption at the voltage cutoffs. At the 3.55 V cutoff, the order in which the algorithms drop below the cutoff corresponds to how well peaks are recreated, with algorithms preserving higher peaks better being closer to both the cutoff time and total energy. For 3.17 V the results do not match what we would expect with the algorithms reconstructing peaks with there being no apparent order. Because the power load of the Argon deviates significantly from constant power as discussed previously, perhaps this result should be disregarded.

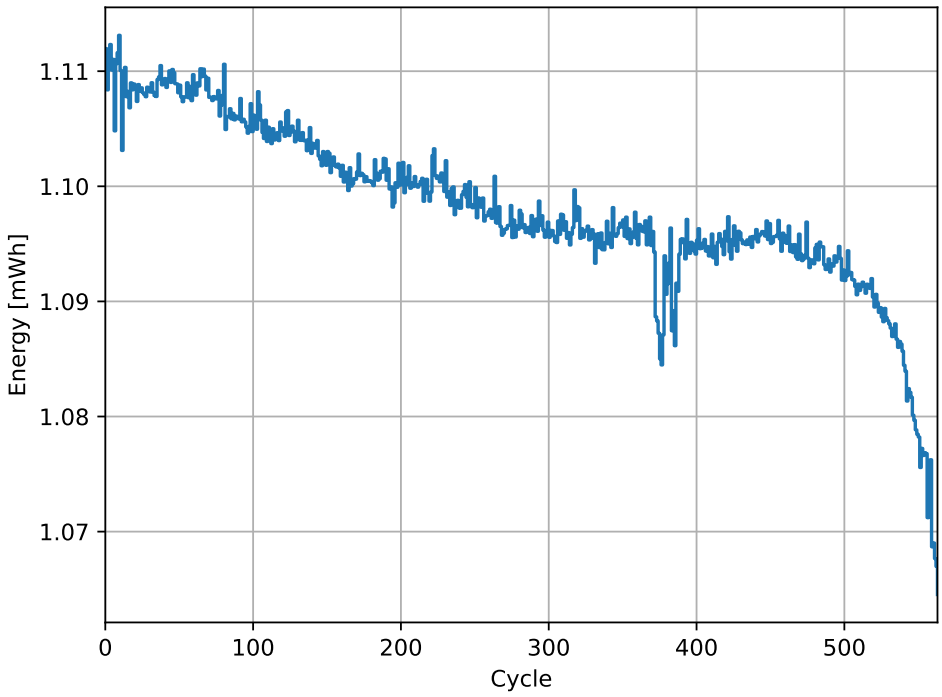


Figure 5.3: Energy consumption per cycle of Particle Argon. It is somewhat constant in the middle of the discharge. It has some variations in the start, and major variations in the end.

Table 5.1: Energy consumption over time. The algorithms are very similar and differ slightly from Argon, less than 1% after 50 minutes.

	10 min	**50 min**
Argon	29.97 mWh	148.50 mWh
High peak resolution	29.57 mWh (−0.40 mWh/−13.5‰)	147.09 mWh (−1.41 mWh/−9.5‰)
Downsample - 8 sps	29.61 mWh (−0.37 mWh/−12.2‰)	147.08 mWh (−1.42 mWh/−9.6‰)
Two steps	29.57 mWh (−0.40 mWh/−13.4‰)	147.12 mWh (−1.38 mWh/−9.3‰)
One peak	29.41 mWh (−0.57 mWh/−18.9‰)	147.04 mWh (−1.45 mWh/−9.8‰)
	100 min	**150 min**
Argon	296.24 mWh	443.25 mWh
High peak resolution	294.23 mWh (−2.01 mWh/−6.8‰)	441.36 mWh (−1.88 mWh/−4.3‰)
Downsample - 8 sps	294.20 mWh (−2.03 mWh/−6.9‰)	441.32 mWh (−1.92 mWh/−4.3‰)
Two steps	294.23 mWh (−2.00 mWh/−6.8‰)	441.35 mWh (−1.90 mWh/−4.3‰)
One peak	294.09 mWh (−2.15 mWh/−7.3‰)	441.13 mWh (−2.12 mWh/−4.8‰)
	200 min	
Argon	590.09 mWh	
High peak resolution	588.51 mWh (−1.58 mWh/−2.7‰)	
Downsample - 8 sps	588.46 mWh (−1.63 mWh/−2.8‰)	
Two steps	588.47 mWh (−1.62 mWh/−2.7‰)	
One peak	588.17 mWh (−1.92 mWh/−3.3‰)	

