electronics

MDPI

Article

Toward Energy Efficient Battery State of Charge Estimation on Embedded Platforms

Haris Turkmanović *, Ivan Popović and Vladimir Rajović

Department of Electronics and Digital Systems, School of Electrical Engineering, University of Belgrade, Bulevar kralja Aleksandra 73, 11120 Belgrade, Serbia; popovici@etf.bg.ac.rs (I.P.); rajo@etf.bg.ac.rs (V.R.)
* Correspondence: haris@etf.bg.ac.rs

Abstract: Recent studies have focused on accuracy as the key state of charge (SoC) estimation algorithms' performance metrics, whereas just a few of them compare algorithms in terms of energy efficiency. Such a comparison is important when selecting an algorithm that should be implemented on a resource-constrained, low-power embedded system. In this paper, recursive model-based SoC estimation algorithms, such as the extended Kalman filter, have been identified as well-suited solutions for implementation on an embedded platform, providing a good compromise between estimation accuracy and computational complexity that is correlated to energy consumption. Assuming that a decrease in the estimation rate will result in a decrease in both accuracy and energy consumption of the estimator, a theoretical analysis has been conducted to establish how these two metrics depend on the estimation rate. To verify results obtained in theory, two extended Kalman filter-based SoC estimation algorithms of different complexities have been implemented and compared in terms of accuracy, quantified by root mean square error (RMSE), and energy consumption. The obtained results confirm that for a selected type of recursive model-based SoC estimation algorithm, it is possible to achieve an optimal algorithm estimation rate in the sense of satisfactory accuracy and acceptable energy consumption. The analysis and results presented in this paper establish a foundation for a future development of energy-efficient algorithms for SoC estimation in applications where the energy consumption of the estimation process is comparable to the energy consumption of the complete system.

Keywords: state of charge estimation; embedded systems; energy efficiency; Extended Kalman Filter; estimation period

check for updates

Citation: Turkmanović, H.; Popović, I.; Rajović, V. Toward Energy Efficient Battery State of Charge Estimation on Embedded Platforms. *Electronics* **2024**, *13*, 4256. https://doi.org/10.3390/electronics13214256

Academic Editor: Jahangir Hossain

Received: 11 September 2024
Revised: 14 October 2024
Accepted: 28 October 2024
Published: 30 October 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Estimating battery state parameters within battery-powered embedded systems is an indispensable task for proper system operation. One of the battery's most important state parameters is the state of charge (SoC). It represents the ratio of the current remaining capacity to the maximum battery capacity [1,2]. Within a battery-powered system, the SoC value is used not only to provide real-time information about the remaining battery lifetime but also to prevent accelerated battery degradation due to overcharge and over-discharge.

Accurately estimating the SoC value is challenging due to the varying technical and environmental behavior of the battery, including the effects of aging, temperature fluctuations, and charge–discharge cycles [3–5]. The list of challenges is even longer when the SoC estimation needs to be implemented on energy and computationally constrained platforms such as battery-powered embedded systems.

Countless SoC estimation methods have been developed to enhance estimation accuracy [6–18]. Most research introducing novel SoC estimation methods primarily focuses on quantifying accuracy separately or together with computational time achieved on general-purpose machines. A few papers that discuss implementations on actual embedded platforms provide computational complexity analysis along with accuracy analysis [19–22] but without analysis of the SoC algorithm's impact on total system energy consumption.

Enhancing the energy efficiency of battery-powered devices is a crucial issue [23], especially for battery-powered embedded systems with low operating currents and battery capacities, such as IoT sensor nodes [24], wireless sensor networks [25], battery-powered medical devices, and others. To increase energy efficiency, these systems typically operate in two modes: active mode, when data are being processed, and low power mode, when the system is inactive. Implementation of an SoC estimation algorithm, to be executed while the system is in active mode, requires the selection of an algorithm with low computational cost. Effective usage and implementation of such algorithms can reduce the duration of the active mode and extend the remaining useful lifetime of the battery. In the context of energy-constrained battery-powered embedded systems, selecting an SoC estimation algorithm may be easier if a comprehensive evaluation, considering both accuracy and energy consumption, is available. Therefore, an analysis that correlates these two important metrics for SoC estimation algorithms should enable easier selection of SoC estimation algorithms that have the potential to be implemented on battery-powered embedded systems.

In this paper, various SoC estimation methods have been reviewed to identify SoC estimation algorithms that may provide a good trade-off between computational cost and accuracy and which have the potential to be implemented as a pure software solution on an embedded platform. The extended Kalman filter (EKF) [5] has been identified as a well-suited choice. The recursive algorithms execute periodically with an estimation rate, acquiring voltage and current samples during each run cycle and processing these samples based on the estimator's state history. Therefore, there is flexibility in selecting the estimation rate, as reducing the estimation rate may decrease the estimator's accuracy, but if this decrease is acceptable, it leads to energy savings.

This study aims to quantify the portion of total system energy consumption consumed by an SoC estimation algorithm in the case of low-power embedded systems. It also investigates how variations in the estimation rate impact both the accuracy of SoC estimation and the overall system energy consumption, with the goal of identifying the optimal estimation rate for a recursive SoC estimation algorithm. Finally, it evaluates the trade-offs between selecting a more complex, energy-demanding, high-accuracy algorithm and a simpler, low-energy, lower-accuracy alternative.

This research first conducts a theoretical analysis, deriving dependencies of the SoC estimation error and energy consumption on the algorithm's estimation rate for a case of an ideal recursive estimator. The estimation error has been quantified by root mean square error (RMSE), whereas we introduced relative algorithm consumption (RAC) to quantify energy consumption. The derived dependencies then serve as the primary research criteria during further analysis. Additionally, an optimization cost function is introduced to enable the numerical determination of the optimal sampling period in scenarios where the configuration of the battery-powered system (including consumption, algorithm duration, CPU clock speed, etc.) and discharge profiles are well known.

To demonstrate the application of the theoretical analysis in a real-world scenario, two EKF algorithms of varying complexities were implemented and evaluated based on their estimation accuracy and the duration of a single run cycle. One of the algorithms is from the category of the computationally simpler algorithms and employs a simple Rint battery model. The algorithm is less precise and has a relatively low energy consumption due to lower computational complexity. The second analyzed algorithm is the EKF, which employs a 2nd-order RC battery model and belongs to the category of complex computational algorithms. It exhibits a high estimation precision and a relatively high energy consumption due to higher computational complexity. The algorithms have been implemented on a PC to verify the functionality and accuracy of the algorithms. To obtain the duration of algorithms single run cycles, both algorithms have been implemented on a microcontroller-based embedded platform.

The paper contributes to the evaluation of algorithms for SoC estimation and the trade-off between invested energy and accuracy. The contributions could be summarized in the following way:

- After a comprehensive review of the literature, the research gap in the area of energy consumption analysis of SoC estimation algorithms has been identified.
- The impact of the estimation rate of recursive SoC estimation algorithms on total energy consumption and estimation accuracy of a battery-powered embedded system has been theoretically quantified.
- Two extended Kalman filters of different complexities have been implemented on an embedded platform and compared in terms of algorithm accuracy and computational complexity.
- The foundation for a future development of energy-efficient SoC estimation for battery powered embedded systems has been laid.

The rest of the paper is organized as follows: In Section 2, the SoC estimation algorithms are reviewed in terms of energy consumption. Theoretical considerations as well as the discussion on the fundamental concepts important for the trade-off between accuracy and energy consumption analysis are given in Section 3. Section 4 describes the experimental setup and the experimental procedures. The extracted values of the estimation model parameters are presented in Section 5. The results obtained are presented in Section 6. Concluding remarks and directions for future work are presented in Section 7.

2. Related Works

Numerous methods for estimating a battery SoC have been developed. Understanding their overall performance, especially in terms of accuracy and computational complexity, is crucial for selecting the one that best meets specific system and battery requirements. The proper way to begin the selection process is to review the existing methods and classify them based on their common characteristics, application fields, advantages, disadvantages, and computational complexity [5].

Several research papers deal with the classification of SoC estimation algorithms based on different selection criteria. Compared to the other research papers that perform classification of SoC estimation algorithms [2,4], the paper [5] provides the most detailed classification of the existing SoC estimation methods, classifying them into six different categories: conventional methods, filter-based methods, observer-based methods, data-driven methods, hybrid methods, and other estimation methods.

Conventional methods are easy to implement and have an acceptable SoC estimation accuracy under certain conditions, therefore qualifying as a possible solution for implementation on embedded platforms [12–14]. However, most of these methods are not used for online battery SoC estimation because they require knowledge of the initial SoC values, involve significant experimental investments, and have low resilience to measurement noise [1]. Data-driven methods are too complex in terms of computational complexity, and that disqualifies them from being implemented on modest computational platforms such as embedded systems [15–17]. Hybrid methods are primarily developed to increase SoC estimation accuracy for specific applications, without analyzing their execution performances. Model-based methods seem the best solution, with a potential to provide the best trade-off between SoC estimation accuracy and algorithm complexity [6–11,18,26–29] and are the most suitable choice for online SoC estimation with respect to the high estimation accuracy and reasonable computing effort [30]. Consequently, the model-based methods appear to be a good choice for implementation on resource-constrained embedded platforms.

The implementation process of model-based SoC estimation methods includes two main steps: (1) battery modeling and extraction of parameters; (2) selection and implementation of an observer estimation algorithm [31,32]. Selection of a battery model is the first and crucial step, as the computational performance and accuracy of SoC estimation heavily depend on the model selected [33]. The battery model refers to physical aspects of a battery, such as electrical, thermal, aging, and a combination of these [34]. Based

on [4,35], battery models can be classified into three types, two of which are extensively used within estimation algorithms: physics-based electrochemical models [36–38], and electrical equivalent circuit models [39,40]. Electrochemical models have high accuracy because they describe reactions inside batteries based on chemical/electrochemical kinetics and transport equations [30]. Thus, they are built upon partial differential equations that are difficult to solve with online estimation methods [35,41]. Some of the most used physics-based electrochemical models are the pseudo-two-dimensional model [41] and single particle model [42]. Electrical equivalent circuit models depict the dynamic characteristics of batteries by utilizing a combination of electrical elements, such as voltage sources, resistors, and capacitors, to describe a battery's behavior. The parameters used within this model depend on a battery's aging status and its operating conditions, thus resulting in a gradual degradation of SoC estimation accuracy over time [30]. However, compared to the other models, the electrical equivalent circuit models have the potential to provide the best trade-off between model complexity and accuracy and have been widely used within different estimation algorithms [31,43]. Some of the electrical equivalent circuit models are the internal resistance model [44], the nth-order RC model [45], and the partnership for a new generation of vehicles (PNGV) model [46]. Compared to the other models of this type, the 2nd- and 1st-order RC models are extensively employed because they offer the most balanced compromise between complexity and precision for different discharge profiles.

After selecting a battery model, it is required to perform model parameter identification by either online or offline identification methods [47,48]. The online parameter identification methods [49,50] include determination of parameter values during SoC algorithm execution, whereas the offline parameter identification methods use experimental data to determine values of model parameters before the model is deployed within an observer estimation algorithm. Because the battery's technical characteristics and environmental factors change during operation, online methods for identification of parameters provide better corrections of SoC estimation during operation but lead to a complex implementation and a higher SoC computational time. On the other hand, under certain conditions, offline estimation methods provide reasonable accuracy and at the same time reduce the SoC estimation complexity.

Numerous model-based algorithms can be used to estimate the SoC value. Among them, the algorithms from the Kalman filter family are widely used, because of their simplicity, in combination with various electrical models [51]. Because a battery is modeled as a nonlinear system, a non-linear Kalman filter must be used to estimate the SoC value [31]. Extended Kalman filter (EKF) requires less computing power compared to the other non-linear Kalman filters or filter-based estimation techniques in general [52–54]. However, in most practical implementations, the adaptive extended Kalman filter (AEKF) is utilized instead of the EKF. The AEKF operates similarly to the EKF, but it adjusts the measurement and noise matrices during estimation to enhance robustness to noise, which is very important in real applications.

Many research papers present EKF and AEKF implementations that utilize electrical battery models. The paper [55] proposes an online model-based SoC estimation method that utilizes an AEKF. The 1st-order RC network is used as the equivalent circuit, and model parameters are estimated online using the recursive least square (RLS) method with multiple forgetting factors. The accuracy of the proposed method was tested within a simulation environment, indicating a maximum absolute error less than 0.015%. The EKF presented in the paper [56] provides strong robustness to the uncertainty of the model and the initial value of the noise covariance matrix. The battery model used is the 2nd-order RC model, and the parameters are extracted offline. Battery current and voltage are measured by a custom measurement system based on an STM32 microcontroller. The algorithm presented in [57] can estimate SoC and state of health (SoH) simultaneously. An AEKF uses the 1st-order RC battery electrical model, with online parameter identification based on the RLS method. The reported estimation accuracy is about 2%.

The paper [58] proposes a hybrid approach to achieve a better trade-off between accuracy and computational complexity. The proposed approach combines a low computational cost ampere-hour integral SoC estimation method and an AEKF of high accuracy. It is demonstrated that accuracy is almost the same compared to the case when only an AEKF is utilized while computational cost is reduced by almost 25%. The paper [21] compares two model-based algorithms for SoC estimation, unscented Kalman filter (UKF) and EKF, in the sense of computational time and accuracy, in the case of signals with different noise levels. Both algorithms utilize the 1st-order RC network and are implemented on the Linux-based embedded platform, Raspberry Pi. It is demonstrated that the UKF requires more computational time in comparison to the EKF, at the same time providing high accuracy and robustness to noise. The work in [22] utilizes the 1st-order RC model and implements EKF on an FPGA platform to achieve a real-time SoC computation. It introduces computational techniques, so matrix multiplication time is up to 4 times reduced compared to the conventional matrix multiplications. This is a rare case where results of computational time on a real platform are provided. The paper [59] presents an EKF based on the 1st-order RC electrical model. The battery model parameters are extracted offline, using the RLS method. The EKF is first analyzed in MATLAB/Simulink (https://www.mathworks.com/) and then implemented on a PIC18F4550 microcontroller. A similar implementation on the Arduino embedded platform is performed within research in [19]. Paper [20] introduces dual EKF, which enables a high robustness to the noise of the current sensor and incorrect filter initialization. The algorithm simultaneously estimates battery state and the 2nd-order RC model parameters. This filter is implemented on the Linux-based embedded platform NXP KL25Z128. Besides the SoC and model parameter estimation, the algorithm estimates the SoH and system state of power. It is demonstrated that the achieved estimation error does not exceed 5%.

