Online State-of-Health Assessment for Battery Management Systems

Mihai Victor Micea, *Member, IEEE*, Lucian Ungurean, Gabriel N. Cârstoiu, and Voicu Groza, *Fellow, IEEE*

***Abstract*—Battery-powered embedded systems have known a rapid evolution in recent years, as nickel–metal hydride (Ni–MH) battery technology has enabled important reductions in size and proportional increases in total capacity over the older nickel–cadmium (Ni–Cd) and lead–acid battery types. This paper addresses the problem of state-of-health (SoH) estimation and prediction for use in resource-constrained Ni–MH-battery-powered embedded systems. We propose a novel SoH prediction methodology, presenting both a theoretical analysis of the estimation algorithm and the detailed description of hardware and software implementation. Two versions of estimation algorithms are proposed, along with the analysis of their performances in terms of prediction accuracy and required processing power, as the SoH prediction is designed to run online, being part of an embedded battery management system.**

***Index Terms*—Battery management, battery-powered device, nickel–metal hydride (Ni–MH), state-of-health (SoH) prediction.**

I. Introduction

THE NUMBER OF battery-powered mobile and portable devices has seen an exponential growth in recent years, among the most obvious examples being laptops, mobile phones, personal digital assistants (PDAs), smart sensors [1], [2], etc. The processing power of these devices has also been rapidly growing, thus determining the development and evolution of both primary and secondary battery technologies needed to support longer usage times between recharges and replacements of batteries.

Nickel–metal hydride (Ni–MH) technology offers lower cost compared to lithium ion (Li-ion)-based solutions in many applications, because it provides high storage capacity (up to 2700 mAh for standard AA cells) and high energy density, contains chemical elements which do not have a bad environmental impact, and features good charging and discharging rate capabilities [3], [4]. Another advantage of Ni–MH batteries worth mentioning is the good tolerance to fast charging (using charging currents up to 1 C). The main disadvantage of using this technology is its low robustness when it comes to extreme conditions which affect the battery storage capacity and, therefore, its performance [5]. Another problem that remains is the self-discharge rate which is not negligible, but it can be quantified and used in calculations.

All of the aforementioned advantages favor the usage of Ni–MH cells in battery-powered embedded systems which are resource constrained in terms of size, processing power, and battery capacity. Although most consumer handheld and portable devices use Li-ion batteries as their energy source, Ni–MH batteries can still be found in devices such as power tools, walkie-talkies, digital enhanced cordless telecommunication (DECT) phones, GPS receivers, digital cameras, etc. Furthermore, Ni–MH is the preferred energy source for smart sensor nodes used in monitoring and surveillance applications.

A. Previous Work

Previous papers in the field of battery management systems (BMSs) for Ni–MH cells have targeted areas such as charge termination techniques [6], accurate state-of-charge (SoC) determination [4], innovative charger designs [7], [8], and discharge coordination algorithms to improve battery lifetime [9].

Considering the area of charge termination techniques, many of the novel developments that have been published thus far have focused on improving certain characteristics of the already-known and accepted techniques in common use for Ni–MH batteries [6], such as the rate of change of the battery terminal voltage (dV slope and voltage plateau detection) and the maximum rate of change in battery temperature (dT/dt).

Among the notable ideas are the ones presented by Diaz *et al.* in [8], in which they describe a charge termination technique based on detection of the battery voltage second slope, effectively stopping the charging process as soon as the time windows between consecutive increases in terminal voltage grew over a certain time threshold.

SoC determination is one of the key issues relating to battery-powered devices, and throughout the history of the development of secondary battery cells and their applications, several systems and techniques have been specified and developed [4]. Many recent developments in the field have concentrated on batteries with alkaline [10], lead–acid [11], and Li-ion chemistries. The authors in [12] and [13] have proposed and demonstrated an ingenious solution for improving the accuracy of SoC determination by combining the classical Coulomb-counting method with a SoC correction algorithm, based on a second-order Randles model of the batteries. However, the proposed algorithm is based on offline parameter identification and assumes a fixed lookup table relating the open-circuit voltage (OCV) to the SoC, which is scarcely true for real batteries.

Manuscript received July 21, 2010; revised September 30, 2010; accepted November 14, 2010. Date of publication March 17, 2011; date of current version May 11, 2011. This work was supported in part by the Romanian Ministry of Education and Research under Grant PNCDI II ID-22/2007-2010. The Associate Editor coordinating the review process for this paper was Dr. Kurt Barbe.

M. V. Micea, L. Ungurean, and G. N. Cârstoiu are with the Department of Computer and Software Engineering, “Politehnica” University of Timisoara, 300223 Timisoara, Romania (e-mail: mihai.micea@cs.upt.ro).

V. Groza is with the School of Information Technology and Engineering, University of Ottawa, Ottawa, ON K1N 6N5, Canada.

Digital Object Identifier 10.1109/TIM.2011.2115630

Research work focusing on smart charger designs for Ni–MH batteries has been growing in recent years [7], [8] as these types of batteries are very sensitive to charge and discharge termination conditions and also as a result of more demanding requirements for reduced charging times and increased charging efficiency. The approach in [6] focuses on the fast charging process, providing a limited degree of intelligence by determining what amplitude of charge current is required (fast or trickle) by using two levels of discrimination for the battery terminal voltage (V_{batt}), at 10% of V_{batt} and at 80% of V_{batt}. The design proposed in [8] implements the charging algorithm using an 8-bit microcontroller and is based on generating current pulses and observing the battery terminal voltage. It offers a certain degree of flexibility by allowing the user to select between four predefined nominal capacities for the batteries used.

A complete charger system with SoC measurements and a battery capacity learning feature is presented in [14]. The authors focus on minimizing the charging system power consumption and succeed in attaining a power envelope of under 100 μA. However, the reported accuracy of the measurements is not supported by experimental results, and the hardware implementation is costly due to its large number of discrete analog components.

B. Proposed Design

The main idea of this paper is to present in detail a complete BMS, which can be utilized stand-alone or as part of a battery-powered embedded device. We will discuss the novel features introduced by our design and will evaluate the validity of our implementation by presenting a case study and the detailed experimental results.

Our solution provides in-system charging of the attached battery pack, SoC calculation, and state-of-health (SoH) prediction of the remaining number of cycles or the remaining useful life (RUL), as defined in [15]. Similar to other intelligent charger designs [7], our charge controller module makes use of a combination of charge termination techniques, relying mainly on voltage- and temperature-based methods, as well as on the accurate measurement of the total supplied energy via a hybrid Coulomb-counting algorithm.