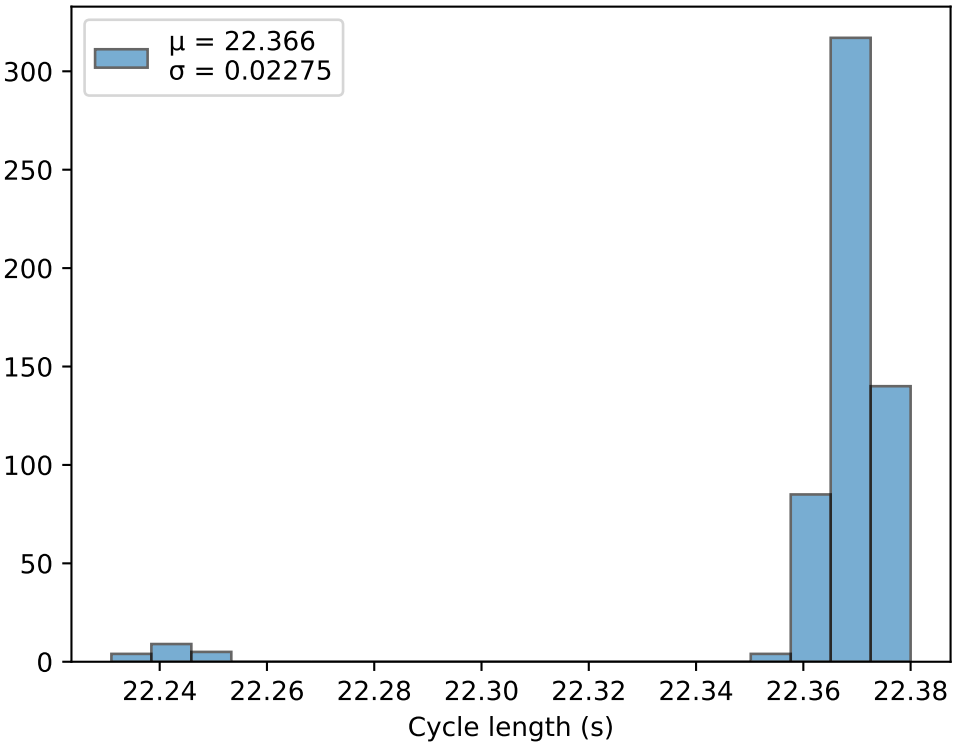


Figure 5.4: Distribution of the active-sleep cycle length for Particle Argon, with mean μ and standard deviation σ.

Table 5.2: Energy consumption and time duration until voltage cut-off. There is a bigger variation in algorithms at 3.55 V than 3.17 V. At 3.55 V, more accurate maximum peak amplitude recreation renders results closer to Argon.

	3.17 V
Argon	620.04 mWh - 210 min
High peak resolution	609.64 mWh (−1.7%) - 207 min (1.6%)
Downsample - 8 sps	610.4 mWh (−1.6%) - 207 min (1.4%)
Two steps	617.08 mWh (−0.5%) - 210 min (0.3%)
One peak	614.1 mWh (−1.0%) - 209 min (0.8%)
	3.55 V
Argon	501.4 mWh - 170 min
High peak resolution	538.36 mWh (7.4%) - 183 min (−7.7%)
Downsample - 8 sps	553.37 mWh (10.4%) - 188 min (−10.8%)
Two steps	559.14 mWh (11.5%) - 190 min (−11.9%)
One peak	479.22 mWh (−4.4%) - 163 min (4.0%)