Table 1 provides a summary of the mentioned research papers, employing SoC estimation algorithms based on EKF or AEKF algorithms coupled with battery electrical models. For each entry in Table 1, the battery model used, the parameter identification method, and the type of estimation algorithm. Additionally, each paper is categorized based on the type of analysis conducted: (A) for accuracy analysis, (C) for computational complexity analysis, and (E) for algorithm energy consumption analysis.

Table 1. Overview of research papers which utilize SoC estimation based on EKF and electrical models.

Ref.	Battery Model	Parameter Identification Method	Estimation Algorithm	Analysis			Implemented on Embedded Platform
				A	C	E	
[55]	1st-order RC	Online/RLS	AEKF	Yes	No	No	No
[56]	2nd-order RC	Offline/RLS	AEKF	Yes	No	No	No
[57]	1st-order RC	Offline/RLS	Dual AEKF	Yes	No	No	No
[58]	1st-order RC	Offline/curve fitting	AEKF/Ampere-Hour	Yes	Yes	No	No
[59]	1st-order RC	Offline/RLS	EKF	Yes	No	No	Yes
[19]	1st-order RC	Offline/RLS	EKF	Yes	No	No	Yes
[20]	2nd-order RC	Offline/curve fitting	Dual EKF	Yes	No	No	Yes
[21]	1st-order	Offline/curve fitting	EKF/Unscented Kalman Filter	Yes	Yes	No	Yes
[22]	1st-order RC	Offline/curve fitting	EKF	Yes	Yes	No	Yes
This paper	Rint/2nd order RC	Offline/curve fitting	EKF	Yes	Yes	Yes	Yes

Reviewing the papers that utilize battery state of charge (SoC) estimation methods based on recursive model-based algorithms reveals that all of them conduct accuracy analysis. However, only a few of them evaluate computational complexity, even those implementing algorithms on embedded platforms. None of the analyzed papers includes an analysis of the algorithms' energy consumption. In our paper, we conduct a detailed

theoretical analysis of recursive model-based SoC estimation algorithms to establish a systematic approach for evaluating their energy consumption. We apply this analysis to two actual implementations based on the EKF to validate the theoretical results. The theoretical analysis and results that demonstrate the potential of such an analysis in real battery-powered applications are presented in the further text.

3. Theoretical Considerations and Metrics

In the case of a battery-powered embedded system with integrated battery SoC estimation, it is possible to separate the instantaneous power of the SoC estimation algorithm $P_A(t)$ deom the instantaneous power of the rest of the system $P_S(t)$. The total instantaneous battery output power $P_B(t)$ is then the sum of the two powers, as presented in Figure 1.

$$P_{\mathrm{B}}(t) = P_{\mathrm{S}}(t) + P_{\mathrm{A}}(t) \tag{1}$$

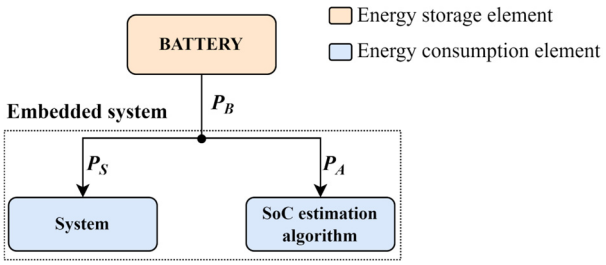


Figure 1. Energy consumption blocks of battery-powered embedded system with integrated SoC estimation algorithm.

The value of instantaneous power without the SoC estimation, $P_{\mathrm{A}}(t)$, depends on the consumption of various hardware and software components that are part of the system and not directly involved in the SoC estimation. These components may be various hardware elements, such as sensors and actuators, and the software that implements the complete system logic (without the SoC estimation) on a CPU based embedded platform. The instantaneous power of an SoC estimation algorithm, $P_{\mathrm{A}}(t)$, is determined by the power consumption of hardware and software components that are part of the SoC estimation algorithm. The hardware components may be shunt resistors for current sensing, operational or instrumentational amplifiers, AD converters, etc. In a general case, the SoC estimation algorithm logic may be implemented in the form of a separate integrated circuit, a dedicated hardware block within a programmable logic device, or as a software only solution implemented on a CPU.

The accuracy and energy consumption of a recursive model-based SoC estimation algorithm are heavily impacted by the estimation period. This section presents a theoretical analysis aimed at numerically quantifying how the estimation rate affects both estimation accuracy and energy consumption. The goal is to derive a general optimization metric that can be used to fine-tune the estimation rate, balancing accuracy and energy efficiency. The first subsection establishes a set of equations that will serve as key metrics for the analysis, while the second introduces the extended Kalman filter, used to validate the theoretical findings.

3.1. Analysis of Estimation Period Influence on an SoC Algorithm Estimation Accuracy and Energy Consumption

In this subsection, a relation that quantifies the impact of different estimation periods on the estimation accuracy and energy consumption of an SoC algorithm is derived. The analysis is divided into three parts. The first part examines how the estimation period affects the accuracy of SoC estimation, whereas the second part analyzes the energy consumption of the algorithm. The third part introduces the optimization criteria that can be used to determine the optimal estimation period.

3.1.1. Estimation Algorithm Accuracy as a Function of Estimation Period

Different metrics are used for assessing the accuracy of state of charge (SoC) estimation algorithms. Common metrics include, among others, absolute maximum error (AME), mean absolute error (MAE), standard deviation (SD), and root mean square error (RMSE) [26]. For now, there is no consensus in the literature which of the metrics is the most appropriate. In this paper, the RMSE is used due to the possibility to make an elegant theoretical analysis and its moderate stringency. Some other metrics might be more useful for a specific application, but the focus of this paper is to show that it is possible to achieve a trade-off between accuracy and energy consumption, and RMSE is used as an exemplary metric. The methodology presented may be used for arbitrary metrics.

If the difference between the estimated and true SoC values at the ith estimation instant is represented as $e[i]$, and there is a total of N estimation instants during a full battery discharge, then the RMSE is expressed by the following equation:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} e[i]^2}{N}} \tag{2}$$

To express RMSE as a function of the estimation period T, it is essential to first establish ground truth or reference SoC values that will be used as nominally accurate values for calculations of RMSE for various estimation periods. For this purpose, the SoC values obtained with the Coulomb counter algorithm (SoC_{CC}) running at the minimum (base) estimation period T_B available on the system are used. When SoC_T represents the SoC values obtained with a recursive model-based SoC estimation algorithm at a selected estimation period $T(\geq T_B)$, then the corresponding RMSE is defined by the following equation:

$$RMSE(T) = \sqrt{\frac{\sum_{i=1}^{N} (SoC_T[i] - SoC_{CC}[i])^2}{N}} \tag{3}$$

In the general case, the RMSE (3) encapsulates two errors: the error due to the utilized model and the error occurring because of the increase in the estimation period. To separate these errors and provide for theoretical analysis of influence of the estimation period on estimation accuracy, the SoC values obtained with the recursive model-based estimation algorithm (SoC_B) running at minimal estimation period T_B are used. The three different used SoC values (SoC_{CC}, SoC_T, and SoC_B), with corresponding differences $e_B = SoC_B - SOC_{CC}$, $e_T = SoC_T - SOC_B$, and $e = SoC_T - SOC_{CC}$, are illustrated in Figure 2.

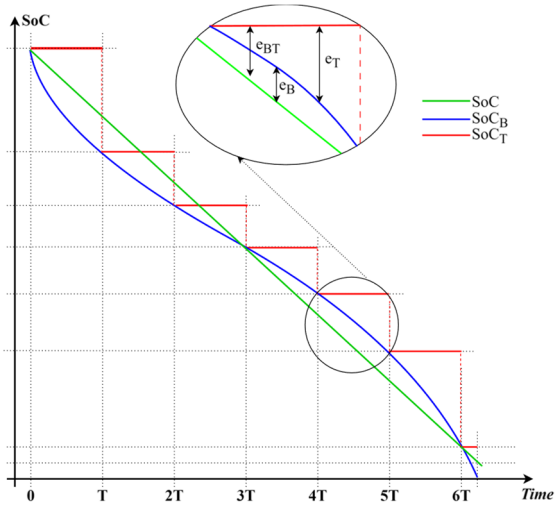


Figure 2. Illustration of SoC values obtained with: Coulomb counter algorithm with minimal estimation period (SoC_{CC}), model-based recursive algorithm with minimal estimation period (SoC_B) and model-based recursive algorithm with an arbitrary estimation period $T > T_B$ (SoC_T).

Based on Figure 2, it is possible to define two different RMSEs:

- $RMSE_{\mathrm{B}}$, quantifying the quality of estimator when executing at a minimum sampling period; this error is due to the model used not being absolutely accurate. The error arises from the difference between SoC_{B} and SoC_{CC}, e_{B}, and can be expressed by the relation

$$RMSE(T) = \sqrt{\frac{\sum_{i=1}^{N}(SoC_{\mathrm{T}}[i] - SoC_{\mathrm{CC}}[i])^2}{N}} \tag{4}$$

where

$$e_{\mathrm{B}}[i] = SoC_{\mathrm{B}}[i] - SoC_{\mathrm{CC}}[i] \tag{5}$$

- $RMSE_{\mathrm{T}}$, representing the error between the estimator running at a minimum estimation period T_{B} and the estimator running at an arbitrary estimation period $T > T_{\mathrm{B}}$. The error arises from the difference between SoC_{T} and SoC_{B}, e_{T}, and can be expressed by the relation

$$RMSE_{\mathrm{T}} = \sqrt{\frac{\sum_{i=1}^{N} e_{\mathrm{T}}[i]^2}{N}} \tag{6}$$

where

$$e_{\mathrm{T}}[i] = SoC_{\mathrm{T}}[i] - SoC_{\mathrm{B}}[i] \tag{7}$$

From Figure 2, total instantaneous error of an estimator running at an arbitrary estimation period $T > T_{\mathrm{B}}$ is $e[i] = SoC_{\mathrm{T}}[i] - SoC_{\mathrm{CC}}[i] = e_{\mathrm{T}}[i] + e_{\mathrm{B}}[i]$. The total RMSE, as defined by (3), may be expressed as function of (4) and (6)

$$RMSE(T) = \sqrt{\frac{\sum_{i=1}^{N} e_{\mathrm{B}}[i]^2}{N} + \frac{\sum_{i=1}^{N} e_{\mathrm{T}}[i]^2}{N} + 2\frac{\sum_{i=1}^{N} e_{\mathrm{T}}[i] e_{\mathrm{B}}[i]}{N}} \tag{8}$$

Assuming that $e_{\mathrm{T}}[i]$ and $e_{\mathrm{B}}[i]$ are uncorrelated, the total RMSE can be expressed as a function of $RMSE_{\mathrm{B}}$ and $RMSE_{\mathrm{T}}$, by the equation

$$RMSE(T) = \sqrt{RMSE_{\mathrm{B}}^2 + RMSE_{\mathrm{T}}^2} \tag{9}$$

The Equation (9) combines two errors and enable analysis of influence of the estimation period on estimation accuracy separately from the analysis of the model-based accuracy.

While the value of $RMSE_{\mathrm{B}}$ can be determined by measurement only, it is possible to analyse $RMSE_{\mathrm{T}}$ theoretically. For the sake of the analysis, the assumption is that the values of SoC_{B} are obtained with an ideal recursive model-based estimator whose $RMSE_{\mathrm{B}}$ equals 0. If such an estimator executes with minimal estimation period T_{B} on a battery powered embedded system whose battery capacity is C and whose average battery load current is I_{B}, the number of estimators run cycles N_{B}, during a full battery discharge interval T_{D} is given by

$$N_{\mathrm{B}} = \left\lfloor \frac{T_{\mathrm{D}}}{T_{\mathrm{B}}} \right\rfloor = \left\lfloor \frac{C}{I_{\mathrm{B}} T_{\mathrm{B}}} \right\rfloor \tag{10}$$

and the sequence of the estimated SoC values $SoC_{\mathrm{B}}[i]$ is

$$SoC_{\mathrm{B}}[i] = \left(1 - \frac{I_{\mathrm{B}} T_{\mathrm{B}}}{C} i\right) 100\%;\ i \in [0, N_{\mathrm{B}}] \tag{11}$$

If the estimator executes with estimation period $T \geq T_{\mathrm{B}}$, there are N estimator run cycles during a full battery discharge interval

$$N = \left\lfloor \frac{T_{\mathrm{D}}}{T} \right\rfloor = \left\lfloor \frac{C}{I_{\mathrm{B}} T} \right\rfloor \tag{12}$$

and the corresponding estimated SoC values $SoC_{\mathrm{T}}[i]$ are

$$SoC_{\mathrm{T}}[i] = \left(1 - \frac{I_{\mathrm{B}}T}{C}i\right)100\%;\ i \in [0, N]. \tag{13}$$

The two SoC sequences are illustrated in Figure 3, for one full battery discharge interval, and $N = 6$. As mentioned above, the estimator is presumed to be ideal when executing at the minimum estimation period.

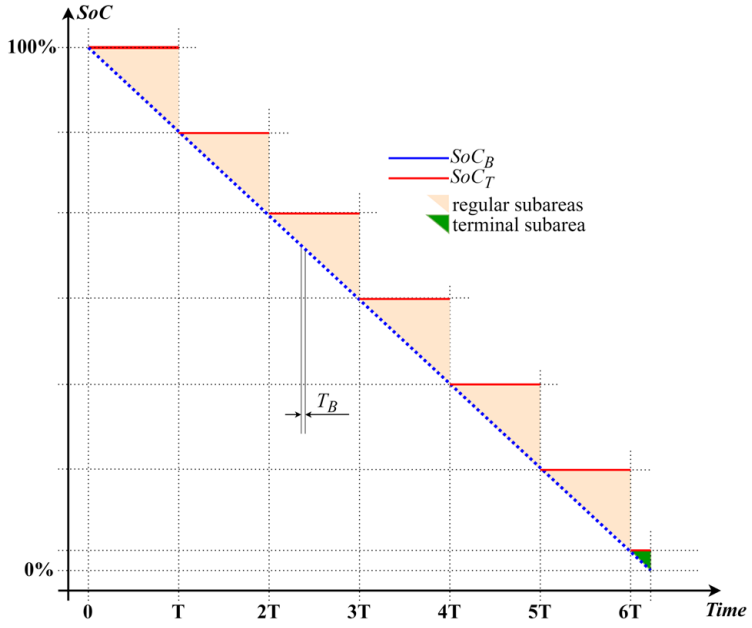


Figure 3. The sequences $SoC_{\mathrm{T}}[i]$ and $SoC_{\mathrm{B}}[i]$ during a full battery discharge interval, for an estimation period T.

