IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 25 November 2024, accepted 14 December 2024, date of publication 18 December 2024,
date of current version 30 December 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3519715

RESEARCH ARTICLE

Computational Cost and Implementation Analysis of a Wavelet-Based Edge Computing Method for Energy-Harvesting Industrial IoT Sensors

JAROMIR KONECNY[1], (Senior Member, IEEE), JAN CHOUTKA[1], RADIM HERCIK[1], JIRI KOZIOREK[1], DANGIRUTIS NAVIKAS[2], DARIUS ANDRIUKAITIS[2], (Member, IEEE), AND MICHAL PRAUZEK[1], (Senior Member, IEEE)

[1]Department of Cybernetics and Biomedical Engineering, VSB—Technical University of Ostrava, 708 00 Ostrava, Czech Republic
[2]Department of Electronics Engineering, Kaunas University of Technology, 44249 Kaunas, Lithuania

Corresponding author: Jaromir Konecny (jaromir.konecny@vsb.cz)

This work was supported in part by the project "Development of Algorithms and Systems for Control, Measurement and Safety Applications X" of Student Grant System, VSB—Technical University of Ostrava (VSB-TU Ostrava), under Project SP2024/021; in part by European Union through the REFRESH—Research Excellence For REgion Sustainability and High-tech Industries project via the Operational Programme Just Transition under Grant CZ.10.03.01/00/22_003/0000048; in part by European Regional Development Fund for the Research Centre of Advanced Mechatronic Systems Project, through the Operational Programme Research, Development and Education, under Grant CZ.02.1.01/0.0/0.0/16_019/0000867; and in part by the Research Council of Lithuania (LMTLT) under Grant S-A-UEI-23-1.

ABSTRACT The rapid advancement of Industrial Internet of Things (IIoT) has heightened the need for efficient data processing and transmission, particularly in energy-constrained environments. This study introduces a novel wavelet-based edge computing methodology designed specifically for low-power IIoT sensors using energy harvesting. Unlike existing implementations that rely on computationally complex instructions, this approach optimizes the wavelet transform (WT) for resource-limited microcontrollers (MCUs) without sacrificing data quality. By leveraging a lightweight assembly-level WT implementation, the proposed solution significantly reduces computational costs and energy consumption. A comprehensive analysis performed on ARM Cortex-M7 MCU on an industrial vibration dataset demonstrates energy savings of assembly language (ASM) up to 87% with discrete wavelet transforms (DWT) and 32.1% with fast wavelet transforms (FWT), compared to C-based implementations. This work is distinct in its ability to dynamically adjust data transmission levels based on available energy, ensuring robust operation in batteryless IIoT environments. Moreover, the method offers flexibility in signal reconstruction, supporting scalable compression ratios and facilitating long-term predictive maintenance applications, making it a pioneering step in sustainable industrial monitoring.

INDEX TERMS Edge computing, energy harvesting, Industrial Internet of Things, implementation optimization, wavelet transform.

I. INTRODUCTION

Predictive maintenance is an essential feature in the concept of Industry 4.0 and enables optimal performance and reliability in industrial equipment [1]. The success of predictive maintenance strongly depends on the information obtained from diagnosed equipment [2]. For this reason, Industry 4.0 smart factories must be fitted with various sensor types (temperature, pressure, optical, vibration, etc.) [3]. Therefore, applying Industrial Internet of Things (IIoT) technologies provides diagnostic data which enables early detection of equipment malfunctions, anomalies [4], reduces maintenance costs, and avoids production losses due to unexpected equipment failures [5]. The main challenge

The associate editor coordinating the review of this manuscript and approving it for publication was Patrizia Livreri.

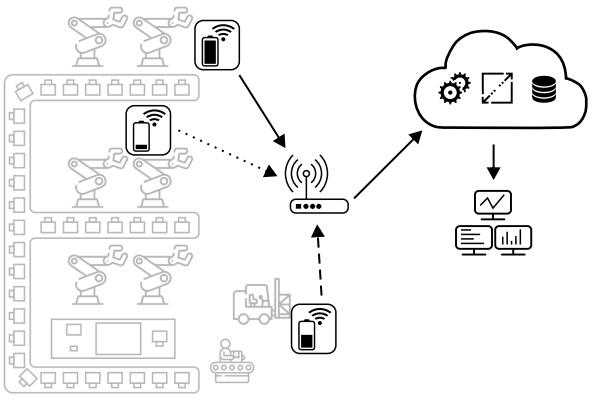

FIGURE 1. IIoT sensors displaying different energy storage states, transmitting collected data to the cloud with varying levels of compression.

in such large-scale measurement processes is in the huge quantity of data generated by the array of sensors in predictive maintenance monitoring systems [6].