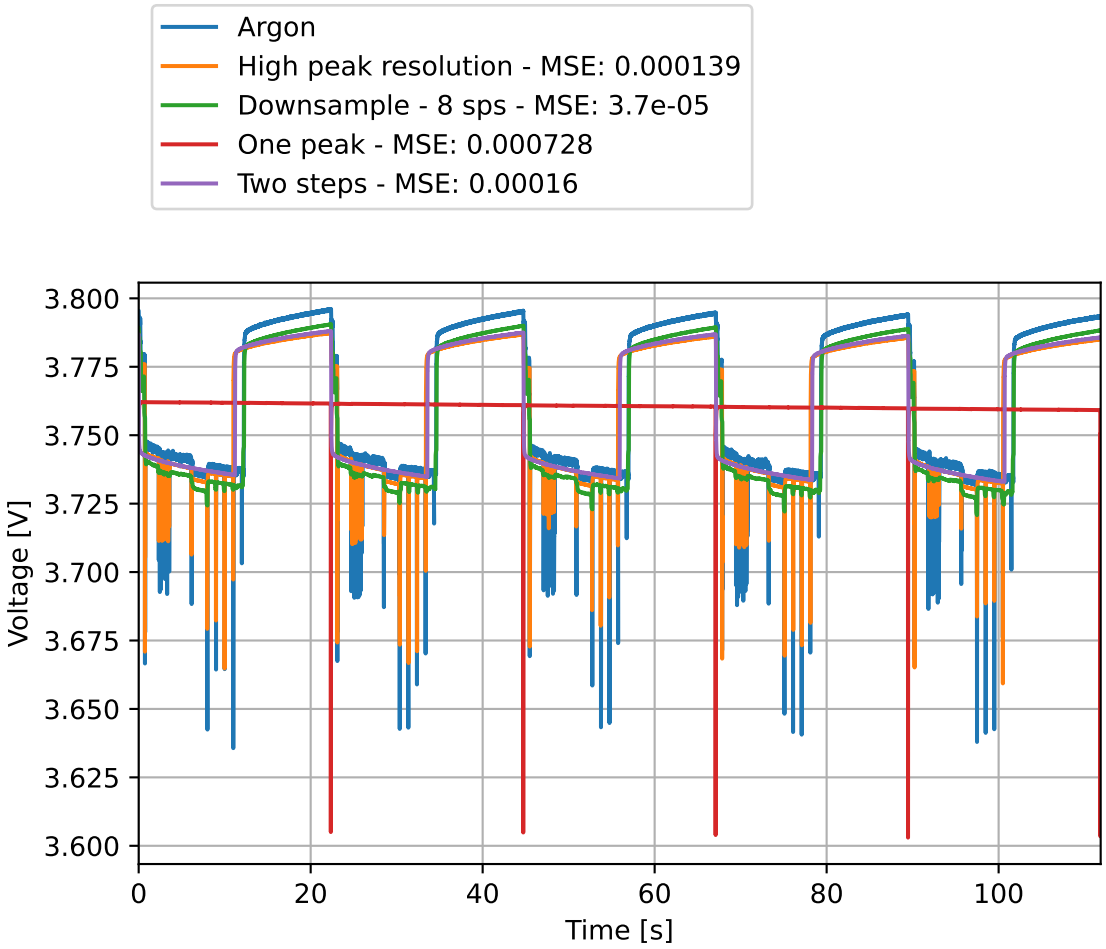


Figure 5.5: The battery voltage over the five periods, with MSE compared to Particle Argon. Downsample has the lowest MSE compared to the other algorithms.

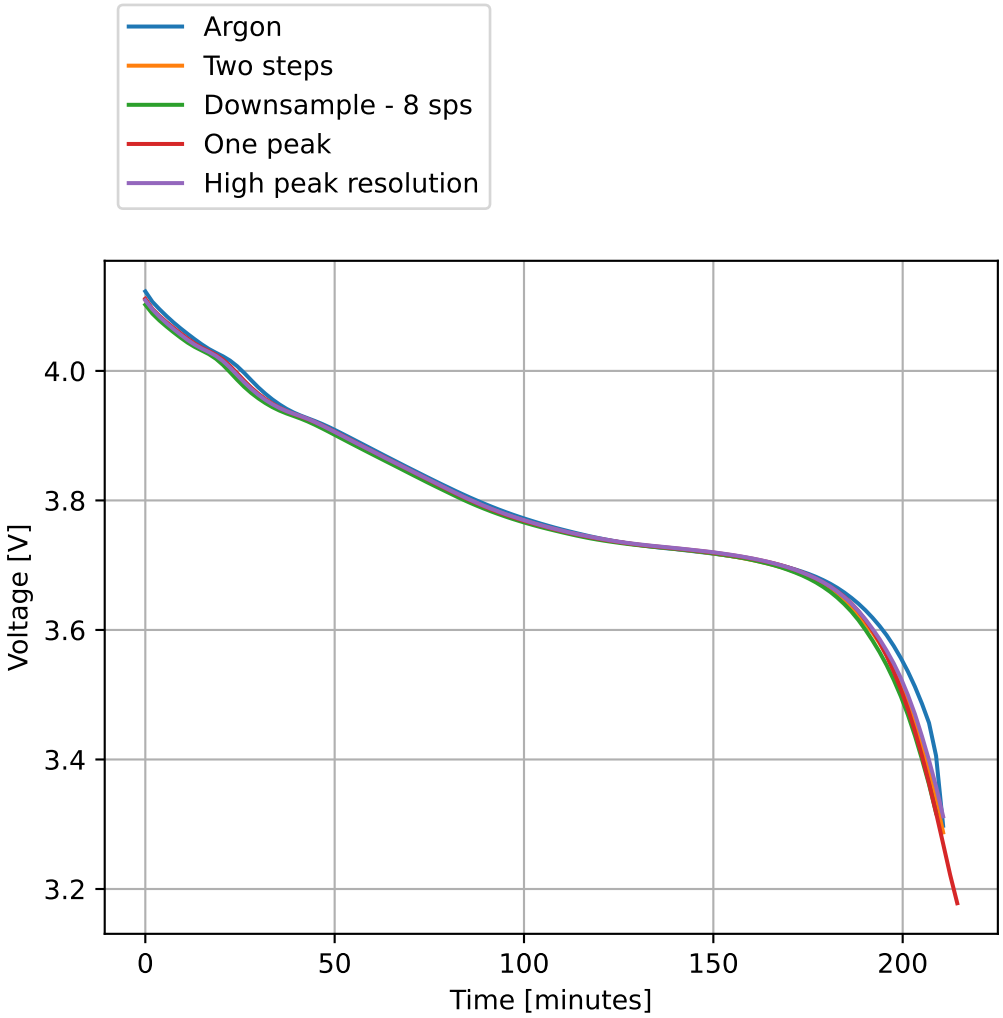


Figure 5.6: The mean voltage per cycle of the battery over time. All of the algorithms largely follow the expected voltage profile of the Particle Argon.