As seen in the figure and inferring from (11) and (13), $SoC_{\mathrm{T}}[i]$ represents a decimated, by factor T/T_{B}, version of $SoC_{\mathrm{B}}[i]$. The area between $SoC_{\mathrm{B}}[i]$ and $SoC_{\mathrm{T}}[i]$ corresponds to the deviation of the decimated estimator to the original and consists of N identical regular subareas (shown in ochre) and a truncated terminal subarea (shown in green).

A detailed view of a kth regular subarea is shown in Figure 4. The blue dots represent ideally estimated SoC values ($SoC_{\mathrm{B}}[i]$) whereas red dots represent decimated ideally estimated SoC values ($SoC_{\mathrm{T}}[i]$).

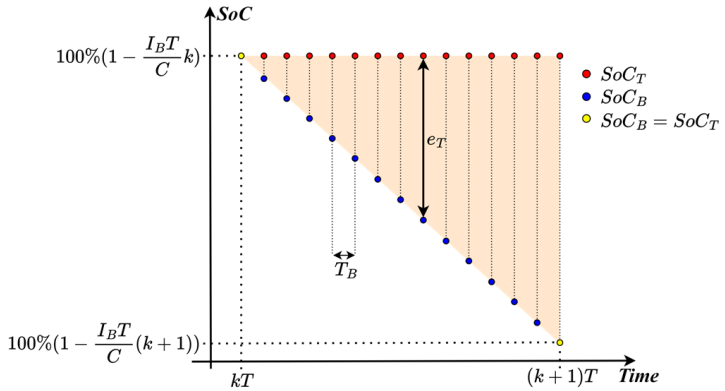


Figure 4. nth full discharge area.

There is a total of $M = T/T_{\mathrm{B}}$ samples (always an integer since T is a multiple of T_{B}) of $SoC_{\mathrm{B}}[i]$ in the regular subinterval, and a total of $M_{\mathrm{T}} = N_{\mathrm{B}} - NM$ samples in the terminal

subinterval. It is possible to derive an equation quantifying error of the decimated estimator compared to the ideal

$$e_{\mathrm{T}}[i] = \frac{I_{\mathrm{B}}T}{CM}i = \frac{I_{\mathrm{B}}T_{\mathrm{B}}}{C}i, i \in [0,\ M-1] \tag{14}$$

The squared error sum over a regular kth discharge subinterval SES_{R}, and the squared error sum over the truncated terminal subinterval SES_{T} can be expressed as

$$SES_{\mathrm{R}} = \sum_{i=0}^{M-1} e_{\mathrm{T}}^2[i] = \left(\frac{I_{\mathrm{B}}T_{\mathrm{B}}}{C}\right)^2 \sum_{i=0}^{2M-1} i^2 = \left(\frac{I_{\mathrm{B}}T_{\mathrm{B}}}{C}\right)^2 \frac{(M-1)M(2M-1)}{6}, \tag{15}$$

$$SES_{\mathrm{R}} = \sum_{i=0}^{M_{\mathrm{T}}-1} e_{\mathrm{T}}^2[i] = \left(\frac{I_{\mathrm{B}}T_{\mathrm{B}}}{C}\right)^2 \sum_{i=0}^{2M_{\mathrm{T}}-1} i^2 = \left(\frac{I_{\mathrm{B}}T_{\mathrm{B}}}{C}\right)^2 \frac{(M_{\mathrm{T}}-1)M_{\mathrm{T}}(2M_{\mathrm{T}}-1)}{6} \tag{16}$$

Now, $RMSE_{\mathrm{T}}$ (6) can be expressed as a function of M

$$RMSE_{\mathrm{T}} = \sqrt{\frac{N \times SES_{\mathrm{R}} + SES_{\mathrm{T}}}{N_{\mathrm{B}}}} = \frac{I_{\mathrm{B}}T_{\mathrm{B}}}{C}\sqrt{\frac{N(M-1)M(2M-1) + (M_{\mathrm{T}}-1)M_{\mathrm{T}}(2M_{\mathrm{T}}-1)}{6N_{\mathrm{B}}}} \tag{17}$$

In the case when T_{D} is a multiple of T (and consequently T_{B}), then $M_{\mathrm{T}} = N_{\mathrm{B}} - NM = \frac{T_{\mathrm{D}}}{T_{\mathrm{B}}} - \frac{T_{\mathrm{D}}}{T}\frac{T}{T_{\mathrm{B}}} = 0$, and Equation (17) simplifies to

$$RMSE_{\mathrm{T}} = \frac{I_{\mathrm{B}}T_{\mathrm{B}}}{C}\sqrt{\frac{(M-1)(2M-1)}{6}} \tag{18}$$

Equations (17) and (18) are suitable for a numerical analysis of influence of the estimation period T on RMSE (3). For such an analysis, it is useful to determine the range of values of the estimation period wherein the analysis makes sense. The estimation period cannot be smaller than the base estimation period T_{B}. Regarding the maximum possible estimation period, it is determined from the duration of full battery discharge for a given battery load I_{B}—when two executions of the estimation algorithm are performed over the entire period, at the beginning and at the end of the period. In that case $N = 1$ and the maximum estimation period T_{M} is

$$T_{\mathrm{M}} = \frac{C}{I_{\mathrm{B}}} \tag{19}$$

When estimating at the maximum estimation period, $RMSE_{\mathrm{T}}$ (18) reaches its maximum value, obtained after substitution of (19) in (18) and application of the assumption $\frac{C}{T_{\mathrm{B}}I_{\mathrm{B}}} \gg 1$

$$RMSE_{\mathrm{T_M}} \approx \sqrt{\frac{1}{3}} = 0.58 \tag{20}$$

From now on, Equations (17) and (18) will be used as the SoC estimation accuracy metrics.

3.1.2. Estimation Algorithm Consumption as a Function of Estimation Period

A recursive SoC estimation algorithm periodically executes within the software of a microprocessor. If the estimation period is T, the duration of execution of a single run cycle of the algorithm lasts a portion of the period D $(0 < D \leq T)$. During the execution, the current consumption of the algorithm may be represented by a current I_{Aex}. The current originates from the microprocessor consumption due to the additional duration of its activity, as well as the consumption of peripheries used only for the estimation (AD converter(s) and accompanying circuitry). When the algorithm is not executing, its current consumption is zero. The above is illustrated in Figure 5.

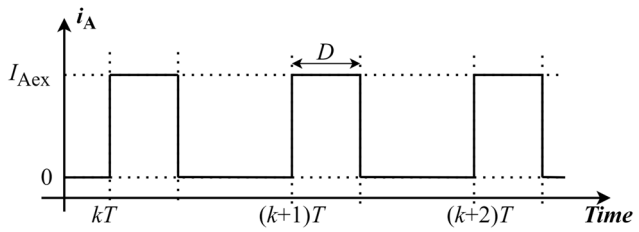


Figure 5. A recursive SoC estimation algorithm execution over several sampling periods.

The relative battery consumption (RAC) may be expressed as a portion of battery capacity consumed for its SoC estimation during a full discharge interval

$$RAC_{\mathrm{T}} = \frac{I_{\mathrm{A}} \cdot T \cdot N}{C} \tag{21}$$

where I_{A} is average current of the SoC estimation algorithm, T is the estimation period, and N is the number of the estimator run cycles during a full battery discharge interval (12).

For the algorithm current consumption as shown in Figure 5, the average algorithm consumption I_{A} is

$$I_{\mathrm{A}} = I_{\mathrm{Aex}} \frac{D}{T} \tag{22}$$

Since the battery load current I_{B} consists of current consumption of the SOC estimation algorithm I_{A} and the current consumption of the rest of the system I_{S}, the number of the algorithm run cycles during a full battery discharge interval N (12) may be written as

$$N = \left\lfloor \frac{C}{(I_{\mathrm{A}} + I_{\mathrm{S}})T} \right\rfloor \tag{23}$$

After substitution of (22) and (23) in (21), and simplifying $\lfloor x \rfloor \approx x$, there is

$$RAC_{\mathrm{T}} = \frac{I_{\mathrm{A}}}{I_{\mathrm{A}} + I_{\mathrm{S}}} = \frac{I_{\mathrm{Aex}}}{I_{\mathrm{Aex}} + I_{\mathrm{S}} \frac{T}{D}} \tag{24}$$

It has been mentioned earlier that the minimum estimation period is determined by T_{B}. The minimum estimation period cannot be lower than the algorithm run cycle duration D, but it could be further restricted by system characteristics, for example, the conversion time of the AD converter.

From (24) implies that, when the average current consumption of the rest of the system I_{S} is negligible when compared to the algorithm execution consumption I_{Aex}, RAC becomes close to $RAC_{\mathrm{T_M}} = 1$.

3.1.3. Estimation Algorithm Performance as a Function of Estimation Period

As shown above, $RMSE(T)$ increases as the estimation period increases. At the same time, RAC decreases. Thus, it should be attainable to identify a value of estimation period T_0 when these two metrics are balanced, that is, there are acceptable both error and current consumption. To combine two different quantities, those should be normalized with respect to maximum values:

$$\widetilde{RAC}_{\mathrm{T}} = \frac{RAC_{\mathrm{T}}}{RAC_{\mathrm{T_M}}}, \; \widetilde{RMSE}(T) = \frac{RMSE(T)}{\sqrt{\frac{1}{3} + RMSE_{\mathrm{B}}^2}} \tag{25}$$

This is a multi-objective optimization method, which could in general be tackled by various optimization strategies. Since it cannot be told which objective is more important, and due to its relative simplicity, the weighted sum method has been chosen. Then, the analytic form of the estimator performance metrics EP could be defined as

$$EP(T) = \frac{1}{k_E \widetilde{RAC}_T + k_R \widetilde{RMSE}(T)} \tag{26}$$

where k_E and k_R are weights for energy consumption and accuracy respectively, dependant on their relative importance, both in the range $[0,\ 1]$ and under the limitation $k_E + k_R = 1$. The optimal value T_0 is obtained as the value of estimation period when the estimator performance has a maximum.

Obtained estimator performance Equation (26) can also be used to compare the two algorithms A_1 and A_2 in terms of achieved performance for a selected estimation period T_S and a given discharge rate. This parameter is named estimators' performance ratio (r), and it is defined as:

$$r = \frac{EP_{A_1}(T_S)}{EP_{A_2}(T_S)} \tag{27}$$

If r is greater than 1, then A_1 achieves better performance for the selected estimation period; otherwise, A_2 provides better performance. However, it should be noted that the two performances might not have a maximum value for the same value of estimation period.

3.2. Computational Complexity of Extended Kalman Filters

To apply the theoretical analysis and showcase its potential for selecting the right algorithm based on well-defined energy requirements, this research focused on selecting two representative algorithms with differing computational complexity while keeping other features, such as initial estimation capability, constant. It was also important to select algorithms that could be implemented purely in software on resource- and energy-constrained platforms. On the basis of a literature review, model-based methods for SoC estimation were chosen, as they offer the best trade-off between accuracy and complexity [60]. The extended Kalman filter (EKF) was selected for its lower computational requirements compared to other nonlinear filters. However, in many practical cases, the adaptive extended Kalman filter (AEKF) is preferred due to its improved noise filtering, even though it requires slightly more computational power [20]. This can reduce the need for additional hardware and software required when implementing the algorithm on low-cost embedded platforms with noisy current and voltage readings. Although the AEKF is often favored in practical applications due to its enhanced noise filtering, it operates on almost the same fundamental principles as the standard EKF. The primary difference is that AEKF continuously adapts its noise matrices during runtime, adding slight complexity and requiring more computational power. Given that the AEKF offers improvements over the EKF, which are not relevant for this research but also increase demand for resources, it was chosen to proceed with the EKF in this research. On the basis of implementations in [19–22,59], it is demonstrated that the EKF provides sufficient accuracy for online SoC estimation while maintaining low computational complexity, making it suitable for implementation on embedded platforms.

EKF algorithm state equations in discrete time domain as well as utilized battery models, and their parameters are described in Appendix A. In a general case, all the battery model parameters are temperature dependent. Because the focus in this research is to analyse EKF algorithm from a computational complexity perspective, from now on in this research, the temperature effects will be neglected, and all the experiments will be conducted at the same ambient temperature.

The EKF is a recursive algorithm that executes periodically, where two procedures are conducted at each execution: predict and update. Table 2 outlines the steps performed in each algorithm iteration.

The steps presented in Table 2 include various matrix operations like addition (A), subtraction (S), multiplication (M), division (D), inversion (I), and transposition (T). To obtain algorithms of different complexities, two EKF algorithms that utilize two battery models are analyzed. Utilized battery models determine the computational complexity of the algorithm by the order of the matrices involved in the calculations. One of the

algorithms is from the category of the computationally simpler algorithms and employs a simple Rint battery model. The algorithm is less precise and has a relatively low energy consumption due to lower computational complexity. The second analyzed algorithm is the EKF, which employs a 2nd-order RC battery model and belongs to the category of complex computational algorithms. It exhibits a high estimation precision and a relatively high energy consumption due to higher computational complexity. Table 3 presents the number of matrix operations within each step as well as the total number of basic mathematical operations such as addition (a), subtraction (s), multiplication (m), and division (d).

Table 2. Extended Kalman filter equations.

Procedure	Equations
Predict	$\hat{x}_{k/k-1} = A_k \hat{x}_{k-1/k-1} + B_k u_{k-1}$ $P_{k/k-1} = A_k P_{k-1/k-1} A_k{}^T + Q_K$
Update	$K_k = P_{k/k-1} H_k{}^T \left(H_k P_{k/k-1} H_k{}^T + R_k \right)^{-1}$ $\tilde{y}_k = y_k - H_k \hat{x}_{k/k-1}$ $\hat{x}_{k/k} = \hat{x}_{k/k-1} + K_k \tilde{y}_k$ $P_{k/k} = (I - K_k H_k) P_{k/k-1}$

Table 3. Number of equations within one EKF iteration.