One of the novelties introduced by this paper is the integration of the battery pack, the battery charge and discharge controller, the SoC measurement hardware, and the battery management software into a single module, thus providing a configurable black-box solution that is ready to be integrated into battery-powered devices. We will present the implementation details of our complete hardware–software solution, as well as a case study implementation in Section III.

The ability of an intelligent BMS to accurately measure the total energy supplied to and removed from the battery pack is mandatory for correct SoH determination. Our design implements SoC calculation based on a reduced version of the mixed algorithm proposed by Codeca *et al.* in [12], in which they demonstrate that very accurate SoC measurements can be obtained when combining the classical Coulomb-counting approach with feedback from an electrical battery model, based on the OCV–SoC dependence. The novelty introduced by our solution, as detailed in Sections II and III, consists of the online recalculation of the OCV–SoC model parameters, as opposed to the static and offline model identification presented in [12]. This idea is based on the observation that the battery model parameters need to be modified as each battery cycle is consumed, to take into account the subsequent SoH deterioration caused by battery aging.

The SoH prediction capability is central to our design and represents another novel feature. It provides the users with a vital piece of information about the health and the remaining operating cycles of the integrated battery pack, allowing them to know in advance when to replace the used cells. Although several SoH estimation algorithms have been reported in the art [15], [16], their complexity makes them hardly suitable to be implemented in a low-cost and low-power embedded system, and to our knowledge, no such attempts have been reported.

Our proposed SoH estimation and prediction solution is based on a second-order parabolic regression algorithm, and in Sections III and IV, we will elaborate on the numerical methods that we have found suitable to be implemented on an embedded processor. We will analyze two proposed implementation concepts in terms of relative estimation error versus actual battery data and the CPU and memory usage of the embedded implementation case study.

Section V draws the conclusions and summarizes the obtained results.

II. Theory of Operation

A. SoC Calculation Principles

The SoC is an important parameter for all battery-powered devices, as it is used to provide an indication of the remaining operating time for the current discharge cycle or the remaining time until the batteries are charged, in the charging phase. Although many methods exist for SoC indication [4], [10], [12], the most widely studied and most used ones are the following: Coulomb counting, which is relatively simple to implement and can be used with all battery types; impedance spectroscopy, which can also be generally used but is expensive to implement; fuzzy logic approaches, which can provide accurate estimates but present large memory requirements; and Kalman filters, which are robust and adaptable to multiple battery types but require a complex battery model and are difficult to implement.

There are multiple ways in which SoC can be defined, but the most commonly accepted definition is the following: The SoC of a battery pack is the maximum remaining useful capacity stored in the respective battery pack. SoC can also be described as a percentage value relative to the nominal capacity of the battery pack, expressed in ampere-hours (Ah)

$$SoC(t) = \frac{Q_{\text{nom}} - \int_0^T I(t)dt}{Q_{\text{nom}}} \cdot 100. \tag{1}$$

In (1), the SoC is denoted by $SoC(t)$, and it can hold a value from 0 to 100; the nominal capacity is Q_{nom}, and the current flowing to/from the batteries is denoted by $I(t)$ and can have a positive value in the charge phase and a negative value during discharge.

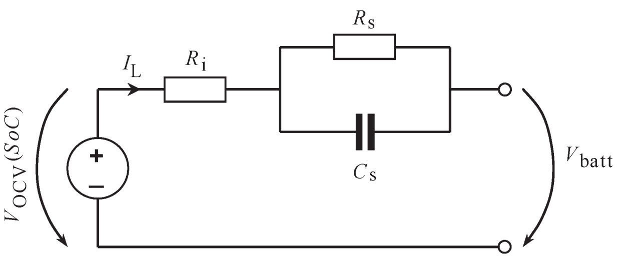


Fig. 1. Proposed Ni–MH electrical battery model.

A direct consequence of the SoC definition is that the SoC can be determined by accumulating the changes in the charge level, effectively integrating the current flow over the operating time. This bookkeeping method is known as Coulomb counting. Although its implementation is simple and suitable for a microcontroller-based BMS, this method presents some drawbacks: Its performance is directly dependent on the accuracy of current measurements, requiring very precise current sensing, and significant offset errors can be introduced if the initial SoC $(SoC(0))$ is not known precisely [13]. A more accurate solution is presented in [12]. Here, the authors make use of a second-order resistor–capacitor Randles model. The model inputs are the load current and an OCV–SoC lookup table. The model parameters can be identified by measuring the voltage levels and settling times after applying charge and discharge current pulses. The model identification and the OCV–SoC lookup table determination are done offline using a PC-controlled test bench.

Our proposed SoH estimation algorithms need accurate SoC measurements as inputs; therefore, we have implemented an adaptation of the mixed algorithm described earlier. The applications targeted by our BMS design will use currents in the range of 200–1000 mA. In the field of battery engineering, this range corresponds to an interval of 0.1–0.5 C of a 2000-mAh battery pack, for which the rated discharge current of 1 C is equal to 2000 mA. As a consequence, according to Viera *et al.* [5], the maximum available SoC does not depend on the charge or discharge current applied to the batteries. Furthermore, the OCV–SoC curve changes as the batteries advance in age, leading to the observation that the OCV and the maximum available SoC are functions of the battery SoH.

Thus, we propose an adaptation of the mixed algorithm for SoC determination described in [12]. Its most important and novel feature is the possibility of determining and recalculating the battery model parameters online, as part of the system operating time. We have considered a simplified first-order Randles model for the Ni–MH batteries, shown in Fig. 1, in order to make it feasible to be implemented as part of an embedded BMS and to shorten the testing times necessary for identifying its parameters.

The proposed model expresses the battery terminal voltage V_{batt}, considering a load current I_L, as a function of the battery OCV $V_{\text{OCV}}(SoC)$. The battery overpotential $(V_{\text{OCV}}(SoC) - V_{\text{batt}})$ is modeled by the voltage drop across its internal resistance R_i, and the response to a step current signal is handled by the parallel R_s–C_s group

$$V_{\text{batt}}(t) = V_{\text{OCV}}(SoC) - I_L\left(R_i + R_s\left(1 - e^{\frac{-t}{R_sC_s}}\right)\right). \quad (2)$$

In the Laplace domain, (2) can be represented with the following relation:

$$V_{\text{batt}}(s) = V_{\text{OCV}}(SoC) - I_L\left(R_i + \frac{R_s}{1 + sR_sC_s}\right). \quad (3)$$

The aforementioned relations link the directly measurable parameters V_{batt} and I_L to the model parameters and the OCV, where t represents the elapsed time of the current cycle. This relation will be used by the BMS software implementation to periodically recalculate the model parameters as needed.