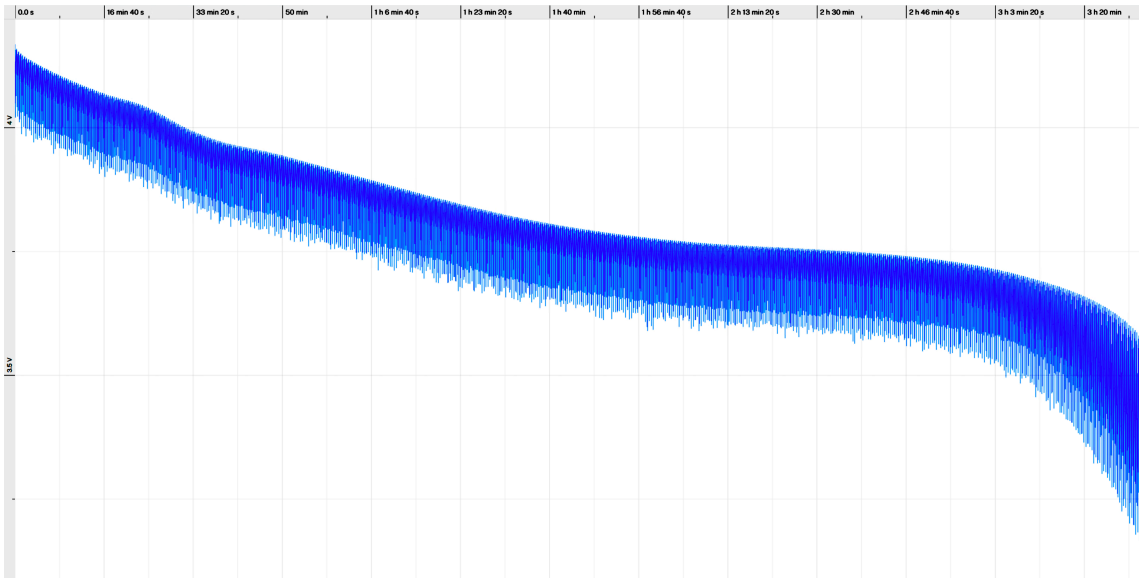


Figure 5.7: Voltage curve of the Particle Argon over a full discharge.

Chapter 6

Discussion

6.1 Evaluation

Figure 5.5 together with the power consumption of different algorithms in Figure 5.1 is a representation of how the voltage appears as a mirrored image of the power. We observe that higher peaks lead to a bigger voltage drop. In this case, the amplitude of a peak becomes important because when the voltage drops there is a possibility that it drops below the voltage cutoff. This leads to reaching the cutoff earlier than if the peak had a lower amplitude. Figure 5.7 makes this more apparent as the effect of peaks becomes more prominent as the battery voltage decreases, meaning that the battery becomes more peak sensitive as it is depleted. One peak is one of the algorithms preserving peaks and it will in most cases contain higher peaks than High peak resolution because the maximum amplitude is kept. This is the likely reason it performs better in terms of how close it gets to the Argon in consumed energy and battery life. It does however reach the cutoff voltage earlier than the Argon and has at that point been able to get less energy out of the battery. One peak is an extreme case that is rougher on the battery than the Argon is, due to the peak being of maximum amplitude for 10 ms which is longer than the corresponding maximum peaks of the Argon. Longer high peaks have a greater effect on the battery voltage than shorter ones as described in Section 2.1.1, because the voltage does not have enough time to drop all the way before the load is removed for shorter peaks. The fact that the voltage may not have time to drop as low for shorter peaks of the same amplitude means that it might not reach the cutoff voltage as early, which is likely the reason why Argon gets longer battery life with power consumptions with more attenuated peaks, as when using downsampling to 8 sps or two steps. High peak resolution is the second closest but is not as extreme a case as One peak in that the peaks are preserved but slightly attenuated. Consequently, it gets a longer battery life than the Argon and is able to consume more energy before hitting the voltage cutoff.

6.2 Comparison metrics

One of the research questions for this study is how the algorithms should be evaluated and what constitutes a satisfactory result. This is not self-evident. There are three important metrics when discussing batteries and their longevity, namely voltage level, consumed energy and the battery life for a specific application.

When evaluating the voltage, our primary thought was to look at the five periods that are cycled and compare the voltage curves as is done in Figure 5.5. However, several factors make this a difficult task. Firstly, the initial state of the battery can be different, leading to curves being offset on the voltage axis. Secondly, there is also an offset on the time axis. A reason for this is that the start time can vary slightly between different runs. Another reason is that the battery scripting has a rise time which causes the peaks to be shifted slightly depending on their amplitude, described with greater detail in Section 6.3.1. Given the foregoing, the actual voltage level at all points in time is not of that much interest to the target application. This renders Figure 5.5 and the calculation of MSE insignificant in a broader context. MSE is calculated by squaring the difference between each point on the curves, and then taking the mean of these squares. When comparing the curves point by point, a shift of 1 ms, i.e. one sample, is enough to cause the error to be calculated on the wrong samples. In addition, if the state of charge is not matched, this will introduce an MSE which is simply a representation of the difference in battery state. Furthermore, it is clear that, point by point, an algorithm such as one peak will deviate from the voltage of Argon, since the voltage curve appears as a mirrored image of the power.