Procedure	Matrix Operations				Basic Operations					
					Rint Model			2nd-Order RC Model		
	A/S	M	INV	T	a/s	m	d	a/s	M	d
Predict	1	2	0	0	1	2	0	6	12	0
	1	2	0	1	1	2	0	21	18	0
Update	1	4	1	2	1	4	1	15	24	1
	1	1	0	0	1	1	0	3	3	0
	1	1	0	0	1	1	0	3	3	0
	1	2	0	0	1	2	0	18	27	0
$\sum$	6	12	1	3	6	12	1	66	87	1

As seen in Table 3, the number of matrix operations remains consistent regardless of the model used. However, since the matrix order depends on the utilized model, so does the total number of basic operations. When the EKF is implemented on embedded platforms, the duration of computation depends on both the number and types of the fundamental operations. For instance, execution time of multiplication is significantly longer than in a case of subtraction or addition. In the case of a simple Rint model, all matrices are of the 1st order, meaning that all the matrix operations are performed as scalar operations, thus significantly reducing the computational time. In opposition, the 2nd-order model involves matrices of the 3rd order, so the number of basic mathematical operations and computational time are increased. In an ideal case, the execution time of an SoC algorithm based on the EKF equals the duration of all the basic operations performed in each step. In addition, the ratio of two different model-based EKF cycle run execution times depends on the ratio of the numbers of operations within algorithms. As seen in Table 3, the EKF algorithm based on the 2nd-order RC model contains about 8 times more operations than the EKF algorithm based on the simple Rint model. Therefore, it is expected that the EKF based on the 2nd-order RC model executes at least 8 times longer and consumes 8 times more energy than the EKF based on the simple Rint model. However, accuracy improvements achieved with the EKF based on a 2nd-order RC model may not be proportional.

4. Experimental Setup

The equations derived in the previous section describe the relationship between algorithm estimation accuracy and energy consumption as a function of the estimation period. The values of the parameters introduced within these equations must be experimentally determined before the optimization of estimation period is performed. To identify these parameters and validate the theoretical results in a real-world scenario, a series of measurement procedures were carried out. A low-cost microcontroller-based system, named the battery evaluation system (BES), was developed to simplify this process. Within Appendix A are described hardware and software implementation details of BES, while this subsection outlines the experimental procedures for extracting battery parameters and assessing the impact of the estimation period on the accuracy and energy consumption of the EKF algorithm.

A single-cell *Tattu* LiPo battery with a capacity of 500 mAh and a maximum discharge rate of 45C has been used. The values of the battery's voltage and current are sampled with the AD converter integrated inside the BES and streamed to a PC, where they are logged and processed offline within a corresponding Python script.

The BES is used to generate battery load currents of different amplitudes that mimic a system consumption profile. To obtain the values of voltage and current during a full battery discharge interval, the following procedure is conducted:

(1) Battery is fully charged until it reaches its full charge voltage of 4.25 V
(2) Battery rests for 2 h
(3) Battery is fully discharged with a selected current, until its terminal voltage reaches a terminal value of 3.2 V
(4) Battery rests for 4 h

Steps 1–4 are repeated for different discharge rates: C/5 (100 mA), C/2 (250 mA), C (500 mA), 2C (1 A), and 3C (1.5 A). During the discharge process, the values of battery voltage and current are obtained with the minimum (base) sampling period (T_B) available on the BES, which is 2.176 ms.

After the data acquisition process is completed, post-processing is performed offline by means of custom Python scripts. To analyze the influence of different estimation periods on $RMSE(T)$ (9) and RAC_T (21), a data decimation technique is performed so variable estimation periods are obtained from the analyzed dataset.

Two types of EKF algorithms, of different computational complexities, are implemented:

(1) Kalman 0 (K_0)—Kalman algorithm that utilizes simple Rint model.
(2) Kalman 2 (K_2)—Kalman algorithm that utilizes the 2nd-order RC model.

The Python scripts were used for analysis of the algorithms' accuracy, whereas C implementations of both algorithms were run on a STM32H7 microcontroller to obtain the algorithms' run cycle execution time.

The microcontroller hardware configuration is detailed in Table 4. The EKF implementations were designed as pure bare-metal solutions, operating on a minimum hardware configuration. All peripherals were disabled, code execution is performed from Flash memory, and the cache memory was deactivated.

Table 4. Microcontroller configuration.

Name	STM32H723
CPU speed	400 MHz
Active mode consumption	~100 mA
Low power mode consumption	~1 mA

For all the tests, the estimation period varied from the minimum estimation period available on the BES, (2.176 ms), to the maximum estimation period defined by (19). At the maximum estimation period, there are only two EKF run cycles during a full discharge

interval: one at the beginning and one at the end of the interval. Table 5 shows the minimum and maximum estimation periods for the different discharge rates.

Table 5. Range of estimation periods for selected discharge rates.

Discharge Profile	Range of Estimation Period	
	Minimum [ms]	Maximum [s]
C/5	2.176	18,000
C/2	2.176	7200
C	2.176	3600
2C	2.176	1800
3C	2.176	900

The values of SoC_B and SoC_{CC}, as defined in Section 3.1.1 are obtained from values of battery voltage and current at the minimum estimation period. RAC_T, as defined in (24), is calculated from EKF cycle run execution time measured on embedded platform D, the used battery capacity of 500 mAh, and consumption of the embedded platform I_{Aex}.

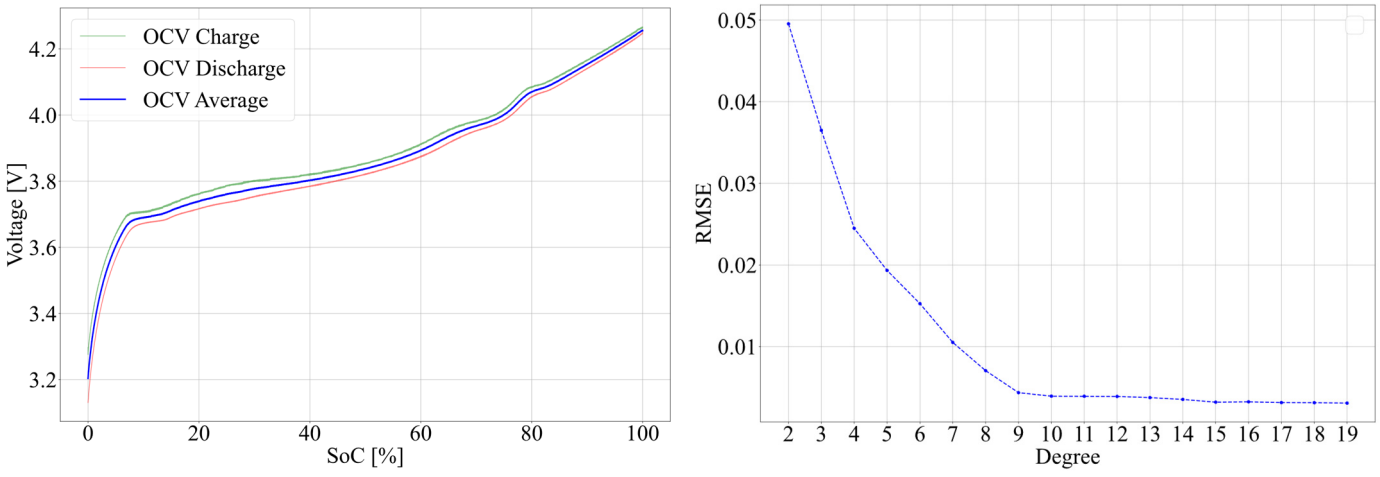
The normalized values $\widetilde{RMSE}(T)$ and $\widetilde{RAC}_T$ are used to find the optimal estimation period for the selected algorithms by application of the estimator performance function (26). Three different optimization criteria were considered for further analysis:

(1) OPT: Algorithm accuracy and energy consumption are equally important, so $k_E = k_R = 0.5$;
(2) OPT_A: The algorithm accuracy is more important than energy consumption, so $k_R = 0.9$ and $k_E = 0.1$;
(3) OPT_E: The algorithm accuracy is less important than energy consumption, so $k_R = 0.1$ and $k_E = 0.9$.

5. Estimated Value of the Model Parameters

The extraction of the values of battery parameters was performed first in the experiment. As a result, obtained are the open circuit voltage (OCV) as a function of SoC [61] and values of resistors and capacitors [51,53–56]. All the parameter extractions were performed offline, using dedicated Python scripts that implement polynomial and exponential fitting techniques.

Figure 6a represents the OCV obtained by averaging battery terminal voltage during charge (green) and discharge (red) for a charge/discharge rate of C/10.



(**a**) (**b**)

Figure 6. (**a**) Open circuit voltage as a function of state of charge; (**b**) RMSE of polynomial approximation of open circuit voltage.

After the OCV function has been obtained, polynomial functions of different orders are tried to find the best numerical approximation. Figure 6b illustrates RMSE for different orders of polynomials in the range from 2 through 19. It is noticeable from the figure that RMSE changes a little from the 9th order of polynomial. Since the increase of the polynomial order yields a negligible increase of accuracy yet increases the computational complexity, the polynomial of the 9th order has been used in the subsequent work. The coefficients of the polynomial are given in Table 6.

Table 6. Coefficients of 9th-order polynomial approximation of OCV function.

a_9	a_8	a_7	a_6	a_5	a_3	a_2	a_1	a_0
1172.22	−5688.17	11,729.04	−3389.35	9238.41	1034.69	−158.71	13.30	3.23

To extract R and C values for both the simple Rint and the 2nd-order RC battery model, the battery is discharged with the dynamic current load profile shown in Figure 7. Parts of the battery terminal voltage waveform used to extract corresponding parameters are also shown in the figure.

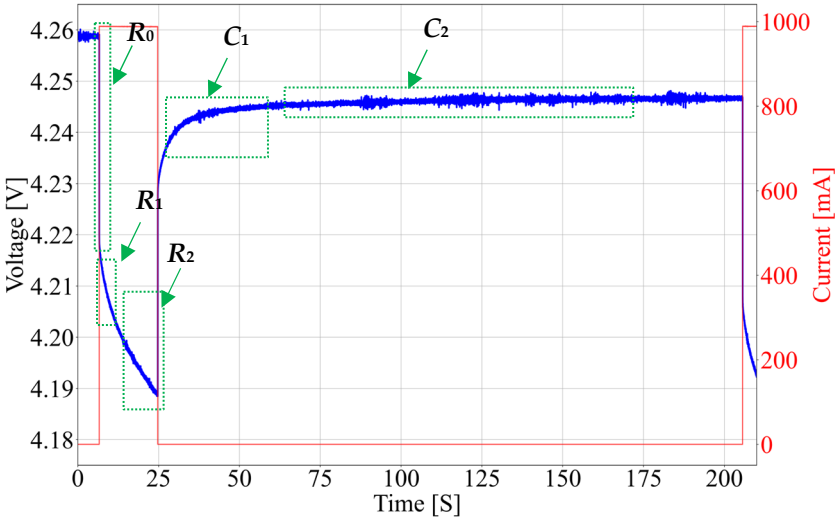


Figure 7. Discharge current pulses (red) and battery terminal voltage (blue) used to extract Rint and the 2nd-order battery model parameters.

The applied discharge profile produces periodical current pulses over the full discharge interval. To obtain the parameters with a resolution of 1% SoC, the current pulse length equals 18s, and the pulse amplitude equals 1 A (2C). The pause between discharge pulses should be at least 10 times longer than impulse length so the battery has enough time to relax.

When determining the values of model parameters, instead of using a single average value for the entire SoC range, the method [18] improves the accuracy of SoC estimation algorithms by using averaged values across several smaller SoC ranges. Rint as a function of SoC, for the simple Rint model, is given in Figure 8. The dashed red lines represent Rint averaged over SoC ranges, chosen to follow Rint changes and improve SoC estimation accuracy. The averaged values are presented in Table 7.

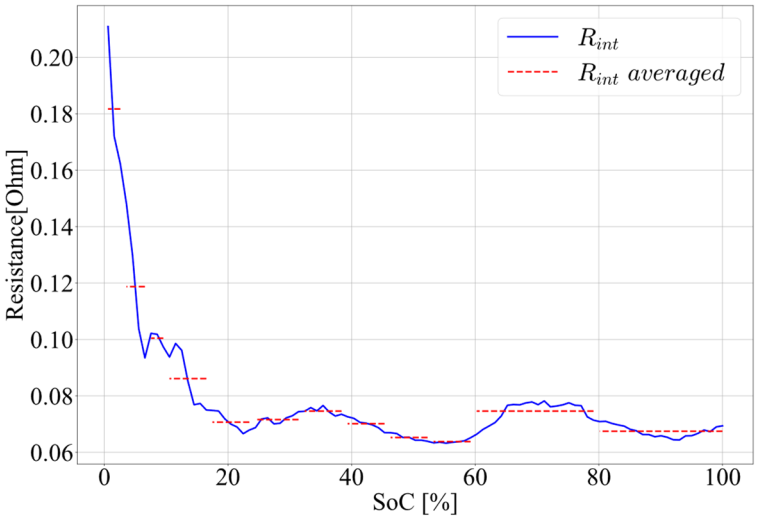


Figure 8. Rint as a function of SoC.

Table 7. Average Rint over chosen SoC ranges.

[% SoC]	**[100, 80)**	**[80, 60)**	**[60, 55)**	**[55, 45)**	**[45, 40)**	**[40, 32.5)**
R_{int} [mΩ]	67.48	74.61	63.81	65.23	70.15	74.61
[% SoC]	**[32.5, 25)**	**[25, 17.5)**	**[17.5, 10.5)**	**[10.5, 7.5)**	**[7.5, 3.5)**	**[3.5, 0)**
R_{int} [mΩ]	71.16	70.68	86.12	100.44	118.67	181.69

Averaging values across smaller SoC ranges improves algorithm accuracy, with the best results achieved using the smallest possible intervals. Maximum accuracy is obtained when the number of intervals matches the number of Rint calculation points. However, smaller intervals increase resource, including storage and computational time, usage. Therefore, it is important to balance accuracy and resource consumption. In this research, regions were chosen to enhance accuracy in the range 10–100% of SoC, as a battery is mostly used when SoC is within this range. The region limits for estimation for the Rint model, presented in Table 8, are also used as limits for 2nd-order RC model parameters. The 2nd-order RC model parameters as a function of SoC are given in Figure 9. The dashed lines present averaged parameters over SoC ranges, the same as in the case of the simple Rint model. The averaged values are presented in Table 8.

Table 8. Average 2nd-order parameters values over chosen SoC ranges.