B. SoH Estimation

The SoH is a metric which indicates the battery condition related to a new battery. SoH determination is not a simple task. In most cases, several parameters are involved in this process, like cycle numbers, accurate SoC determination, etc. There is a strong dependence between the SoH and both the battery and the type of applications running on the battery-powered device. This degree of unpredictability can be controlled using adaptive systems based on neural networks, Kalman filtering, or fuzzy logic [17]. The problem with these systems is the high computational power needed for implementation, which is unfeasible for low-cost consumer smart chargers.

A simple approach is presented in [17]. The SoH indication is based on the stored maximum capacity function C_k for some charge/discharge cycle k. This paper focuses on the improvement of this paradigm by introducing curve modeling and estimation using polynomial regression. We consider that this simple algorithm is well suited for consumer smart chargers with small computational power, as we will show in the following sections.

A least squares approach is proposed to implement the second-order polynomial regression which estimates the C_k function.

The polynomial representation of C_k is given by

$$C_k = ak^2 + bk + c, \qquad a < 0. \quad (4)$$

The constraint imposed to a ensures a better curve approximation because of its monotonically decreasing part.

Consider a battery operating scenario with a total of n charge/discharge cycles completed up to the current moment. As a result, the maximum capacity value C_k for each successive battery cycle can be obtained. The curve estimation using least squares algorithms is determined by solving the following system of equations:

$$\begin{cases} a\sum_k k^2 + b\sum_k k + cn = \sum_k C_k, \\ a\sum_k k^3 + b\sum_k k^2 + c\sum_k k = \sum_k kC_k, \\ a\sum_k k^4 + b\sum_k k^3 + c\sum_k k^2 = \sum_k k^2C_k, \end{cases} \quad k = \overline{1,n}; \quad n \geq 3. \quad (5)$$

To solve the system and to obtain the a, b, and c parameters, a simple algorithm, such as Cramer's algorithm, can be used. It consists of computing the following determinants:

$$\Delta = \begin{vmatrix} \sum_k k^2 & \sum_k k & n \\ \sum_k k^3 & \sum_k k^2 & \sum_k k \\ \sum_k k^4 & \sum_k k^3 & \sum_k k^2 \end{vmatrix} \tag{6}$$

$$\Delta_a = \begin{vmatrix} \sum_k C_k & \sum_k k & n \\ \sum_k kC_k & \sum_k k^2 & \sum_k k \\ \sum_k k^2 C_k & \sum_k k^3 & \sum_k k^2 \end{vmatrix}$$

$$\Delta_b = \begin{vmatrix} \sum_k k^2 & \sum_k C_k & n \\ \sum_k k^3 & \sum_k kC_k & \sum_k k \\ \sum_k k^4 & \sum_k k^2 C_k & \sum_k k^2 \end{vmatrix}$$

$$\Delta_c = \begin{vmatrix} \sum_k k^2 & \sum_k k & \sum_k C_k \\ \sum_k k^3 & \sum_k k^2 & \sum_k kC_k \\ \sum_k k^4 & \sum_k k^3 & \sum_k k^2 C_k \end{vmatrix}.$$

In this case, the values of the a, b, and c parameters are expressed using

$$a = \frac{\Delta_a}{\Delta} \qquad b = \frac{\Delta_b}{\Delta} \qquad c = \frac{\Delta_c}{\Delta}. \tag{7}$$

For calculating the sums in the determinants specified in (6), the following recurrent formulas are suited for embedded implementation:

$$S_{k+1} = S_k + (k+1), \quad S_{(k+1)^2} = S_{k^2} + (k+1)^2, \quad \text{etc.} \tag{8}$$

Using the polynomial function obtained, we can compute the first cycle m for which the discharged capacity is under a threshold value, denoted as $F \cdot C_{\text{nominal}}$, and the SoH is considered 0%.

The computation of m is reduced to solving the inequality

$$F \cdot C_{\text{nominal}} > ak^2 + bk + c. \tag{9}$$

Since $a < 0$, m becomes

$$m = \left\lfloor \frac{-b - \sqrt{b^2 - 4ac}}{2a} \right\rfloor \tag{10}$$

where $\lfloor x \rfloor$ (floor) represents the largest integer less than or equal to x.

The least squares algorithm is well suited for linear systems where the approximation has the highest accuracy. Of course, if the values considered for interpolation present nonlinearities, the accuracy decreases. However, the applications targeted by the proposed BMS design use discharge currents between 200 and 1000 mA, which are well below the maximum rated discharge current (specified by the battery manufacturer datasheets). Thus, nonlinearities relating to the degradation of the battery capacity have a low probability of appearance [5].

The SoH estimation principle that we have adopted for integration in the BMS is based on previously stored measurements of the total available discharge capacity of the battery pack. As a consequence, we have identified two methods for SoH estimation using the available battery data and the number of elapsed battery cycles: the *time-window algorithm* and the *history-based algorithm*.

The *history-based algorithm* for SoH estimation has been devised starting from the observation that the typical Ni–MH cell discharge curve presents a relatively constant decreasing curve over its expected lifetime [18]. This method makes use of the whole charge–discharge history of the cell up to the current charge/discharge cycle. The basic idea supporting this algorithm is that the estimation will be more accurate as more inputs are used.

The *time-window algorithm* is based on the following idea: Considering that the current battery cycle is denoted by C_n, then the measured capacities of the previous $n-1$ cycles (C_1 through C_{n-1}) will be used to estimate the remaining u useful cycles. The value of u is defined according to the aforementioned requirements. As the number of battery cycles advances, so do both boundaries of the considered time window, providing the estimation algorithm with a moving time window of the previous $n-1$ cycles. Among the advantages of this approach, we can note that, in the case of a new battery pack, $n \ll u$, and therefore, relatively few logged discharge cycle data are needed to estimate the RUL. For example, the experimental results presented in Section IV show that a number of up to $u = 60-70$ future cycles can be estimated by considering only $n = 25$ recorded discharge capacities.

Considering the two data-gathering approaches for the SoH estimation algorithm, the BMS designer must choose the best suited implementation, taking into consideration such factors as the nominal cell capacity, the number of cells/pack, and the available computing power and memory storage (for keeping the measured battery capacity logs).

III. BMS Implementation Case Study

The algorithms presented in the previous section have been implemented on an embedded power management module called the Power Management Board (PMBoard). This board is part of the wireless intelligent terminal (WIT) architecture specified by the Collaborative Robotic Environment–The Timisoara Experiment (CORE-TX) project [2], [19]. The PMBoard has several functions: to supply power to the entire system, to measure and manage the power consumption [20] of the entire WIT, to increase the life and SoH of the battery pack, and to provide battery system parameters to the upper levels of the system power management (SPM) software.