Instead of looking at the voltage at every point in time, the paramount consideration is how the voltage is changing over some period of time, and how this might affect the battery. This is why we included Figure 5.6 which shows the mean voltage of five cycles over time and is as such easier to use for comparison. This graph still has the issue of being offset in the voltage axis due to a slight difference in initial levels of charge, but it is in fact the gradient rather than the absolute values that are of most interest.

Consumed energy can be measured both over time, and over the voltage until a certain voltage cutoff. The goal is to retain the amount of energy consumed when the battery reaches a set cutoff voltage, which means that the total energy at the cutoff is an important metric. In addition, the time it takes to reach said voltage cutoff is of importance in order to evaluate the battery life for the application.

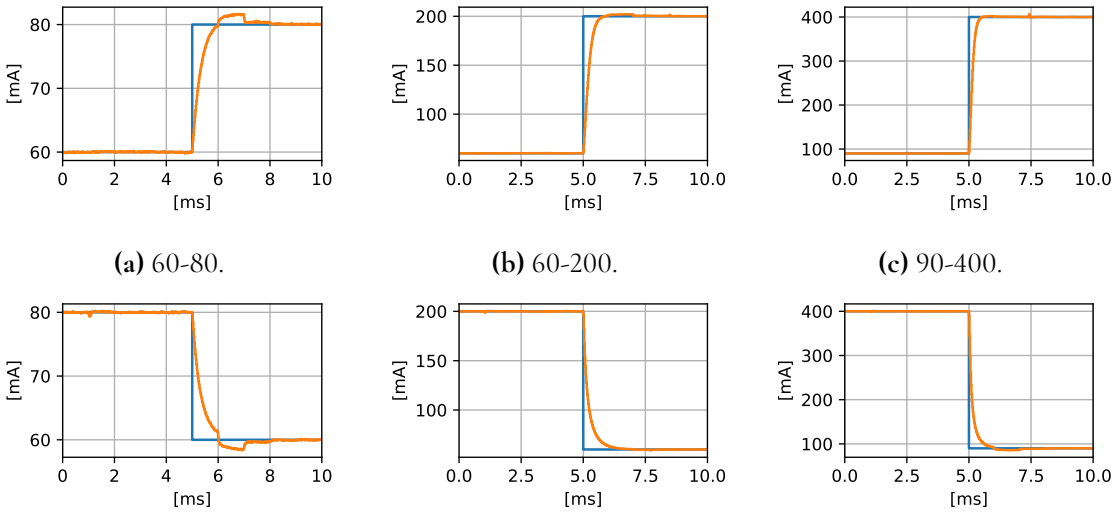
6.3 Hardware limitations and consequences

6.3.1 Battery script accuracy

The input to Ace while battery scripting is essentially a step function of current or power values. Naturally, this can not be reproduced with 100 percent accuracy using the hardware when regulating the value. To demonstrate this we observed the behavior of the Ace on three different changes in current, going from low to high and high to low. Each level was held for 5 ms. Figures 6.1a-6.1c represent increasing the sink current, i.e going from sinking a lower current to a higher one, while Figures 6.1d-6.1f represent the corresponding decrease in current sink, going from sinking a higher value to a lower. When regulating the current

and changing the set-point three primary metrics are considered: rise time, settling time and overshoot. The rise time quantifies the duration for the system to shift from the current state to a new set-point. This system has a rise time of around 1 ms. The settling time describes the time for the system to stabilize around the desired set-point. This is generally longer than the rise time for the current regulation, seeing there is presence of over- and undershoot. To investigate how well Ace battery scripting can represent the script, we integrate the curves in Figure 6.1 and compare them. The integral is relevant since this will translate to energy consumption, since energy is voltage multiplied by current over time. The results are listed in Table 6.1. We can see a slight difference in area, however, it is highly possible that these errors will cancel each other out since going from low to high gives a bigger area and high to low a very similar difference. However, there are some worst-case scenarios. If there is a short pulse of 1 millisecond, there is a risk that the value never has time to rise to the set point, which might unintentionally attenuate some short peaks with a high amplitude.

To investigate how this hardware limitation affects the result of this study, we compared the area of the expected values, i.e. the output from the algorithm, to the power measured by the Ace when executing the battery script. The outcomes are placed in Table 6.2. The difference in area can be estimated to an energy difference by multiplication with the average voltage. Using the nominal voltage of the battery, 3.7 V, the estimated energy is in the last column. This difference is less than 0.1‰and can be considered insignificant. As expected, the high peak resolution algorithm which tries to preserve peaks shows the highest error.



(a) 60-80. **(b)** 60-200. **(c)** 90-400.

(d) 80-60. **(e)** 200-60. **(f)** 400-90.

Figure 6.1: Trying to regulate the current using Ace battery scripting, with input in blue and output in orange. There is some over- and undershoot, as well as ~1 ms settling time.