Edge Computing (EC) principles provide solutions to several technical challenges, including data reduction and satisfaction of resource constraints in IIoT sensors [7]. EC techniques solve problems with extensive data streams by performing data analysis or compression closer to the source and offloading cloud processing demands to local IIoT devices. The second feature of EC methods is a significant reduction in transmission demands, leading to lower energy consumption, potentially longer operating times, and reduced outages in energy harvesting devices [8].

Batteryless IIoT sensors apply energy harvesting techniques that demonstrate stochastic behavior. The research challenge lies in dynamically controlling the IIoT node's operation to avoid device failure due to energy outages and ensuring that data is immediately online in the cloud. The proposed solution is illustrated in Fig. 1. The goal of this EC method is to transmit data with different compression levels. These levels are represented by various line styles in the scheme. The compression level is selected based on the charge available in the device. This method also allows the transmission of missing information when the stored energy is at a sufficient level. The advantage of this approach is immediate uploading of rough collected data to the cloud and detailed information subsequently transmitted with an acceptable delay [9]. Rough data can be used for immediate monitoring, while detailed data can be stored in the cloud for long-term analysis.

The challenge lies in managing the vast data streams generated by sensors while maintaining energy efficiency in battery-constrained environments. The proposed lightweight Wavelet Transform (WT) method addresses this by reducing computational complexity, optimizing data transmission for edge computing, and minimizing energy consumption through efficient data compression tailored to resource-limited settings.

WT offers unique properties that make it particularly suitable for addressing the challenges of data compression in energy-constrained IIoT environments. Unlike traditional compression methods, WT allows for multi-resolution analysis, enabling the decomposition of signals into both coarse and fine details. This capability supports the dynamic scaling of compression ratios, allowing systems to adapt to fluctuating energy availability—a critical consideration for energy harvesting technologies where harvested energy is distributed unequally. Additionally, WT retains the ability to transmit specific details of the original data for future processing, bridging the gap between immediate data reduction and the need for preserving information integrity. The presented methodology leverages these unique WT properties to propose a lightweight, flexible compression technique that aligns with the requirements of IIoT applications, addressing a critical gap in prior works.

Advancements have been made in Industrial Internet of Things (IIoT) systems and energy-efficient data transmission. However, a critical issue remains. It is challenging to balance energy consumption with computational efficiency. This is especially true for batteryless and energy-harvesting devices. This study bridges that gap by introducing a novel WT-based data compression and transmission method, optimized for microcontrollers in IIoT systems, enabling perpetual operation without compromising data integrity.

Current research lacks methods that offer lossless compression while prioritizing information density during data transmission. Building on the previous work by [9], this study emphasizes the potential of WT. WT enables scalable compression ratios and allows for dynamic signal quality adjustment. It achieves this by progressively transmitting detailed information. While earlier research explored the Energy-Constrained (EC) method, this work presents a novel lightweight WT implementation designed specifically for low-power microcontrollers (MCUs) in IIoT devices.

Although wavelet-based EC methods for IIoT have filled certain gaps in the research, data processing remains challenging in terms of computational power and energy requirements. This article contributes the following:

- A methodology for lightweight WT implementation that avoids computationally expensive instructions.
- A comprehensive analysis comparing the computational costs on an MCU, with and without hardware support for floating-point instructions.
- A case study using an industrial vibration dataset, demonstrating the levels of data loss at various compression rates.

The article is organized into seven sections: Section I introduces the study's innovation; Section II addresses related works and related review; Section III describes the theoretical background for WTs, the use of digital signal processors (DSP) and various compiler settings, and also their effects on compression time; Section IV describes the experiment's

design and setup; Section V presents the results; Section VI discusses the results in the context of the study's innovative approach; Section VII concludes the paper and outlines possible future work.

II. RELATED WORKS

This section addresses the literature review. The related reviews concerning with energy harvesting are introduced and the overview of IIoT solution that are using EC techniques and data compression is analyzed.