The algorithms for the management of the power consumption were implemented on an ARM7 based microcontroller (MCU) [22] and rely on accurate measurements of the voltage, current, and temperature values. The temperature is measured using a TMP101 sensor [21], and the voltage is measured by using a 10-bit successive approximation register analog-to-digital converter (ADC) [22]; for the current measurement, a high-side current monitor schematic has been implemented, as

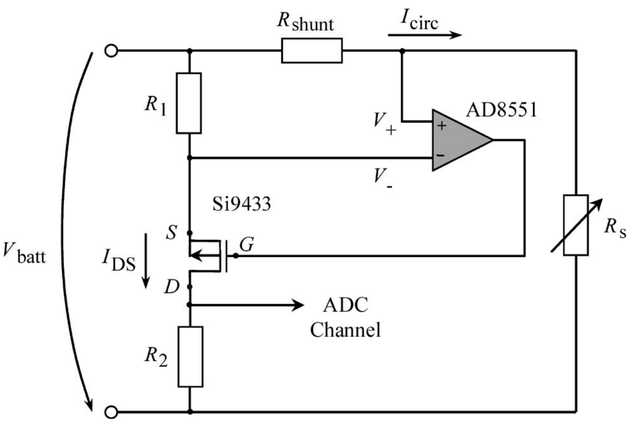


Fig. 2. High-side current monitor schematic.

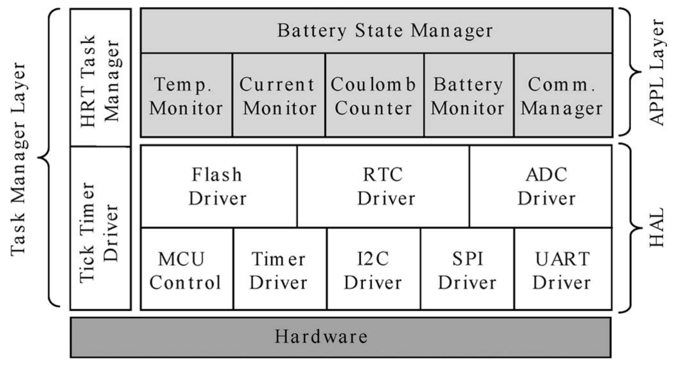


Fig. 3. BMS software layered architecture.

shown in Fig. 2, for a particular load channel which corresponds to an individual component board of the WIT.

In the aforementioned figure, R_s is the load resistance, I_{circ} is the current through the circuit, and R_{shunt} is the shunt resistor. We used 0.5% accuracy resistors and software averaging techniques [23], to increase the accuracy of the measurements and also to reduce the influence of noise.

Details concerning the power consumption of the monitor schematic are given in the following. The power dissipated on the high-side current monitor is the sum of partial powers on its components. R_1, R_2 and Si9433 are connected in series to V_{batt}, such that the power dissipated on them can be minimized only by reducing I_{DS}. This current has to be an order of magnitude higher than the input current of the ADC connected to R_2. Given the high impedance of the ADC, we set I_{DS} to be in the range of microamperes. The AD8551 operational amplifier draws 750 μA from the battery, such that its contribution cannot be modified by design. The power consumption of the shunt resistor (which has the highest power profile in the schematic) is proportional to the current through the load and has a maximum value of 500 mW, considering a 0.5-Ω shunt and the maximum load current of 1000 mA. This power figure can be reduced by decreasing the value of the shunt resistor.

SoC determination is done using the model shown in Fig. 1. The system has the ability to identify the model parameters and to recalculate them every time the batteries are replaced. The identification of the system parameters is done by utilizing additional hardware for adjusting the amplitudes of the charge and discharge pulse currents. The details of this additional logic are beyond the scope of this paper.

The power management software, as a key part of the PM-Board BMS implementation, is schematically shown in Fig. 3. It consists of a modular embedded application which must ensure the following three layers of functionality.

1) The battery state manager (APPL layer) provides the measurement, storage, and processing of the main battery parameters, as described earlier, as well as the SPM command and response handling.
2) The task manager layer provides the scheduling and execution of the hard real-time (HRT) tasks, according to the HRT compact kernel model [24].
3) The hardware abstraction layer contains the drivers and interfaces of the microcontroller unit (MCU) and board peripherals.

The estimation software component, which is part of the battery state management layer, handles both the estimation of the remaining operating time of the current discharge and our novel SoH prediction algorithm. As detailed in Section II, the prediction of the RUL is achieved by first calculating the polynomial coefficients via the parabolic regression and then solving the second-order inequality of (9).

Although this algorithm was chosen as a tradeoff between the implementation complexity and the available computing power, it had to be designed to tackle some target-specific problems, related to the 32-bit ARM7 based platform [22] used as case study. Such problems include the following.

1) The magnitude of the intermediate calculation results exceeds the 32-bit boundary and must be represented as 64-bit numbers.
2) The MCU architecture offers no native support for floating-point operations; therefore, the algorithm inputs (e.g., the battery charge measurements) must be scaled and truncated to integers.
3) The calculation of the square root of a 64-bit number is required for solving the final inequality, while the standard C library provides a slow inefficient version of the `sqrt()` function.

One of the main requirements of the BMS software is that its components must be designed to be executed within real-time constraints and that its duty cycle must be minimized in order to reduce the system power consumption. For these reasons, the SoH estimation component must be implemented using all available optimizations to reduce its execution time and memory footprint.

The identification of the main optimization aspects has been carried out by analyzing the mathematical relations detailed in Section II. We have found that the best method of calculating the four third-order determinants, in terms of favoring additions and subtractions over multiplications, is the recursive expansion in cofactors using the first row [25] rather than the classical Sarrus or triangle rules. Using this method, a total of 12 second-order determinants will be generated. Another key observation that we have made is that only seven of the aforementioned determinants are distinct, yielding a significant reduction of the required calculations.

In Section II, we have mentioned that an immediate optimization would be to calculate the summation terms recursively without the need to retain all the previously measured battery charge values, thus reducing memory space. In addition, a significant optimization of the processing power and memory space has been achieved by observing that, out of the total of 24 summation terms implied by the second-order determinants, only seven are distinct.

As to the square-root calculation, we have implemented an iterative successive approximation method based on the observation that, for a given integer number of b bits, its square root is represented by, at most, $b/2$ bits.