6.3.2 Cycle variation

Because of the non-deterministic nature of wirelessly connected IoT devices [10], connecting the DUT to WiFi can make the cycle periods vary in length. This is due to setting up the WiFi

Table 6.1: Battery script accuracy for different set-point changes.

Sink current	Expected	Result	Diff
60 mA - 80 mA	0.70 mAs	0.696 mAs	0.006%
60 mA - 200 mA	1.30 mAs	1.272 mAs	0.021%
90 mA - 400 mA	2.45 mAs	2.408 mAs	0.017%
80 mA - 60 mA	0.70 mAs	0.705 mAs	-0.007%
200 mA - 60 mA	1.30 mAs	1.331 mAs	-0.024%
400 mA - 90 mA	2.45 mAs	2.490 mAs	-0.016%

Table 6.2: Area under the battery script expected output and actual measured power over time, with the difference in area and how this translates to energy at 3.7 V.

	Expected	Result	Area diff	Energy diff
High peak resolution	2191.02 Ws	2191.16 Ws	0.14 Ws	0.040 mWh
Downsample - 8 sps	2194.97 Ws	2194.92 Ws	-0.05 Ws	-0.014 mWh
Two steps	2218.66 Ws	2218.59 Ws	-0.07 Ws	-0.019 mWh
One peak	2210.76 Ws	2210.76 Ws	-0.00 Ws	-0.000 mWh

connection to an access point which sometimes needs more time. Our initial experiments used a DUT with a firmware which was supposed to connect to a local WiFi network and stay connected for a few seconds before disconnecting and going to sleep. There were many issues with this firmware because of some as-yet-unknown incompatibility with the WiFi access point which led to the device not getting connected and timing out at some points during the recording. This resulted in a great variation in the cycle lengths and also the energy consumed during those affected cycles.

For the above-mentioned reasons, and because we want as little variation as possible between cycles, we have instead used a different custom firmware where the credentials for the WiFi are wrong. This means that the device will try to connect to the WiFi network, making the power consumption profile look realistic, but fail due to the password being wrong and eventually time out. The timeout is set to a fairly short time of 9 seconds which makes the cycle lengths more consistent while still being realistically representative of a real wireless IoT application.

As can be seen in Figure 5.4, the cycle period lengths have a mean of 22.366 s with a standard deviation of 0.02275 s. There are also a few outliers at around 22.24 s, of which the cause is unknown except for potential differences in the sleep cycle of the DUT. Selecting a cycle to segment representative of an average cycle while maintaining the ability to compare its results to the reference is not viable due to the small but existent variation in cycles. Because of this, we select five consecutive cycles for the segmentation to get a better representation of an average cycle. Any greater variations have been handled as mentioned above. This has implications when comparing the segmentation to the references as the same exact periods need to be compared for the results to be reasonable, but it also makes it easier to visualize the decrease in voltage between cycles.

6.4 Battery health

The manufacturer has stated that the expected degradation in capacity of the Li-Polymer battery used in this study is expected to be no more than 30% when undergoing 500 certain charge/discharge cycles [13]. The cycle is as follows:

- Charge with constant current 0.5 C or 95 mA to 4.2 V.
- Charge with constant voltage 4.2 V to 0.05 C or 9.5 mA.
- Rest for 30 minutes.
- Discharge with constant current 0.5 C or 95 mA to 3.0 V.

The test is carried out in room temperature conditions, namely 20 ± 5°C. Note that the charging regimen in this study deviates from the manufacturer's specification of the test for capacity loss. In this study, the battery is charged at a faster rate, with a higher constant current. Before starting the study, the battery had been subject to approximately 30 charge-discharge cycles. In light of these considerations, it is reasonable to anticipate some decline in battery health, as in a potential decrease in capacity. However as the charge-discharge cycles were done as consecutive runs, the effect of this decline on the results should be minimal. In a perfect study, one would try to execute each test run on a fresh battery, each validated to have similar performance to that of the group of batteries. In spite of that, there would still be a source of error considering it is impossible to find even a pair of batteries with identical performance, nonetheless a larger set of them. Because all runs were done on the same battery, there is no error contributed by variations between different batteries.

6.5 Answers to research questions

In this thesis, the aim was to find an algorithm for segmenting a power consumption curve from a device to imitate the load on a battery with a limited number of segments. The study examines two general research questions:

- RQ1: What would an algorithm generating a non-uniform piecewise constant approximation of a power consumption profile look like?
- RQ2: How should the error of the resulting discharge curve be calculated in order to evaluate whether an approximation is satisfactory?

Considering RQ1 it appears that with the limitation in number of steps, exact recreation and composition of the cycle are not very important, as long as the total energy is conserved. We have found that the amplitudes and lengths of the peaks are important for the voltage to drop down to the cutoff voltage and become more important towards the end. With a different battery that is more or less peak sensitive than the one used in this study, the results might differ. Thus more data using different kinds of batteries is needed in order to make any general conclusions.

For RQ2, looking at the voltage level curve gives only a good overview of how well the battery behavior is approximated. However, to obtain a more complete picture, additional

data is required such as the consumed energy over time combined with the time of a specific voltage cutoff and the total energy up until that point.