[% SoC]	**[100, 80)**	**[80, 60)**	**[60, 55)**	**[55, 45)**	**[45, 40)**	**[40, 32.5)**
R_0 [mΩ]	41.46	40.90	41.87	43.03	45.02	47.37
R_1 [mΩ]	11.18	12.13	10.27	9.68	9.59	9.49
R_2 [mΩ]	14.82	21.56	11.65	12.47	15.53	17.7
C_1 [F]	215.70	198.56	233.69	247.92	250.34	253.04
C_2 [F]	2306.94	1608.13	2901.73	2709.37	2184.51	1906.07
[% SoC]	**[32.5, 25)**	**[25, 17.5)**	**[17.5, 10.5)**	**[10.5, 7.5)**	**[7.5, 3.5)**	**[3.5, 0)**
R_0 [mΩ]	49.55	52.46	56.69	61.77	69.97	96.66
R_1 [mΩ]	9.16	9.54	11.52	14.35	15.12	20.04
R_2 [mΩ]	12.85	8.67	17.92	24.37	33.58	64.96
C_1 [F]	261.90	252.17	209.74	167.53	159.87	119.91
C_2 [F]	2714.06	4022.56	2209.57	1385.49	1329.62	519.20

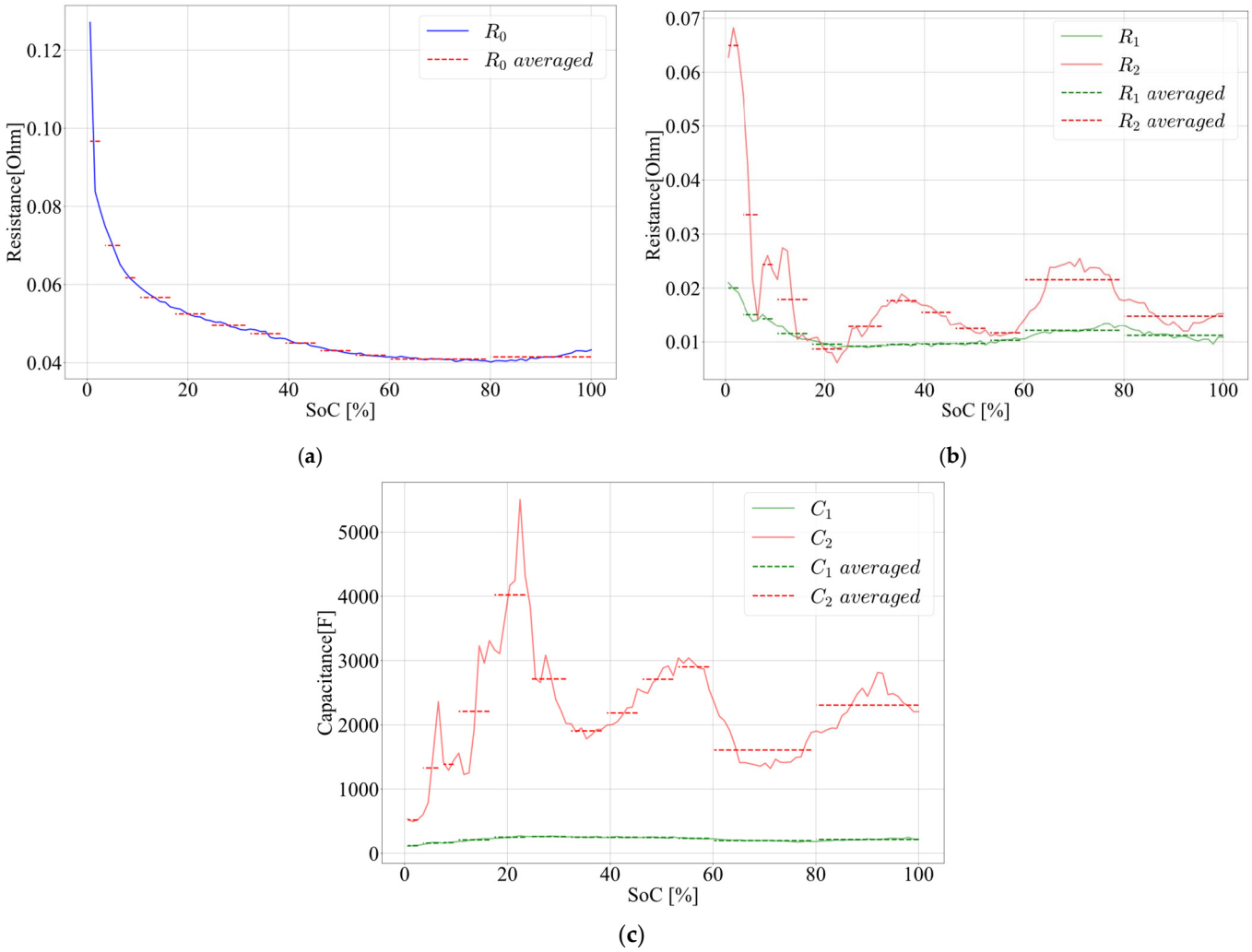


Figure 9. The 2nd-order RC model parameters as functions of SoC: (**a**)—R_0; (**b**)—R_1 and $\boldsymbol{R_2}$; (**c**)—C_1 and C_2.

6. Results

In this section, $RMSE(T)$ (9) obtained by the experiment has been checked against theoretical expectations in the first subsection. The selected algorithms' performance analysis has been performed in the second subsection. A use case has been provided in the third subsection.

6.1. Estimation Accuracy

Experimentally obtained values of $RMSE_{\mathrm{B}}$ for different discharge rates that mimic consumption of a system and two estimation algorithms are presented in Table 9, along with the relative yield of Kalman 2.

Table 9. Values of $RMSE_{\mathrm{B}}$ for Kalman 0 and Kalman 2 for different system consumption rates.

I_S	3C	2C	C	C/2	C/5	C/10
$\boldsymbol{RMSE_B(K_0)}$[%]	1.77	1.55	1.35	0.99	0.94	0.88
$\boldsymbol{RMSE_B(K_2)}$[%]	1.52	1.32	1.22	0.92	0.87	0.81
relative yield of $\boldsymbol{K_2}$[%]	14.1	14.8	9.6	7.1	7.4	8.0

For the selected discharge rates, the values of $RMSE_{\mathrm{B}}$ for Kalman 2 are lower than the values for Kalman 0, as expected since Kalman 2 incorporates a more complex battery

model. As discharge rate decreases, the Kalman 2 gets better, yielding from 15% to 7% $RMSE_{\mathrm{B}}$ improvement over Kalman 0.

Values of $RMSE_{\mathrm{T}}$ both calculated from (17) and normalized by $RMSE_{\mathrm{M}}$ (19), as well as experimentally obtained, for Kalman 0 and Kalman 2, are presented in Figure 10 in log–log scale.

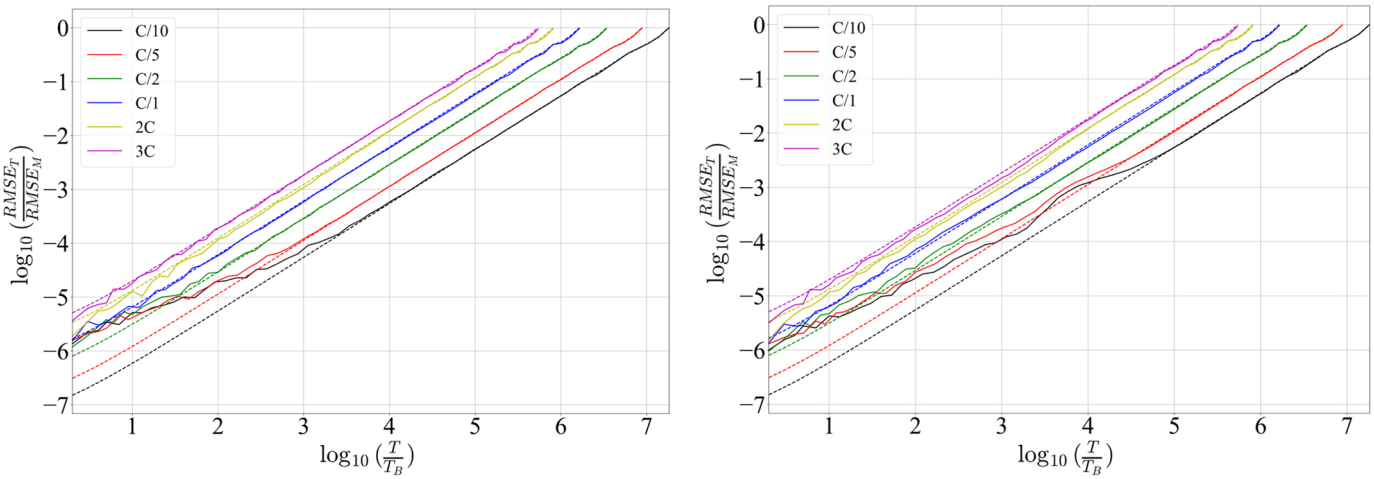


Figure 10. Normalized $\mathrm{RMSE}_{\mathrm{T}}$ values obtained by theory (dotted lines) and by experiment (full lines) for: (**a**) Kalman 0; (**b**) Kalman 2.

The values of $RMSE_{\mathrm{T}}$ closely match for higher estimation periods for both EKFs. At lower estimation periods, there is a difference between theory and the experiment, increasing as discharge rate decreases. The difference occurs because the theory assumes a perfect estimator, which is not the case. Moreover, the difference of $RMSE_{\mathrm{T}}$ for Kalman 2 is noticeable within a wider range of estimation periods when compared to Kalman 0. Consequently, $RMSE_{\mathrm{T}}$ (18) may not be applicable to small estimation periods. However, for both EKF methods, even within the range where $RMSE_{\mathrm{T}}$'s mismatch, the values of $RMSE_{\mathrm{T}}$ are at least 10 times smaller than the corresponding $RMSE_{\mathrm{B}}$ values shown in Table 9. Therefore, when considering the total $RMSE(T)$ (9), crucial for calculating the energy performance curve (26), the deviation of $RMSE_{\mathrm{T}}$ from theory, as well as $RMSE_{\mathrm{T}}$ value in general, is negligible compared to $RMSE_{\mathrm{B}}$ in the region of relatively low estimation periods. Additionally, within the region of small estimation periods, RAC values are higher, and they contribute more to the energy performance curve (26).

Figure 11 shows the differences between the total RMSE, $RMSE(T)$ defined by (9), where experimentally obtained values of $RMSE_{\mathrm{B}}$ given in Table 10 are used, and the experimentally obtained values for the total RMSE.

It can be seen in the figure that, for the selected range of discharge rates, the difference between theory and the experiment is about 1% for the highest discharge rate. However, for lower discharge rates, this difference is under 0.25%.

Table 10. Extended Kalman filter single run cycle execution time.

	Average Number of CPU Cycles	**Execution Time [uS]**
Kalman 0	35 636	89
Kalman 2	51 676	129

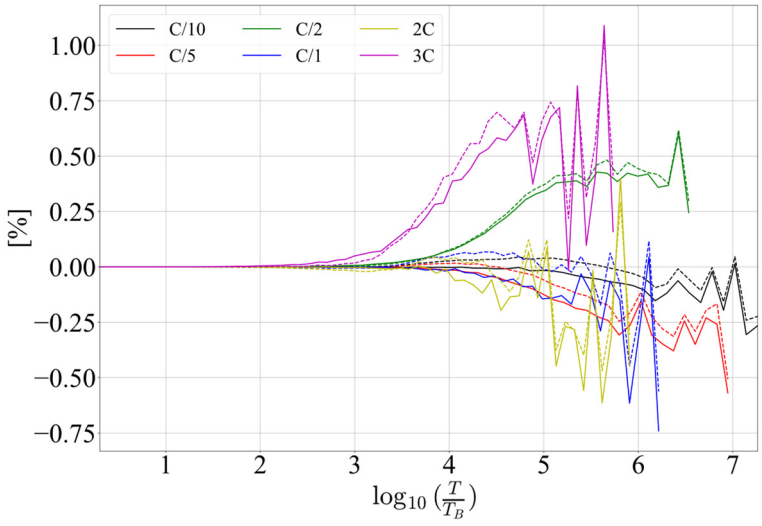


Figure 11. Difference between RMSE obtained by experiment and by theory, for Kalman 0 (full lines) and Kalman 2 (dotted lines).

6.2. Performance

To perform energy consumption analysis for an actual microcontroller-based system, both EKF algorithms are implemented in software and quantified in terms of time required to perform a single cycle run. The results achieved on the STM32H723 MCU configured as shown in Table 4, for the EKF algorithms based on Rint and the 2nd-order battery RC model, are presented in Table 10.

From the data presented in Table 11, Kalman 0 implementation is about 30% faster than the Kalman 2 implementation. The information presented inside the table, combined with the microcontroller properties presented in Table 4, is sufficient to determine the algorithm's energy consumption when EKF is running as the sole software component on the microcontroller. Therefore, one can conclude that Kalman 2 consumes 30% more energy when compared to Kalman 0.

Table 11. Optimal estimation periods, and RMSE and RAC values at base and optimal estimation periods, for Kalman 0 and Kalman 2, and for different discharge rates.

Kalman	I_S	C/5	C/10	C/25	C/50	C/100	C/200
Kalman 0	$\boldsymbol{RMSE_B}$[%]	0.870	0.810	0.750	0.690	0.630	0.570
	$\boldsymbol{RAC_{T_B}}$[%]	3.546	6.849	15.527	26.880	42.371	59.522
	$\boldsymbol{T_0}$[s]	7.435	14.410	35.345	68.159	132.932	256.848
	$\boldsymbol{RAC_{T_0}}$[%]	0.001076	0.001110	0.001132	0.001174	0.001204	0.001246
	$\boldsymbol{RMSE(T_0)}$[%]	0.870314	0.810317	0.750329	0.690333	0.630347	0.570358
Kalman 2	$\boldsymbol{RMSE_B}$[%]	0.810	0.770	0.700	0.640	0.590	0.530
	$\boldsymbol{RAC_{T_B}}$[%]	5.882	11.110	23.808	38.459	55.553	71.427
	$\boldsymbol{T_0}$[s]	8.583	17.020	41.398	79.601	154.694	300.558
	$\boldsymbol{RAC_{T_0}}$[%]	0.001584	0.001598	0.001643	0.001708	0.001758	0.001810
	$\boldsymbol{RMSE(T_0)}$[%]	0.810450	0.770465	0.700484	0.640489	0.590501	0.530526

Once the execution times of the EKF algorithms are obtained, RAC (24) can be determined for both EKFs and the discharge profiles listed in Table 10. The performance curves (26) for two estimators can be compared when combining RAC and $RMSE(T)$ obtained theoretically and experimentally. Figure 12 presents these theoretical and experimental performance curves, EP_{T} and EP_{M} respectively, for the OPT, criterion where $k_{\mathrm{E}} = k_{\mathrm{R}} = 0.5$, for both Kalman 0 and Kalman 2.