Considering all these observations, the implementation of the proposed SoH estimation algorithm for the calculation of the number of RUL cycles (n_{RUL}) is presented in Algorithm 1. Here, n denotes the current number of measured battery cycles, $n_{\min}$ is the minimum number of cycles which are necessary for the start-up of the algorithm, and Δ_1 to Δ_7 are the seven distinct second-order determinants previously mentioned. This requirement applies to both variants of the algorithm and implies that the SoH estimation cannot be performed until at least a number of $n_{\min}$ battery cycles have been recorded. The minimum number of battery cycles is equal to the time-window size, and different values can be chosen, as detailed in the next section.

```
Algorithm 1. RUL estimation algorithm.
1: **while** not (battery cycle completed) **do**
2:        wait
3: **end while**
4: increment $n$
5: **if** $n \geq n_{\min}$ **do**
6:        Calculate $Sum_1, Sum_2, \ldots, Sum_7$ with (8)
7:        Calculate $\Delta_1$, $\Delta_2$, $\Delta_3$
8:        Calculate $\Delta$
9:        Calculate $\Delta_4$, $\Delta_5$, $\Delta_a$
10:         Calculate $\Delta_6$, $\Delta_b$
11:         Calculate $\Delta_7$, $\Delta_c$
12:         // Scaling of $\Delta$, $\Delta_a$, $\Delta_b$, $\Delta_c$
13:               $\Delta = \Delta/128$
14:               $a = (\Delta_a$ left shifted 20 bits$)/\Delta$
15:               $b = (\Delta_b/\Delta)$ left shifted 20 bits
16:               $c = (\Delta_c/\Delta)$ left shifted 20 bits
17:         Solve (9) and calculate $m$ according to (10)
18:         $n_{\mathrm{RUL}} = m - n$
19: **end if**
```

The differences between the time-window and the history-based algorithms reside in the calculation of the seven summation terms (Sum_1 to Sum_7) and will be discussed in the following sections. Additional optimizations have been introduced and are shown in lines 12 to 16 of the algorithm. They address the numerical adaptation of the polynomial coefficients (a, b, and c) due to the fact that we are dealing with large numbers, which need to fit 64 bits, without compromising the accuracy of the calculations.

IV. Performance Evaluation

For the performance evaluation, we have considered two sets of data. The first set contains the estimated values for the cycle in which the battery capacity drops under a fraction of the nominal capacity, which is a value for which the SoH of the battery is considered 0%. This value is specific for each type of battery, and usually, it can be found in the documentation provided by the manufacturers. The second set of data contains the real cycle number for which the capacity reaches the terminal SoH value specified earlier.

The data have been obtained from cycling two types of Ni–MH battery packs: GP batteries with a nominal capacity of approximately 2400 mAh and aged Sanyo batteries designed for 2200 mAh. For the evaluation, a factor of 80% of the nominal capacity is considered to be the threshold for which the terminal SoH is reached. Our experimental data show that the respective condition occurs at the 85th cycle for the GP pack and at the 69th cycle for the Sanyo pack.

The cycling of the batteries through multiple charge and discharge phases has been accomplished using an automated test setup constructed using the PMBoard presented in Section III, and the gathered data have been used as inputs for the aforementioned data sets. The batteries have been configured in packs of two cells connected in series at a nominal voltage of 2.4 V. The average discharge currents have been 210 mA or 0.09 C (at the rated capacity of 2400 mAh) for the GP battery packs and 420 mA or 0.19 C (at the rated capacity of 2200 mAh) for the Sanyo packs.

In charge mode, a total of 2300–2600 mAh has been supplied to the battery pack in each cycle, using the constant current charging strategy and terminating the charge at the activation of one of the classical termination techniques, such as the "V plateau" [14]. In discharge mode, the battery pack has been automatically switched to supply a programmable resistive dc load using specialized hardware embedded into the PMBoard to simulate the typical current consumption of a WIT.

A. Evaluation of the SoH Estimation Algorithms

The maximum discharge capacities for the tested battery packs have been recorded, and the cycle number at which the 0% SoH point is reached has been noted. A comparative depiction of these data is shown in Fig. 4. The cycle number represents the number of test cycles elapsed since the start of the experiment and not the absolute discharge cycle of the pack elapsed since first use. The figure shows the degradation trend lines of the two types of batteries, emphasizing the lower SoH of the Sanyo batteries.

Two types of SoH estimation algorithms have been implemented: the time-window algorithm, with a window of size $W = 25$ and $W = 30$ cycles, and the history-based technique. In the first case, the estimation has been made using the values recorded in the last 25 and last 30 cycles. For the history-based algorithm, the estimation is made based on all the recorded values up to the current cycle on which the estimation is computed.

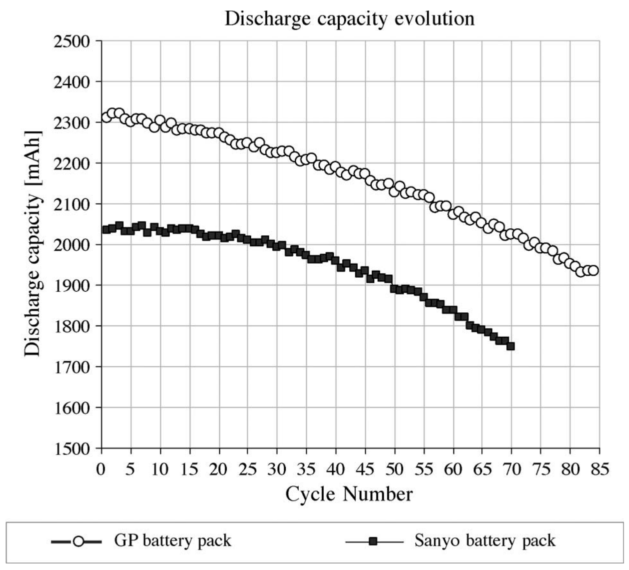


Fig. 4. Discharge capacity evolution for the GP and Sanyo battery packs.

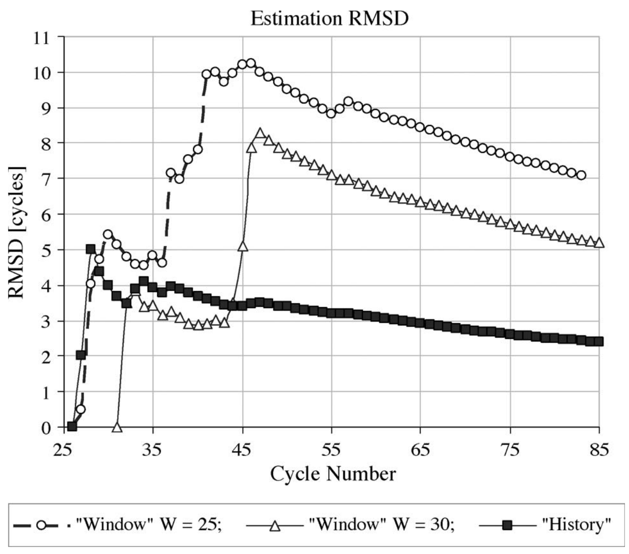


Fig. 6. Estimation RMSD for the embedded implementation.