The Table 1 presents related reviews to IoT systems powered by energy harvesting techniques. The table presents modern trends closely related to the topic of this study, as it introduces a novel methodology dedicated to energy harvesting systems. Sanislav et al. [10] provide a comprehensive overview of energy harvesting technologies and assess their potential energy outputs. Zeadally et al. [11] offer a study that elaborates on energy harvesting systems for IoT, detailing the specific components that comprise these devices. Ashraf Virk et al. [12] provide a taxonomy and critical review of various energy harvesting techniques to guide engineers in selecting effective solutions. Kucova et al. [13] explore the possibilities of thermoelectric generators as alternative power sources, highlighting the integration of machine learning algorithms to manage and predict energy, which reveals their suitability for low-power applications despite low energy efficiency. Chen et al. [14] address indoor photovoltaic technology for energy harvesting by generating electricity from light energy, a promising technique for IIoT devices. He et al. [15] focus their research on piezoelectric energy harvesters, examining their structural designs, fabrication techniques, performance factors, and strategies for improving efficiency. This study highlights advancements in flexible and stretchable devices for wearable technologies and environmental monitoring.

The related reviews highlight the importance of energy harvesting in IoT systems and the need for efficient solutions to manage available energy effectively. To the best of our knowledge, no study has considered the compression of transmitted data. To fill this gap, this study proposes a methodology for lightweight compression specifically designed for IIoT systems.

EC in combination with remote monitoring is used in many industrial application areas. Table 2 provides a summary of related state-of-the-art (SOTA) studies which have investigated IIoT solutions for EC and compression. Vibration monitoring for structural health monitoring is one of the applications of IIoT. This is discussed in the work by Zhang et al. [17]. Since vibration monitoring typically generates a significant amount of data, compression techniques can be considered to reduce data transmission. The authors in [17] also deal with vibration monitoring and explore an edge-cloud collaborative framework to integrate edge, cloud infrastructures to support efficient artificial intelligence-based data compression and reconstruction with quantized deep compressed sensing (QDCS) in IIoT networks. QDCS network is designed for both linear and nonlinear measurements to improve the performance of industrial data compression and reconstruction.

Other studies have applied EC techniques to maintain image compression. In [18], the authors present a lossless image processing algorithm that uses atomic functions which aims to reliable protection features with convenient image representation. An important parameter to consider in IIoT monitoring is the Age of Information. In [19], the authors demonstrate computational efficient image compression algorithm suitable for resource-constrained industrial IoT applications. In [20], the authors present both lossless and lossy LoRaWAN data compression, which allows for enhanced data throughput without signal downsampling, aiming to optimize network efficiency by minimizing the average Age of Information. Chen et al. [21] introduces a deep reinforcement learning-based data compression method for IoT-generated data in smart railroad management, addressing challenges in storage, processing, and transmission. Serhii and Vasyl [22] presents an innovative approach to improving data compression in IoT systems by using edge technologies and a neural network-based compression algorithm, significantly reducing data transmission volume, enhancing transmission speed, and lowering costs, while simultaneously improving scalability, reducing transmission errors, and strengthening data security within the network.

A review of related works shows that most EC methods are based on data compression or extraction of information without considering the transmission of raw data for future processing. These methods are not fully aligned with energy harvesting technologies, which distribute harvested energy unequally. To fill this gap the methodology for lightweight WT implementation for IIoT is presented. None of previous papers considers the compression which is able to dynamically scale the compression ratio and consequent transmission of the details.

III. METHODS

This section introduces the methods applied in the current study, describing the WTs, compiler optimization and execution of signal instructions.

A. WAVELET TRANSFORM

The proposed approach uses either the discrete wavelet transform (DWT) or fast wavelet transform (FWT) for data compression. The wavelet transform (WT) of an input signal $x(t)$ in $L^2(\mathbb{R})$ space is defined as:

$$WT\{x(t)\} = \frac{1}{\sqrt{|a|}} \cdot \int_{-\infty}^{\infty} x(t) \cdot \overline{\psi\left(\frac{t-b}{a}\right)} \mathrm{d}t, \qquad (1)$$

where $\psi(t)$ is the mother wavelet, $a \in \mathbb{R} \setminus 0$ is the dilation parameter, and $b \in \mathbb{R}$ is the translation parameter.

The DWT/FWT decomposes the input signal into a wavelet spectrum, which consists of approximation coefficients a_m and detail coefficients $d_1, d_2, \ldots, d_m$. These coefficients are

TABLE 1. Related reviews concerning with IoT systems powered by energy harvesting techniques.