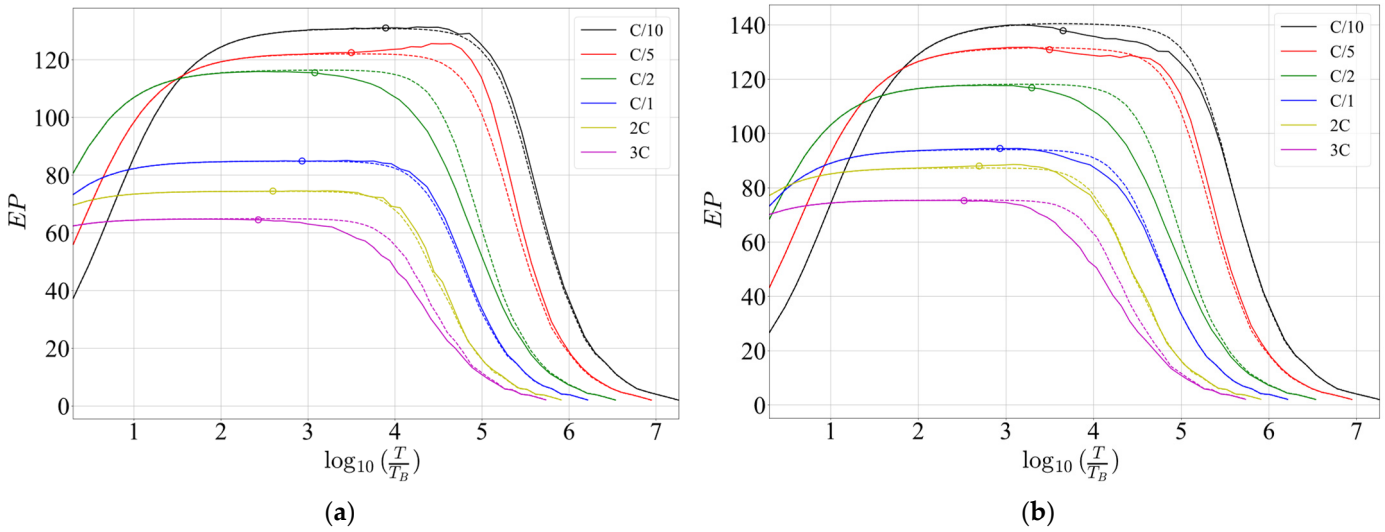


Figure 12. Estimator performance curves obtained theoretically (dotted lines) and experimentally (full lines) for various discharge rates, for: (**a**) Kalman 0; (**b**) Kalman 2.

The optimal sampling period T_0 for a chosen discharge rate and estimation algorithm can be determined from Figure 12. In the figure, the circles on the EP_{M} curves represent the estimation periods for which EP_{T} has a maximum. It is evident that the maximums, obtained the experiment, agree with the theoretical expectations. Based on Figure 12, it can be inferred that it is feasible to use the theoretical analysis presented in the Section 3 to predict the estimator performance for any discharge rate if the prior information about corresponding $RMSE_{\mathrm{B}}$ is known.

6.3. Use Case

At lower discharge rates, comparable to the consumption of an SoC algorithm, the value of relative algorithm consumption RAC_{T} is higher. In this case, selection of the optimal estimation period for the SoC algorithm results in greater energy savings compared to higher discharge rates. While performance analysis at low discharge rates is important, conducting such analysis experimentally is impractical. Although sophisticated hardware can address the technical challenges of measuring low currents, logging and analyzing voltage and current samples require significant computational and storage resources. Therefore, in such a case, a theoretical analysis can be used for determination of the optimal SoC algorithm estimation period.

To demonstrate the potential of the presented analysis, two possible real-world system design scenarios are analyzed:

1. optimal estimation period should be identified in order to minimize energy consumption and estimation accuracy error.
2. optimal estimation algorithm should be selected for a selected estimation period and selected optimization criterion.

Both design scenarios consider two EKF implementations, Kalman 2 and Kalman 0, as SoC estimation algorithms that should be implemented as software solutions on battery-powered systems based on the STM32H723 microcontroller. These algorithms have already been implemented on this microcontroller, and the obtained single-run execution times are presented in Table 10. Also, both algorithms have been evaluated with BES under discharge rates of C/5 and C/10, and the obtained $\mathrm{RMSE_B}$ values are presented in Table 9. The analyzed system operates under discrete static low discharge profiles (C/5, C/10, C/25, C/50, C/100, and C/200), which are comparable to microcontroller consumption defined in Table 4.

For the first design scenario, relative algorithm consumption RAC_T is calculated from (24) substituting the values presented in Tables 4 and 10. To enable calculation of $RMSE(T)$, the following assumptions are introduced:

- $RMSE_B$ is reduced by approximately 0.07% when discharge rate halves as can be concluded from Table 9.
- $RMSE_B$ of Kalman 2 is about 7% better than $RMSE_B$ of Kalman 0, as noted earlier.

The estimator performance curves for discharge rates of C/5, C/10, C/25, C/50, C/100, and C/200, for Kalman 0 and Kalman 2, and the optimization criterion OPT, are presented in Figure 13. Table 11 lists the optimal period, and RMSE and RAC for base and optimal estimation periods for both implemented estimators and discharge rates.

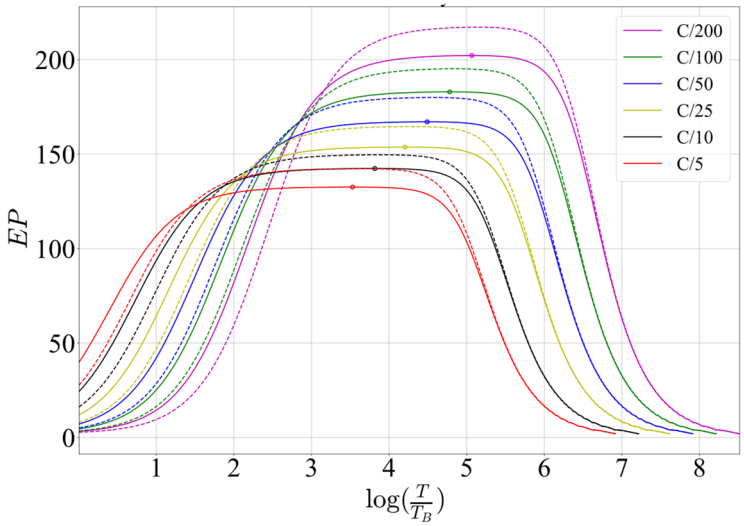


Figure 13. Theoretically obtained estimator performance curves for OPT criterion, for different discharge rates for Kalman 0 (full lines) and Kalman 2 (dotted lines).

Based on the values of RAC_{T_0} and RAC_{T_B} from Table 11, it can be noticed that for the discharge rate of C/200 and EKF based on the simple Rint model, the selection of the optimal estimation period leads to a reduction in the algorithm energy consumption of approximately 71%. In the case of the battery used in the experiment, this reduction amounts to about 355 mAh of capacity, which can be used by other system activities or simply extend the system operation time. The energy reduction is accompanied by a negligible increase in the SoC estimation error, since at the optimal estimation period $RMSE_B$ originating from the estimator imperfectness is much higher than $RMSE_{T_0}$ originating from the increase of the estimation period.

For 2. design scenario, the two estimators are compared in terms of performance using the performance ratio (27). Figure 14a depicts the dependence of the ratio on estimation period for the optimization criterion OPT, Kalman 0 and Kalman 2, and the discharge rate of C/200. On this figure, three ranges of estimation periods are clearly identifiable. In range 1, Kalman 0 performs better, in range 2, Kalman 2 performs better, and in range 3, the estimators' performance is equal. Based on this notion, if the estimation period of an estimator is pre-set, the estimator that performs better can be selected for actual implementation. It is of course possible to employ different optimization criteria and get respective performance ratios. Figure 14b illustrates estimator performance ratios for the same system for $\mathbf{OPT_A}$ and $\mathbf{OPT_E}$ optimization criteria. The figure clearly shows that for a selected estimation period, the estimators have different performances for different optimization criteria. For example, if $T = T_B \times 10^{2.7} = 1$ s, Kalman 2 provides better performance under the $\mathbf{OPT_A}$ criterion, which prioritizes accuracy over consumption. On the other hand, when consumption is the priority, as per the $\mathbf{OPT_E}$ criterion, Kalman 0 achieves better overall performance.

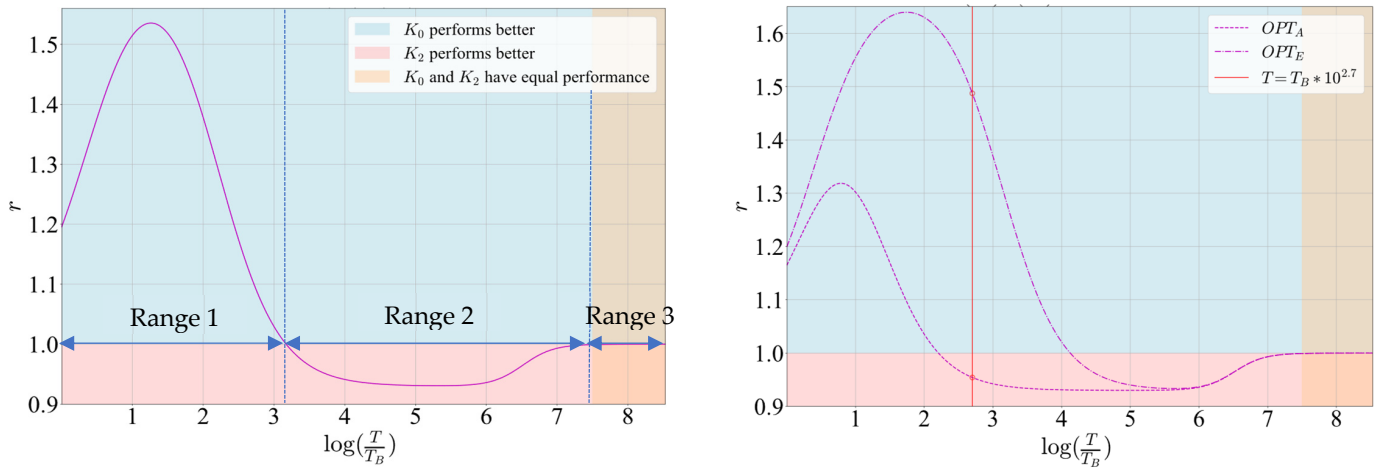


Figure 14. Performance ratio $r(T)$ for Kalman 0 and Kalman 2, for discharge rate C/200 and: (**a**) OPT optimization criterion; (**b**) OPT_A and OPT_E optimization criteria.

7. Conclusions

Selecting an SoC algorithm for battery-powered microcontroller-based embedded platforms is challenging due to the lack of information in the literature regarding algorithm energy consumption. This paper conducts a detailed theoretical analysis to lay the foundation for quantifying the energy consumption of recursive model-based SoC algorithms, which are suitable for implementation on resource-constrained embedded platforms. Consequently, a set of equations was derived, and the estimation period was identified as a configurable parameter that can be adjusted to reduce the algorithms' energy consumption. To link accuracy and energy consumption, the paper proposes an estimator performance equation as a function of the estimation rate, enabling the selection of the optimal estimation rate. Results obtained within this paper demonstrate that there is an acceptable deviation between values obtained with derived equations and values obtained by experiments. Therefore, the theoretical approach may provide acceptable results and reduce experimentation time.

Two types of extended Kalman filters, utilizing models of different complexities, are implemented to evaluate their relative computational complexity per single run cycle on a real embedded platform. It has been obtained that the EKF based on the 2nd-order RC model is 30% slower compared to the EKF based on the simple Rint model in the case of implementation on an STM32H7-based microcontroller. These results are used to demonstrate the potential that the equation derived in this work has in selecting optimal estimation periods and designing low-consumption SoC algorithms for resource-constrained embedded platforms. It has been shown that choosing the optimal sampling period can significantly reduce energy consumption with a slight degradation of SoC estimation accuracy for systems where total system currents are comparable to the current of the microcontroller running the algorithm. For systems with current discharge rates ranging from C/5 to C/200, energy reduction can reach up to 71%. Besides selecting the optimal estimation period, the derived set of equations can also be used to determine which algorithm performs better for the selected estimation period.

It has been demonstrated that, for different SoC algorithms, there are different optimal estimation periods for a selected optimization criterium. This paper establishes a foundation for future work, which may involve the use of different quantifications of accuracy, different optimization strategies, and the development of an algorithm that alternates between different estimators and/or adapts estimation periods based on consumption dynamics. Also, the experimental verification of the results could be extended to other battery load profiles.

Author Contributions: Conceptualization, H.T. and I.P.; methodology, H.T. and V.R.; software, H.T.; validation, H.T., V.R. and I.P.; formal analysis, I.P. and V.R.; investigation, H.T.; resources, H.T.; data curation, H.T.; writing—original draft preparation, H.T.; writing—review and editing, V.R.; visualization, H.T.; supervision, I.P. and V.R.; project administration, I.P.; funding acquisition, V.R. All authors have read and agreed to the published version of the manuscript.

Funding: This work is funded by the Austrian Federal Ministry of Climate Action, Environment, Energy, Mobility, Innovation and Technology, the Austrian Federal Ministry of Digital and Economic Affairs, and implemented by Austria wirtschaftsservice (aws) and the Austrian Research Promotion Agency (FFG) in the frame of the Important Project of Common European Interest (IPCEI) on Microelectronics and Communication Technologies.

Data Availability Statement: Data are contained within the article.