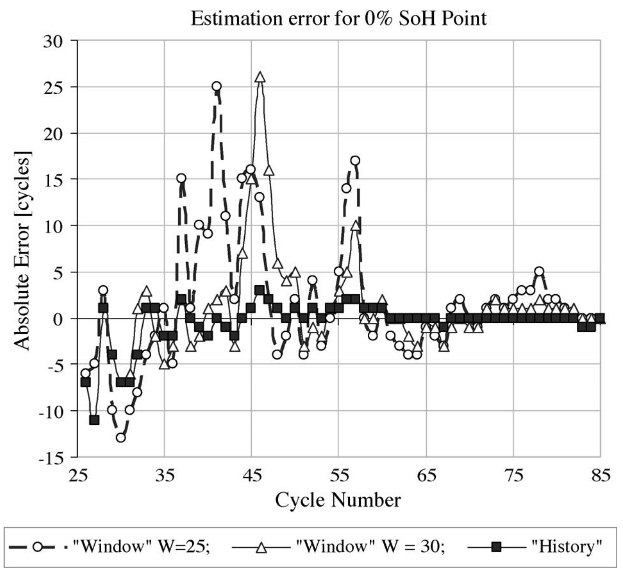


Fig. 5. GP batteries: Absolute estimation error for the embedded implementation.

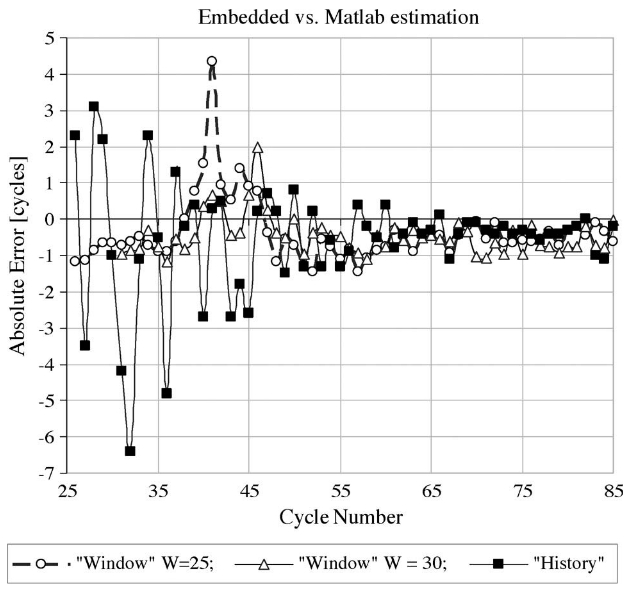


Fig. 7. Absolute error comparison between the Matlab implementation and the embedded implementation.

In the following performance evaluation of the algorithms, we compare the theoretical results obtained from executing the algorithms in a Matlab environment with the experimental results obtained from running the embedded implementation.

The experimental results obtained during the performance evaluation of the embedded algorithm implementation on the GP batteries are shown in Figs. 5 and 6. The embedded implementation is different from Matlab because of the integer arithmetic which causes rounding errors. There is also the problem of number representation. To suppress the rounding errors, the algorithms on the embedded side are implemented using a 64-bit integer representation.

As shown in Fig. 5, the history-based algorithm estimation error approaches zero starting from the 60th execution cycle. However, the algorithm occasionally produces gross estimation errors, which are visible, for example, between cycles 35 and 50. A solution to improve the robustness of the estimation is to filter out the errors at the application level by replacing the out-of-range values with the average of the previous cycles. The root-mean-square deviation (RMSD) of the embedded estimator is specified in Fig. 6. The deviation is higher than the theoretical one and has a maximum value of 10.5 cycles.

We have also evaluated the estimation errors introduced by implementing the algorithm on the embedded platform, as compared to those of the Matlab implementation. For the first 20 estimation cycles, the absolute error varies between -6 and 4 cycles. For the next estimation cycles, the theoretical and the experimental results generate similar values. Fig. 7 shows the absolute error between the Matlab implementation and the embedded implementation.

Fig. 8 shows the absolute estimation error of the embedded implementation for the Sanyo battery pack. Compared to Fig. 5, we can observe that the initial estimation errors are smaller, but the algorithm prediction accuracy is maximized starting

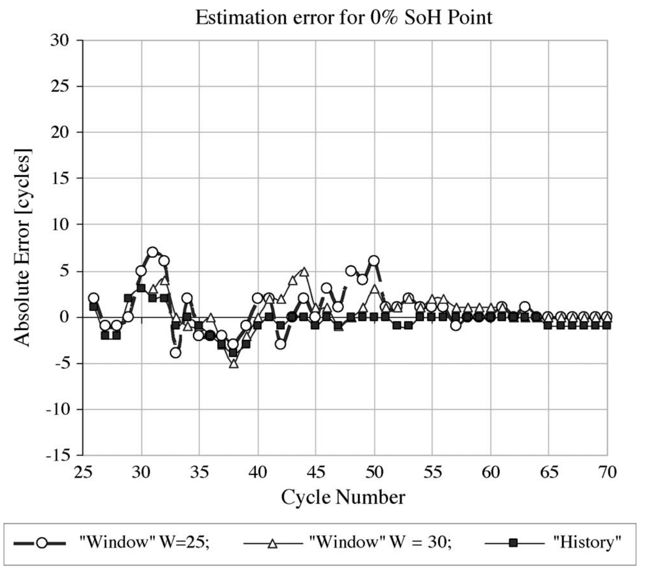


Fig. 8. Sanyo batteries: Absolute estimation error for the embedded implementation.

TABLE I
MEMORY REQUIREMENTS OF THE BMS AND THE SOH ESTIMATION SOFTWARE

Memory	BMS [B]	SoH module [B]	% of BMS
Code ROM	35936	3034	8.44
Constant ROM	2056	32	1.56
Data RAM	8624	176	2.04

approximately from the 55th cycle. Thus, we can observe that the proposed SoH estimation approach is viable even for aged battery packs.

B. Evaluation of the Embedded Resource Requirements

As shown in Section III, we have integrated the embedded software implementation of our proposed SoH estimation algorithms into the battery management layer. We have designed and implemented the SoH estimation calculations observing the real-time requirements imposed by the WIT architecture while minimizing the additional memory and processing power overhead.