6.6 Future work

The results found in this study are based on data from a single battery. To attain a well-rounded idea of the performance of our segmentation algorithms and method, multiple battery types of different chemistries should be tested and analyzed. Because of their different properties and characteristics such as sensitivity to peaks mentioned in Section 2.1.1, they may behave and respond differently to peaks and sleep compared to the battery used in our testing. This could potentially make the method used and described sub-optimal and invalidate the conclusions being made here.

Based on the fact that the power profile of our DUT Particle Argon is not perfectly either constant power or constant current, leading to the difference seen in energy at the two voltage cutoffs, multiple measurement points in the data could be selected for segmentation. This could be e.g. one in the vicinity of 50 % consumed energy as in this study, as well as one closer to the end where the power of the Particle Argon tapers off. How these points should be combined must then be investigated, for example by some form of interpolation. Since not all firmware and devices behave the same, especially with regards to the type of load they behave as described in Section 2.1.2, the analysis should be performed using multiple different DUTs with different firmware. This can also potentially have a profound impact on the findings and conclusions arrived at.

Depending on what findings are made on how peaks impact different types of batteries, the algorithm might be extended with the ability to merge nearby peaks. Currently, the algorithm does not recreate all peaks perfectly, but they are slightly attenuated. Batteries are more affected by high loads during longer periods than if they are allowed to recover momentarily between them such as for nearby peaks. If the algorithm can merge those peaks, the effect on the battery may more closely represent how the full non-attenuated peaks would affect it.

With fewer hardware limitations, such as more steps available, the power consumption curve could be recreated more precisely. Optimally, we would need an infinite amount of steps to perfectly recreate the curve. However, that will not be possible since as we saw in Section 6.3.1, the hardware does take some time to regulate the current. In addition, the clock frequency of the hardware limits the measurements to the 250 ksps listed in Table 2.1. If we had an infinite number of steps, or many more than currently available, another alternative could be to run the segmentation not only for the five periods selected for this study, but for the whole power consumption profile. This would eliminate having to combine multiple measurement points.

Chapter 7
Conclusions

Our aim is to analyze how a power consumption profile for a battery-powered device can be segmented with a limited number of non-uniform steps while keeping the most important characteristics. The most important characteristics were found to be the total consumed energy in conjunction with the heights and lengths of peaks in the curve.

An algorithm, High peak resolution, for recreating peaks was tested against three simple benchmark algorithms. The benchmarks consist of two algorithms performing simple average downsampling and one doing maximum peak amplitude imitation. High peak resolution aims to recreate the peaks with all input data points while replacing the area between peaks with an average. All four algorithms were tested, and consumed within 1.7 % of the total energy of the Particle Argon at the set cutoff of 3.17 V. At different points in time, they were all within around 10 ‰ of the consumed energy of the DUT. The benchmark algorithm consisting of a single maximum amplitude peak was found to most accurately match the total energy spent and battery life of the Particle Argon at a 3.55 V cutoff, after which the behavior of the Argon changes reducing the validity of the results. As metrics for the reproduction of load behavior, the total energy spent and battery lifetime, in addition to the voltage dropoff curve are important.

The study was conducted using a single battery to reduce the effect of variations between batteries. For the scope of the thesis, multiple battery chemistries were not tested. However, had a different battery been used, different results might have been obtained. The same reasoning applies to different IoT devices, where this study is limited to the Particle Argon. Any conclusions made in this thesis are limited by the available data.

References

[1] Kevin Ashton. That 'Internet of Things' Thing. *RFiD Journal*, 22:97–114, 2009.

[2] Leif Bergerhoff, Joachim Weickert, and Yehuda Dar. Algorithms for piecewise constant signal approximations. In *2019 27th European Signal Processing Conference (EUSIPCO)*, pages 1–5, 2019.

[3] Sutando Darmawan and H.L. Chan. A new battery model for use with battery energy storage systems and electric vehicles power systems. In *2000 IEEE Power Engineering Society Winter Meeting. Conference Proceedings (Cat. No.00CH37077)*, volume 1, pages 470–475 vol.1, 2000.

[4] United Nations Population Fund. World Population Dashboard. `https://www.unfpa.org/data/world-population-dashboard`.

[5] Alice Gomstyn and Alexandra Jonker. What is smart farming?, 2023. Accessed: 2024-06-03.

[6] Kareeb Hasan, Neil Tom, and Mehmet Yuce. Navigating battery choices in iot: An extensive survey of technologies and their applications. *Batteries*, 9:580, 12 2023.

[7] Peter Hoffman. High pulse drain impact on CR 2032 coin cell battery capacity. 2011.

[8] Dima Kilani, Baker Mohammad, Hani Saleh, and Mohammad Ismail. Ldo regulator versus switched inductor dc-dc converter. In *2014 21st IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pages 638–641, 2014.

[9] Reiner Korthauer. *Lithium-Ion Batteries: Basics and Applications*. Springer Berlin Heidelberg, 2018.

[10] Yan Li, Yunquan Dong, Pingyi Fan, and Khaled Letaief. How far are wireless networks from being truly deterministic? 11 2022.