Acknowledgments: This work was financially supported by the Ministry of Science, Technological Development and Innovation of the Republic of Serbia under contract number: 451-03-65/2024-03/200103.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

EKF algorithm state equations in discrete time domain are:

$$x[k] = Ax[k-1] + Bu[k-1] + \omega[k-1] \tag{A1}$$

$$y[k] = Cx[k] + Du[k] + v[k] \tag{A2}$$

where k represents the ordinal number of the sample point, x is state variable, u is control input, y is measurement, whereas ω and v represent system and measurement zero mean Gaussian white noise with covariance matrices Q and R, respectively. When Equations (A1) and (A2) are applied for SoC estimation, $u[k]$ and $y[k]$ represent battery current and terminal voltage at the sample point k, respectively. The size of state variables, as well as of the matrices A, B, C, and D, depend on the employed battery electrical model.

A battery model is important to accurately represent the electrical behavior of the battery. Nevertheless, it is important to find an appropriate balance between model complexity and accuracy so that the algorithm can be implemented on a computationally constrained embedded platform. To analyze the impact that a model has on accuracy and system energy consumption, this research utilizes two battery models: the Rint model, computationally simpler but less accurate, and the 2nd-order RC model, computationally complex but of higher accuracy. The two models are presented in Figure A1.

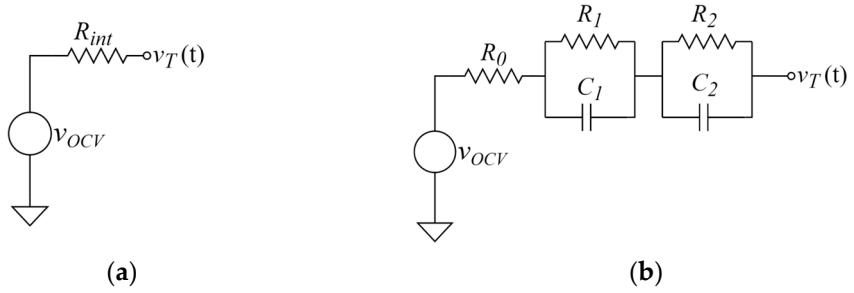


Figure A1. Battery electrical circuit models: (**a**) Rint model; (**b**) the 2nd-order RC model.

The Rint model consists of a voltage generator representing the open circuit voltage (OCV) [48] and a series resistor representing the battery internal ohmic resistance. This model is simple, and it uses comparably less computational power. However, it does not provide sufficient accuracy in some applications.

The 2nd-order RC model adds two parallel RC networks in series to the internal ohmic resistance. This model represents an upgraded Thevenin model that employs a single

RC network. The two RC networks are added to more accurately model the polarization effect of the battery. The first RC network (R_1 and C_1) simulates a fast battery process, whereas the second network (R_2 and C_2) simulates a slow battery process. Consequently, this model better describes the transient and steady-state characteristics of the processes inside a battery. Notwithstanding, although this model offers a higher accuracy, it requires more computational power.

The content of matrices *A*, *B*, *C*, and *D* for the two models is presented in Table A1.

Table A1. Kalman filter matrices.

Matrices	Model	
	Rint	**2nd-Order Thevenin Model**
A	$[1]$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & e^{-\frac{T_s}{C_1 R_1}} & 0 \\ 0 & 0 & e^{-\frac{T_s}{C_2 R_2}} \end{bmatrix}$
B	$\left[-\frac{T_S}{C_{NOM}}\right]$	$\begin{bmatrix} -\frac{T_S}{C_{NOM}} \\ R_1\left(1-e^{-\frac{T_s}{C_{FAST}R_{FAST}}}\right) \\ R_{SLOW}\left(1-e^{-\frac{T_s}{C_{SLOW}R_{SLOW}}}\right) \end{bmatrix}$
C	$\left[\frac{dv_{OC}(SOC)}{dSOC}\right]$	$\left[\frac{dv_{OC}(SOC)}{dSOC} \quad -1 \quad -1\right]$
D	$[R_0]$	$[R_0]$

Appendix B

For a proper evaluation of battery characteristics, an evaluation system that closely matches operating conditions is needed. A low-cost in-house solution has been developed for this purpose, called the battery evaluation system (BES). When compared to traditional equipment used for evaluating battery characteristics, which is often expensive, locked in hardware and software modifications, and unsuitable for analyzing in the domain of relatively low currents, the developed system enables great flexibility and serves as a robust solution for current and future experiments. The system provides for configurable voltage and current sampling rates, streaming over Ethernet, generation of arbitrary consumption profiles, Python scripts for extraction of battery parameters, a user-friendly graphical interface, and other useful features. The complete measurement system software libraries and hardware schematics are available as an open-source solution on the GitHub repository [62].

The BES block diagram is presented in Figure A2. The BES hardware consists of a power board, control board, and MCU board.

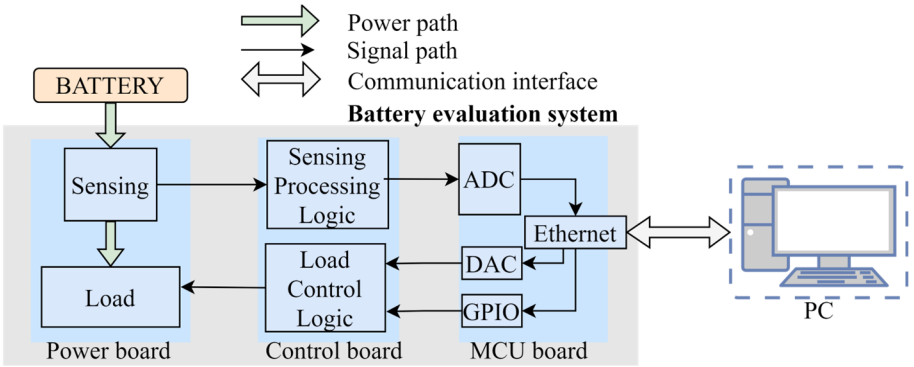


Figure A2. Battery evaluation system block diagram.

The power board enables current measurement and generation of arbitrary consumption profiles. The present version of the board supports current measurement in the range of 5A. The control board integrates analog components in charge of monitoring and control

of the power board. The functions of the board include scaling of the current sense and battery voltages, overvoltage and overcurrent detection, etc. The MCU board used is the official STM32H7-DISCO development board employing the STM32H747XIH6 microcontroller. The microcontroller is selected since it contains AD converters with up to 5 MSps configurable sampling rate. Moreover, this microcontroller includes an Ethernet peripheral capable of establishing a 100 Mbps communication link. Since the BES is implemented as a set of boards, it is possible to modularly change the boards and adapt the system for various experiments and analyses.

The hardware of the BES is presented in Figure A3.

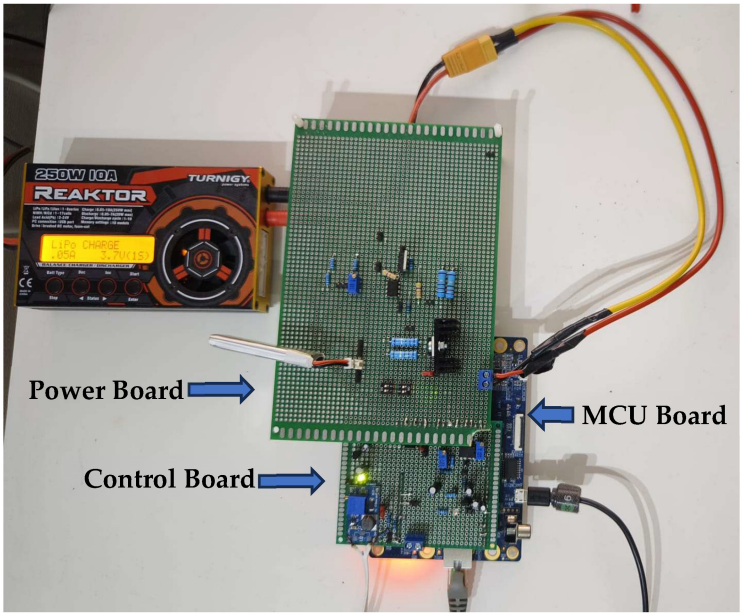


Figure A3. The hardware of the battery evaluation system.

The BES software (version 1.0) architecture is built based on general design guidelines described in [63], and it includes a firmware based on RTOS implemented on MCU, and C++ and Python-based software components implemented on a PC.

The microcontroller firmware is based on the FreeRTOS real-time operating system, which utilizes the LwIP software (https://savannah.nongnu.org/projects/lwip/, accessed on 10 September 2024) communication stack to enable TCP/IP communication. The main firmware blocks, as well as their connections, are illustrated in Figure A4.

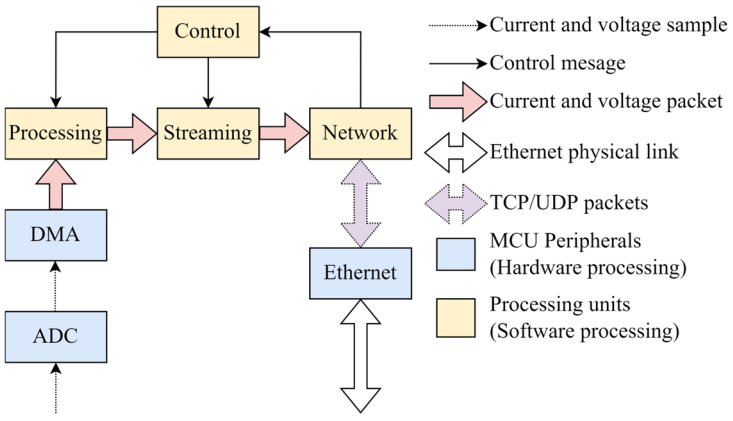


Figure A4. The BES firmware blocks and interconnections.

Battery voltage and current are sampled by the AD converter integrated within the microcontroller. The DMA controller, synchronized with the AD converter, combines

multiple voltage and current samples into a single sample packet. This packet is processed by the processing software block, which encapsulates the sample packet with the corresponding header and forwards it to the streaming block. The streaming block performs packet enumeration and forwards the packet to the network block, which encapsulates the received packet within a UDP message and transfers it to a PC. The system can be remotely controlled by the PC using TCP control messages. The TCP messages are received by the network block, which forwards them to the control block. The control block parses the messages and performs the corresponding action based on the payload.

Software on the PC is used to configure hardware easily, but also to receive, log, and process voltage and current samples. For the sake of efficiency, these activities are implemented within two different parts: the graphical user interface (GUI) application and various Python scripts.

The GUI application is written in a C++-based Qt framework, and it enables the control of the microcontroller, as well as real-time reception and plotting of data samples. Because C++ is very efficient in the processing of raw data, the application can receive data with a minimum CPU utilization [64]. Also, this application provides for logging of data samples into a file, an important feature for further offline data analysis.

Due to the availability of diverse third-party libraries that enable easy data processing, the implementation of various battery-related data analysis algorithms was much simpler in Python when compared to the C++ Qt framework. When data are stored inside a file, it is analyzed by use of corresponding Python scripts. For example, one script implements an algorithm for the extraction of open circuit voltage battery parameters; another extracts values of resistors and capacitors in a model; whereas a third script incorporates the EKF filter algorithm.

References

1. Kalawoun, J.; Biletska, K.; Suard, F.; Montaru, M. From a Novel Classification of the Battery State of Charge Estimators Toward a Conception of an Ideal One. *J. Power Sources* **2015**, *279*, 694–706. [CrossRef]
2. Surendar, M.; Pradeepa, P. Future Challenges in State of Charge Estimation for Lithium-Ion Batteries. *Int. J. Eng. Adv. Technol.* **2020**, *10*, 215–223. [CrossRef]
3. Hannan, M.; Lipu, M.; Hussain, A.; Mohamed, A. A Review of Lithium-Ion Battery State of Charge Estimation and Management System in Electric Vehicle Applications: Challenges and Recommendations. *Renew. Sustain. Energy Rev.* **2017**, *78*, 834–854. [CrossRef]
4. Wang, Y.; Tian, J.; Sun, Z.; Wang, L.; Xu, R.; Li, M.; Chen, Z. A Comprehensive Review of Battery Modelling and State Estimation Approaches for Advanced Battery Management Systems. *Renew. Sustain. Energy Rev.* **2020**, *131*, 110015. [CrossRef]
5. Yang, B.; Wang, J.; Cao, P.; Zhu, T.; Shu, H.; Chen, J.; Zhang, J.; Zhu, J. Classification, Summarization and Perspectives on State-of-Charge Estimation of Lithium-Ion Batteries Used in Electric Vehicles: A Critical Comprehensive Survey. *J. Energy Storage* **2021**, *39*, 102572. [CrossRef]
6. Fu, Y.; Zhai, B.; Shi, Z.; Liang, J.; Peng, Z. State of Charge Estimation of Lithium-Ion Batteries Based on an Adaptive Iterative Extended Kalman Filter for AUVs. *Sensors* **2022**, *22*, 9277. [CrossRef]
7. Hossain, M.; Haque, M.E.; Arif, M.T. Online Model Parameter and State of Charge Estimation of Li-Ion Battery Using Unscented Kalman Filter Considering Effects of Temperatures and C-Rates. *IEEE Trans. Energy Convers.* **2022**, *37*, 2498–2511. [CrossRef]
8. Liu, X.; Fan, X.; Wang, L.; Wu, J. State of Charge Estimation for Power Battery Base on Improved Particle Filter. *World Electr. Veh. J.* **2023**, *14*, 8. [CrossRef]
9. Liu, X.; Wang, L.; Wu, J. Online Identification of Power Battery Parameters for Electric Vehicles Using a Decoupling Multiple Forgetting Factors Recursive Least Squares Method. *CSEE J. Power Energy Syst.* **2020**, *6*, 735–742.
10. He, W.; Williard, N.; Chen, C.; Pecht, M. State of Charge Estimation for Electric Vehicle Batteries Using Unscented Kalman Filtering. *Microelectron. Reliab.* **2013**, *53*, 840–847. [CrossRef]
11. Xia, B.; Sun, Z.; Zhang, R.; Lao, Z. A Cubature Particle Filter Algorithm to Estimate the State of the Charge of Lithium-Ion Batteries Based on a Second-Order Equivalent Circuit Model. *Energies* **2017**, *10*, 457. [CrossRef]
12. Movassagh, K.; Raihan, A.; Balasingam, B.; Pattipati, K. A Critical Look at Coulomb Counting Approach for State of Charge Estimation in Batteries. *Energies* **2021**, *14*, 4074. [CrossRef]
13. Gismero, E.; Schaltz, E.; Stroe, D. Recursive State of Charge and State of Health Estimation Method for Lithium-Ion Batteries Based on Coulomb Counting and Open Circuit Voltage. *Energies* **2020**, *13*, 1811. [CrossRef]
14. Wang, X.; Gong, R.; Yang, Z.; Kang, L. State of Charge Estimation of Lithium-Ion Batteries Based on Online OCV Curve Construction. *Batteries* **2024**, *10*, 208. [CrossRef]