Table I shows the memory requirements of the SoH estimation module in comparison with the total memory footprint required by the BMS software. The implementation contains the routines for both types of algorithms in order to offer increased flexibility to the system designer. The three rows represent the comparison metrics: the total ROM (or Flash memory) used for code storage, the ROM used for constant data, and the total RAM used for run-time data storage. All values are expressed in bytes, and the third row gives the percentage values for the memory requirements of the SoH module implementation with respect to the memory usage of the entire BMS software.

These comparisons clearly show that we have succeeded in implementing the SoH estimation algorithms with a low memory overhead while also considering all the optimization aspects identified in Section III. We can also conclude that a complete BMS implementation, which provides SoH estimation, is entirely feasible on an ARM7 platform similar to the one we have used, [22], because all such MCU platforms integrate a minimum of 64 kB of Flash and 16 kB of RAM, thus becoming a very cost effective solution.

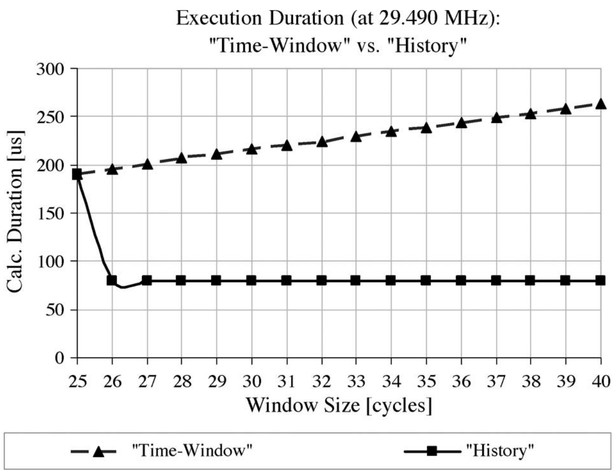


Fig. 9. Time-window versus history-based execution durations.

The power consumption of the BMS is another important evaluation factor. The hardware design has been done with clear low-power principles in mind: PMOS gates as switching devices for the charge current, low-power and high-efficiency buck–boost regulators, and the predominant use of integrated components. For the current BMS implementation, the minimum available CPU frequency of 14.745 MHz is used, and the BMS CPU time is 31%, yielding a total power consumption of 13.95 mW. This power envelope can be further reduced with lower power alternatives such as the Texas Instruments MSP430 or the Energy Micro EFM32 MCUs.

Concerning the required processing power of our proposed implementation, we have done a series of tests to evaluate whether the resulting calculation durations fit within certain real-time boundaries. For this, we have varied the CPU frequency of the used MCU in steps starting from 14.745 up to 58.980 MHz. We have also varied the window size of the SoH estimation algorithm from $W = 25$ to $W = 40$ cycles in order to see the evolution of the calculation times for different window sizes and to offer the system designer an area of solutions in the CPU frequency and window size domains.

After measuring the execution durations for the time-window algorithm for different CPU frequencies, we have observed that the maximum execution duration does not exceed 550 μs even in the worst case, i.e., at 14.745 MHz. This conclusion is important for a system designer who wishes to integrate the SoH estimation into the BMS because it proves that the SoH algorithms add little to the real-time constraints of the system, allowing for more relaxed timing configurations.

To compare the history-based and the time-window algorithms, a similar test has been conducted, emphasizing the effect of the recursive accumulation optimization discussed in Sections II and III. We have considered a fixed CPU frequency of 29.490 MHz, and we have varied the window size as in the previous test. The results are shown in Fig. 9, and it should be noted that, in the case of the history-based algorithm implementation, the window size parameter expresses the actual number of cycles of the battery pack, while with each increment, a recursive accumulation of the summation terms takes place. Thus, we have shown that the history-based algorithm offers a significant improvement in processing time, due to the fact that the summation terms are only calculated once (at the 25th cycle) and the subsequent calculations rely on recursive accumulation, which is more computationally efficient.

V. CONCLUSION

In this paper, we have proposed two algorithms based on the least squares regression method for battery SoH estimation, and we have adapted them for ease of integration into an embedded battery management solution. We have described in detail several important optimizations required for implementing the mathematical algorithms on an MCU, powered by the ARM7 architecture.

The SoH prediction algorithms have been successfully integrated into a BMS solution developed at the Digital Signal Processing Laboratories in Timisoara, Romania, as a central part of the CORE-TX platform.

We have conducted extensive tests in order to evaluate the accuracy and real-time feasibility of the proposed solutions. The results show that the history-based algorithm ensures a more accurate estimation than the time-window technique, although both types present a very good accuracy of estimation, starting from approximately 50% of the RUL of the battery.

Concerning the required execution times, the recursive accumulation optimization produces a dramatic reduction in the execution time of the history-based algorithm.

REFERENCES

[1] O. Kanoun and H. R. Trankler, "Sensor technology advances and future trends," *IEEE Trans. Instrum. Meas.*, vol. 53, no. 6, pp. 1497–1501, Dec. 2004.

[2] M. V. Micea, G. N. Carstoiu, L. Ungurean, D. Chiciudean, V. I. Cretu, and V. Groza, "PARSECS: A predictable data communication system for smart sensors and hard real-time applications," *IEEE Trans. Instrum. Meas.*, vol. 59, no. 11, pp. 2968–2981, Nov. 2010.

[3] R. T. Crompton, *Battery Reference Book*, 3rd ed. Oxford, U.K.: Reed Educ. Prof. Publ., Ltd., 2000.

[4] V. Pop, H. J. Bergveld, P. H. L. Notten, and P. P. L. Regtien, "State-of-the-art of battery state-of-charge determination," *Meas. Sci. Technol.*, vol. 16, no. 12, pp. R93–R110, Dec. 2005.

[5] J. C. Viera, M. Gonzalez, J. C. Anton, J. C. Campo, F. J. Ferrero, and M. Valledor, "NiMH vs NiCd batteries under high charging rates," in *Proc. 28th Annu. INTELEC*, Sep. 2006, pp. 1–6.

[6] T. S. Mundra and A. Kumar, "An innovative battery charger for safe charging of NiMH/NiCd batteries," *IEEE Trans. Consum. Electron.*, vol. 53, no. 3, pp. 1044–1052, Aug. 2007.

[7] M. Gonzalez, F. J. Ferrero, J. C. Anton, and M. A. Perez, "Considerations to improve the practical design of universal and full-effective NiCd/NiMH battery fast-chargers," in *Proc. APEC*, Dallas, TX, 1999, pp. 167–173.