[11] David Linden and Thomas Reddy. *Handbook of Batteries*.

[12] Knud Lasse Lueth. The State of IoT - Spring 2023. May 2023.

[13] MikroElectronica. *Li-ion Polymer Battery*. https://download.mikroe.com/documents/datasheets/HPL402323-2C.pdf.

[14] Particle. *Argon Datasheet*. https://docs.particle.io/reference/datasheets/wi-fi/argon-datasheet/.

[15] Qoitech. Otii Ace Pro by Qoitech. https://www.qoitech.com/otii-ace/.

[16] Alberto De Santis, Tommaso Giovannelli, Stefano Lucidi, Mauro Messedaglia, and Massimo Roma. An optimal non-uniform piecewise constant approximation for the patient arrival rate for a more efficient representation of the Emergency Departments arrival process. Technical report, 2020.

[17] SciPy API reference. scipy.signal.find_peaks. https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html.

[18] SciPy API reference. scipy.signal.peak_prominences. https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.peak_prominences.html#scipy.signal.peak_prominences.

[19] Sairatun Nesa Soheli, Ummay Shabrina Aney, and Shahidul Islam Mahmud. Designing a highly effective dc-dc buck converter for sustainable electronic applications. In *2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*, pages 178–182, 2021.

[20] Texas Instruments. *Understanding the Terms and Definitions of LDO Voltage Regulators*. https://www.ti.com/lit/an/slva079/slva079.pdf?ts=1713328870747.

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2024-05-30

EXAMENSARBETE Segmenting a power consumption profile for approximating battery behavior
STUDENTER Patrik Gyllvin, Maria Svensson
HANDLEDARE Jonas Skeppstedt (LTH), Björn Rosqvist (Qoitech AB), Andreas Olausson (Qoitech AB)
EXAMINATOR Flavius Gruian (LTH)

Imitera effektförbrukningen hos en batteridriven enhet

POPULÄRVETENSKAPLIG SAMMANFATTNING **Patrik Gyllvin, Maria Svensson**

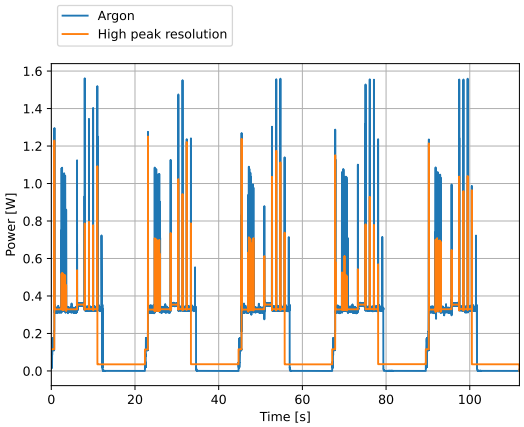
Vid utveckling av en batteridriven hårdvara är valet av batteri en viktig del av processen. Målet med vårt examensarbete är att underlätta denna process genom möjligheten att parallellt utvärdera flera olika batterialternativ.

För denna studie används Qoitech's verktyg Otii Ace Pro som kan lasta batterier efter en förbestämd stegfunktion. Verktyget begränsar stegfunktionen gällande antal steg och målet blir att efterlikna effektförbrukning så bra som möjligt med avseende på denna begränsning. Vår egenutvecklade algoritm, High peak resolution, omvandlar effektförbrukningskurvan till denna stegfunktion. Förbrukningskurvan är en sampling av effekten med en samplingshastighet på 1000 sampel per sekund. Då begränsningen ligger på 1000 steg måste förbrukningskurvan segmenteras i längre segment för att möta begränsningen.

I studien används en Particle Argon som batteridriven enhet. Den har en firmware som cyklar en aktiv period där Argon kopplar upp sig på Wi-Fi, för att sedan går ner i en viloperiod i tio sekunder. Fem sådana cykler skapar en effektförbrukning enligt den blå kurvan i figuren. Enheten användes för att lasta ett batteri tills batterispänning inte räckte för att driva enheten.

Idéen med algoritmen är att ha korta segment kring spikar i effektförbrukningen, då dessa innehåller information som antas ha stor påverkan på batteriet. Emellan pikarna används längre segment. Resultatet ses som kurvan i orange i figuren. Som benchmark används en vanlig nedsampling av kurvan till 8 sps.

Det enklaste sättet att tänka vid val av batteri är *hur länge kan batteriet driva min enhet?* I förlängningen är det kopplat till hur mycket energi som kan nyttjas ur batteriet. Algoritmen utvärderades med avseende på hur bra den matchar energiförbrukningen samt hur länge spänningsnivån räcker för att driva enheten.



Differensen mellan energiförbrukning över tid är väldigt liten mellan algoritmen och Argon, mindre än 1‰ genom stora delar av urladdningen. Tiden tills batteriet är urladdat skiljer sig med dryga procenten. Resultaten är väldigt lika vår benchmark. Algoritmen bör utvärderas på olika enheter med olika batterier för att kunna dra ordentliga slutsatser.