15. Saji, D.; Babu, P.S.; Ilango, K. SoC Estimation of Lithium-Ion Battery Using Combined Coulomb Counting and Fuzzy Logic Method. In Proceedings of the 2019 4th International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT), Bangalore, India, 17–18 May 2019.
16. Shan, C.; Chin, C.S.; Mohan, V.; Zhang, C. Review of Various Machine Learning Approaches for Predicting Parameters of Lithium-Ion Batteries in Electric Vehicles. *Batteries* **2024**, *10*, 181. [CrossRef]
17. Dao, V.Q.; Dinh, M.-C.; Kim, C.S.; Park, M.; Doh, C.-H.; Bae, J.H.; Lee, M.-K.; Liu, J.; Bai, Z. Design of an Effective State of Charge Estimation Method for a Lithium-Ion Battery Pack Using Extended Kalman Filter and Artificial Neural Network. *Energies* **2021**, *14*, 2634. [CrossRef]
18. Yun, J.; Choi, Y.; Lee, J.; Choi, S.; Shin, C. State-of-Charge Estimation Method for Lithium-Ion Batteries Using Extended Kalman Filter with Adaptive Battery Parameters. *IEEE Access* **2023**, *11*, 90901–90915. [CrossRef]
19. Kung, C.-C.; Luo, S.-X.; Liu, S.-H. Implementation of the State of Charge Estimation with Adaptive Extended Kalman Filter for Lithium-Ion Batteries by Arduino. In Proceedings of the International Conference on System Science and Engineering (ICSSE), New Taipei, Taiwan, 28–30 June 2018; pp. 1–6.
20. Nejad, S.; Gladwin, D.T.; Stone, D.A. On-Chip Implementation of Extended Kalman Filter for Adaptive Battery States Monitoring. In Proceedings of the IECON 2016—42nd Annual Conference of the IEEE Industrial Electronics Society, Florence, Italy, 24–27 October 2016; pp. 5513–5518.
21. Hong, S.; Kang, M.; Park, H.; Kim, J.; Baek, J. Real-Time State-of-Charge Estimation Using an Embedded Board for Li-Ion Batteries. *Electronics* **2010**, *11*, 2022. [CrossRef]
22. Ma, Y.; Duan, P.; He, P.; Zhang, F.; Chen, H. FPGA Implementation of Extended Kalman Filter for SoC Estimation of Lithium-Ion Battery in Electric Vehicle. *Asian J. Control* **2019**, *21*, 2126–2136. [CrossRef]
23. Guo, C.; Ci, S.; Zhou, Y.; Yang, Y. A Survey of Energy Consumption Measurement in Embedded Systems. *IEEE Access* **2021**, *9*, 60516–60530. [CrossRef]
24. Callebaut, G.; Leenders, G.; Van Mulders, J.; Ottoy, G.; De Strycker, L.; Van der Perre, L. The Art of Designing Remote IoT Devices—Technologies and Strategies for a Long Battery Life. *Sensors* **2021**, *21*, 913. [CrossRef] [PubMed]
25. Quintero, V.; Estevez, C.; Orchard, M. State-of-Charge Estimation to Improve Energy Conservation and Extend Battery Life of Wireless Sensor Network Nodes. In Proceedings of the 2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN), Milan, Italy, 4–7 July 2017; pp. 153–158.
26. Ali, M.U.; Zafar, A.; Nengroo, S.H.; Hussain, S.; Junaid Alvi, M.; Kim, H.-J. Towards a Smarter Battery Management System for Electric Vehicle Applications: A Critical Review of Lithium-Ion Battery State of Charge Estimation. *Energies* **2019**, *12*, 446. [CrossRef]
27. Ciortea, F.; Rusu, C.; Nemes, M.; Gatea, C. Extended Kalman Filter for State-of-Charge Estimation in Electric Vehicles Battery Packs. In Proceedings of the 2017 International Conference on Optimization of Electrical and Electronic Equipment (OPTIM) & 2017 Intl Aegean Conference on Electrical Machines and Power Electronics (ACEMP), Brasov, Romania, 25–27 May 2017; pp. 611–616.
28. Tulsyan, Y.; Tsai, R.B.; Gopaluni, S.; Braatz, R.D. State-of-Charge Estimation in Lithium-Ion Batteries: A Particle Filter Approach. *J. Power Sources* **2016**, *331*, 208–223. [CrossRef]
29. Zhu, F.; Fu, J. A Novel State-of-Health Estimation for Lithium-Ion Battery via Unscented Kalman Filter and Improved Unscented Particle Filter. *IEEE Sens. J.* **2021**, *21*, 25449–25456. [CrossRef]
30. Bian, X.; Wei, Z.; He, J.; Yan, F.; Liu, L. A Two-Step Parameter Optimization Method for Low-Order Model-Based State-of-Charge Estimation. *IEEE Trans. Transp. Electrif.* **2021**, *7*, 399–409. [CrossRef]
31. Shrivastava, P.; Soon, T.K.; Idris, M.Y.I.B.; Mekhilef, S. Overview of Model-Based Online State-of-Charge Estimation Using Kalman Filter Family for Lithium-Ion Batteries. *Renew. Sustain. Energy Rev.* **2019**, *113*, 109233. [CrossRef]
32. Antonucci, V.; Artale, G.; Brunaccini, G.; Caravello, G.; Cataliotti, A.; Cosentino, V.; Di Cara, D.; Ferraro, M.; Guaiana, S.; Panzavecchia, N.; et al. Li-Ion Battery Modeling and State of Charge Estimation Method Including the Hysteresis Effect. *Electronics* **2019**, *8*, 1324. [CrossRef]
33. Wang, T.; Chen, S.; Ren, H.; Zhao, Y. Model-Based Unscented Kalman Filter Observer Design for Lithium-Ion Battery State of Charge Estimation. *Int. J. Energy Res.* **2017**, *42*, 1603–1614. [CrossRef]
34. Barcellona, S.; Piegari, L. Lithium-Ion Battery Models and Parameter Identification Techniques. *Energies* **2017**, *10*, 2007. [CrossRef]
35. Zhang, C.; Li, K.; Pei, L.; Zhu, C. An Integrated Approach for Real-Time Model-Based State-of-Charge Estimation of Lithium-Ion Batteries. *J. Power Sources* **2015**, *283*, 24–36. [CrossRef]
36. He, W.; Pecht, M.; Flynn, D.; Dinmohammadi, F. A Physics-Based Electrochemical Model for Lithium-Ion Battery State-of-Charge Estimation Solved by an Optimised Projection-Based Method and Moving-Window Filtering. *Energies* **2018**, *11*, 2120. [CrossRef]
37. Madsen, K.; Perera, D.G. Toward Composing Efficient FPGA-Based Hardware Accelerators for Physics-Based Model Predictive Control Smart Sensor for HEV Battery Cell Management. *IEEE Access* **2023**, *11*, 106141–106171. [CrossRef]
38. Sibatov, R.T.; Svetukhin, V.V.; Kitsyuk, E.P.; Pavlov, A.A. Fractional Differential Generalization of the Single Particle Model of a Lithium-Ion Cell. *Electronics* **2019**, *8*, 650. [CrossRef]
39. Zhang, L.; Peng, H.; Ning, Z.; Mu, Z.; Sun, C. Comparative Research on RC Equivalent Circuit Models for Lithium-Ion Batteries of Electric Vehicles. *Appl. Sci.* **2017**, *7*, 1002. [CrossRef]

40. Madani, S.S.; Schaltz, E.; Kær, S.K. A Review of Different Electric Equivalent Circuit Models and Parameter Identification Methods of Lithium-Ion Batteries. *ECS Trans.* **2018**, *87*, 23–37. [CrossRef]
41. Jokar, B.; Rajabloo, M.; Désilets, M.; Lacroix, M. Review of Simplified Pseudo-Two-Dimensional Models of Lithium-Ion Batteries. *J. Power Sources* **2016**, *327*, 44–55. [CrossRef]
42. Zhang, D.; Dey, S.; Couto, L.D.; Moura, S.J. Battery Adaptive Observer for a Single-Particle Model with Intercalation-Induced Stress. *IEEE Trans. Control Syst. Technol.* **2020**, *28*, 1363–1377. [CrossRef]
43. Ren, Z.; Du, C.; Wu, Z.; Shao, J.; Deng, W. A Comparative Study of the Influence of Different Open Circuit Voltage Tests on Model-Based State of Charge Estimation for Lithium-Ion Batteries. *Int. J. Energy Res.* **2021**, *45*, 13692–13711. [CrossRef]
44. Noelle, D.J.; Wang, M.; Le, A.V.; Shi, Y.; Qiao, Y. Internal Resistance and Polarization Dynamics of Lithium-Ion Batteries Upon Internal Shorting. *Appl. Energy* **2018**, *212*, 796–808. [CrossRef]
45. Xu, Y.; Hu, M.; Fu, C.; Cao, K.; Su, Z.; Yang, Z. State of Charge Estimation for Lithium-Ion Batteries Based on Temperature-Dependent Second-Order RC Model. *Electronics* **2019**, *8*, 1012. [CrossRef]
46. Liu, X.; Li, W.; Zhou, A. PNGV Equivalent Circuit Model and SoC Estimation Algorithm for Lithium Battery Pack Adopted in AGV Vehicle. *IEEE Access* **2018**, *6*, 23639–23647. [CrossRef]
47. Miao, H.; Chen, J.; Mao, L.; Qu, K.; Zhao, J.; Zhu, Y. A Novel Online Model Parameters Identification Method with Anti-Interference Characteristics for Lithium-Ion Batteries. *Int. J. Energy Res.* **2021**, *45*, 9502–9517. [CrossRef]
48. Barcellona, S.; Codecasa, L.; Colnago, S.; Piegari, L. Open-Circuit Voltage Variation in LiCoO2 Battery Cycled in Different States of Charge Regions. *Energies* **2024**, *17*, 2364. [CrossRef]
49. Zhang, W.; Wang, L.; Wang, L.; Liao, C.; Zhang, Y. Joint State-of-Charge and State-of-Available-Power Estimation Based on the Online Parameter Identification of Lithium-Ion Battery Model. *IEEE Trans. Ind. Electron.* **2022**, *69*, 3677–3688. [CrossRef]
50. Zhang, S.; Zhang, X. A Comparative Study of Different Online Model Parameters Identification Methods for Lithium-Ion Battery. *Sci. China Technol. Sci.* **2021**, *64*, 2312–2327. [CrossRef]
51. Yang, K.; Tang, Y.; Zhang, Z. Parameter Identification and State-of-Charge Estimation for Lithium-Ion Batteries Using Separated Time Scales and Extended Kalman Filter. *Energies* **2021**, *14*, 1054. [CrossRef]
52. Chung, D.W.; Yang, S.H. SoC Estimation of Lithium-Ion Battery Based on Kalman Filter Algorithm for Energy Storage System in Microgrids. *E3S Web Conf.* **2019**, *57*, 6. [CrossRef]
53. Stroe, J.; Meng, D.; Stroe, M.; Świerczyński, M.; Teodorescu, R.; Kær, S.K. Influence of Battery Parametric Uncertainties on the State-of-Charge Estimation of Lithium Titanate Oxide-Based Batteries. *Energies* **2018**, *11*, 795. [CrossRef]
54. Meng, J. An Overview and Comparison of Online Implementable SoC Estimation Methods for Lithium-Ion Battery. *IEEE Trans. Ind. Appl.* **2018**, *54*, 1583–1591. [CrossRef]
55. He, Z.; Yang, Z.; Cui, X.; Li, E. A Method of State-of-Charge Estimation for EV Power Lithium-Ion Battery Using a Novel Adaptive Extended Kalman Filter. *IEEE Trans. Veh. Technol.* **2020**, *69*, 14618–14630. [CrossRef]
56. Zhang, Z.; Jiang, L.; Zhang, L.; Huang, C. State-of-Charge Estimation of Lithium-Ion Battery Pack by Using an Adaptive Extended Kalman Filter for Electric Vehicles. *J. Energy Storage* **2021**, *37*, 102457. [CrossRef]
57. Xu, W.; Wang, S.; Jiang, C.; Fernandez, C.; Yu, C.; Fan, Y.; Cao, W. A Novel Adaptive Dual Extended Kalman Filtering Algorithm for the Li-Ion Battery State of Charge and State of Health Co-Estimation. *Int. J. Energy Res.* **2021**, *45*, 14592–14602. [CrossRef]
58. Liu, Z.; Li, Z.; Zhang, J.; Su, L.; Ge, H. Accurate and Efficient Estimation of Lithium-Ion Battery State of Charge with Alternate Adaptive Extended Kalman Filter and Ampere-Hour Counting Methods. *Energies* **2019**, *12*, 757. [CrossRef]
59. Ramadan, H.; Becherif, M.; Claude, F. Extended Kalman Filter for Accurate State of Charge Estimation of Lithium-Based Batteries: A Comparative Analysis. *Int. J. Hydrogen Energy* **2017**, *42*, 29033–29046. [CrossRef]
60. Meng, J.; Stroe, D.-I.; Ricco, M.; Luo, G.; Teodorescu, R. A Simplified Model-Based State-of-Charge Estimation Approach for Lithium-Ion Battery with Dynamic Linear Model. *IEEE Trans. Ind. Electron.* **2019**, *66*, 7717–7727. [CrossRef]
61. Abu-Sharkh, S.; Doerffel, D. Rapid Test and Non-Linear Model Characterisation of Solid-State Lithium-Ion Batteries. *J. Power Sources* **2004**, *130*, 266–274. [CrossRef]
62. OpenEPT. Available online: https://github.com/turkmanovic/OpenEPT.git (accessed on 6 September 2024).
63. Turkmanović, H.; Karličić, M.; Rajović, V.; Popović, I. High Performance Software Architectures for Remote High-Speed Data Acquisition. *Electronics* **2023**, *12*, 4206. [CrossRef]
64. Pereira, R.; Couto, M.; Ribeiro, F.; Rua, R.; Cunha, J.; Fernandes, J.P.; Saraiva, J. Ranking Programming Languages by Energy Efficiency. *Sci. Comput. Program.* **2021**, *205*, 102609. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.