[8] J. Diaz, J. A. Martin-Ramos, A. M. Pernia, F. Nuno, and F. F. Linera, "Intelligent and universal fast charger for Ni–Cd and Ni–MH batteries in portable applications," *IEEE Trans. Ind. Electron.*, vol. 51, no. 4, pp. 857–863, Aug. 2004.

[9] S. Sastry, O. Gimdogmus, T. Hartley, and R. Viellette, "Coordinated discharge of a collection of batteries," *J. Power Sources*, vol. 166, no. 1, pp. 284–296, Mar. 2007.

[10] A. B. da Cunha, B. R. de Almeida, and D. C. da Silva, Jr., "Remaining capacity measurement and analysis of alkaline batteries for wireless sensor nodes," *IEEE Trans. Instrum. Meas.*, vol. 58, no. 6, pp. 1816–1822, Jun. 2009.

[11] J. Alvarez, J. Marcos, A. Lago, A. A. Nogueiras, J. Doval, and C. M. Penalver, "A fully digital smart and fast lead–acid battery charge system," in *Proc. 34th Annu. IEEE PESC*, Acapulco, Mexico, Jun. 2003, vol. 2, pp. 913–917.

[12] F. Codeca, S. M. Savaresi, and G. Rizzoni, "On battery state of charge estimation: A new mixed algorithm," in *Proc. IEEE Int. CCA*, Sep. 2008, pp. 102–107.

[13] F. Codeca, S. M. Savaresi, and V. Manzoni, "The mix estimation algorithm for battery state-of-charge estimator—Analysis of the sensitivity to measurement errors," in *Proc. 48th IEEE CDC/CCC*, Shanghai, China, Dec. 2009, pp. 8083–8088.

[14] T. S. Mundra and A. Kumar, "Micro power battery state-of-charge monitor," *IEEE Trans. Consum. Electron.*, vol. 54, no. 2, pp. 623–630, May 2008.

[15] B. Saha, K. Goebel, S. Poll, and J. Christophersen, "Prognostics methods for battery health monitoring using a Bayesian framework," *IEEE Trans. Instrum. Meas.*, vol. 58, no. 2, pp. 291–296, Feb. 2009.

[16] A. Saxena, J. Celaya, E. Balaban, K. Goebel, B. Saha, S. Saha, and M. Schwabacher, "Metrics for evaluating performance of prognostic techniques," in *Proc. Int. Conf. PHM*, Denver, CO, Oct. 2008, pp. 1–17.

[17] H. J. Bergveld, W. S. Kruijt, and P. H. L. Notten, *Battery Management Systems: Design by Modelling*. Norwell, MA: Kluwer, 2002.

[18] R. H. Somogye, "An aging model of Ni–MH batteries for use in hybrid-electric vehicles," MSc. thesis, Dept. Elect. Eng., Ohio State Univ., Columbus, OH, 2004.

[19] R. D. Cioarga, M. V. Micea, B. Ciubotaru, D. Chiuciudean, and D. Stanescu, "CORE-TX: Collective Robotic Environment—The Timisoara Experiment," in *Proc. 3rd Romanian-Hungarian Joint SACI*, Timisoara, Romania, May 2006, pp. 495–506.

[20] A. Pianegiani Boni and D. F. Petri, "Low-power and low-cost implementation of SVMs for smart sensors," *IEEE Trans. Instrum. Meas.*, vol. 56, no. 1, pp. 39–44, Feb. 2007.

[21] *TMP101: Digital Temperature Sensor With I2C Interface*, Texas Instruments, Dallas, TX, Datasheet Rev. G, 2007.

[22] *LPC2131/2/4/6/8 User Manual*, Phillips Semicond., Eindhoven, The Netherlands, Rev. 02, Jul. 2006.

[23] D. Macii and D. Petri, "Accurate software-related average current drain measurements in embedded systems," *IEEE Trans. Instrum. Meas.*, vol. 56, no. 3, pp. 723–730, Jun. 2007.

[24] M. V. Micea, V. Cretu, and V. Groza, "Maximum predictability in signal interactions with HARETICK kernel," *IEEE Trans. Instrum. Meas.*, vol. 55, no. 4, pp. 1317–1330, Aug. 2006.

[25] S. T. Karris, *Numerical Analysis Using Matlab and Excel*, 3rd ed. Fremont, CA: Orchard Publications, 2007.

Mihai Victor Micea (M'02) received the B.Sc., M.Sc., and Ph.D. (*cum laude*) degrees in computer engineering from the "Politehnica" University of Timisoara, Timisoara, Romania, in 1995, 1996, and 2005, respectively.

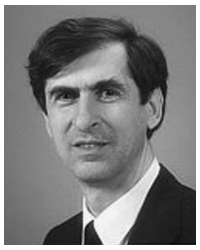
He is currently an Associate Professor with the Department of Computer and Software Engineering, "Politehnica" University of Timisoara, where he coordinates the Digital Signal Processing Laboratories. His research interests include real-time/embedded multiprocessing systems with applications in data acquisition and digital signal processing, robotic collectives, and intelligent sensor networks. He is the author or coauthor of more than 63 technical papers and is the Director or a Team Member of more than 33 R&D grants and contracts.

Lucian Ungurean received the B.Sc. degree in computer engineering from the "Politehnica" University of Timisoara, Timisoara, Romania, in 2009, where he is currently working toward the M.Sc. degree in computer engineering.

His research interests include embedded system communication protocols, power management, and architectures. The research activity is carried out at the Digital Signal Processing Laboratories.

Gabriel N. Cârstoiu received the B.Sc. degree in computer engineering from the "Politehnica" University of Timisoara, Timisoara, Romania, in 2009, where he is currently working toward the M.Sc. degree.

He has been an active Member of the Digital Signal Processing Laboratories R&D Team since 2006. His research interests include real-time embedded systems, advanced power management techniques applied to smart sensors, and real-time communication protocols.



Voicu Groza (M'97–SM'02–F'11) received the Dipl.Eng. degree in computer engineering and the Dr.Eng. degree in electrical engineering from the Polytechnic Institute of Timisoara, Timisoara, Romania.

He was a Professor with the "Politehnica" University of Timisoara, Timisoara. Since 1997, he has been with the University of Ottawa, Ottawa, ON, Canada. His research interests include quantization theory and applications in biomedical measurements, real-time embedded systems, and reconfigurable computers. He is the author or coauthor of more than 150 technical papers.

Dr. Groza is the Chair of the Ottawa Chapter of the IEEE Instrumentation and Measurement Society.