Source	Study overview
Sanislav et al. [10]	The analysis of various energy harvesting technologies in IoT, and demonstrates the potential of solar and radio frequency energy sources through two case studies.
Zeadally et al. [11]	Fundamental components of energy harvesting systems for IoT, evaluating various energy resources, storage devices, and control units.
Ashraf Virk et al. [12]	WSN-based energy harvesting technologies for pipeline monitoring systems and offering a taxonomy and critical review of various energy harvesting techniques to guide engineers in selecting effective solutions.
Kucova et al. [13]	Exploring thermoelectric generators as alternative power sources, highlighting the integration of machine learning to manage and predict energy, revealing suitability for low-power applications despite low energy efficiency.
Chen et al. [14]	Indoor photovoltaic technology energy harvesting solution by generating electricity from light energy, typical application scenarios for self-powered IoTs, and a summary and outlook on IPV technology.
He et al. [15]	Piezoelectric energy harvesters, their structural designs, fabrication techniques, performance factors, and efficiency improvement strategies harvesters, highlighting advancements in flexible and stretchable devices for wearable technologies and environmental monitoring.

TABLE 2. Overview of IIoT solutions designed for EC and data compression techniques.

Application domain	Study overview	Operating principles
Vibration monitoring [16]	Exploring the implementation of output-only System Identification models at the extreme edge to alleviate network congestion in large-scale structural monitoring.	Outlines the adaptation and customization of System Identification algorithms, harnessing EC capabilities for monitoring solutions, and presents explicit algorithms and implementation procedures to facilitate this process.
Vibration of CNC machines [17]	Quantized Deep Compressed Sensing Network (QDCS-Net) has an ability to enhance industrial data compression and reconstruction by accommodating both linear and nonlinear measurements.	Customized quantization layers, dual-path structures, and swish activation functions work synergistically to enable efficient low-dimensional embedding and high-precision data reconstruction, especially at higher compression ratios.
Lossless image processing algorithm [18]	The primary innovation of this work lies in the development of a low time complexity lossless image processing algorithm based on atomic functions.	This algorithm offers a combination of compression, data protection, and image representation, enabling its direct application in artificial intelligence technologies such as recognition, classification, retrieval, without the need for preprocessing steps.
Artificial reduction image compression [19]	Blind image compression artifact reduction network designed for industrial IoT systems, optimizing limited storage and computational resources to improve image quality.	Recurrent convolution structure and redesigned recurrent blocks to efficiently reduce compression artifacts from unknown JPEG quality factors.
LoRaWAN data compression [20]	Compression algorithms, both lossless and lossy, are explored to enhance data throughput without signal downsampling, aiming to optimize network efficiency by minimizing the average Age of Information.	Remote estimation techniques, coupled with the concept of Age of Information, are employed to reduce network loads while ensuring accurate signal reconstruction.
Smart railroad [21]	Introduces a deep reinforcement learning (DRL)-based data compression method for IoT-generated data in railroad management, optimizing data transmission and storage while improving compression rates by over 18% in real-time applications.	The method intelligently selects and discards redundant data points while preserving essential information, optimizing data storage and transmission in resource-constrained environments, such as edge servers.
General method for IoT [22]	Presents an innovative approach to improving data compression in IoT systems by using edge technologies and a neural network-based compression algorithm.	optimizing data compression and transmission in IoT networks using advanced techniques such as deep reinforcement learning, neural networks, and edge computing.

computed by convolving the input signal with low-pass filter $\mathbf{H}(n)$ for the approximation and high-pass filter $\mathbf{G}(n)$ for the detail coefficients:

$$a_1(n) = \sum_k \mathbf{H}(k - 2n)x(k), \tag{2}$$

$$d_1(n) = \sum_k \mathbf{G}(k - 2n)x(k). \tag{3}$$

The choice of mother wavelet significantly affects compression performance. A closer match between the wavelet and the signal shape leads to better compression results [23]. The proposed EC approach allows flexibility in data transmission, where detail coefficients can be omitted or added later, enabling gradual signal refinement in resource-constrained environments.

At the first level of decomposition, the output vector consists of both approximation and detail coefficients. If needed, the approximation coefficients can be further decomposed to achieve a deeper level of decomposition. When all approximation and detail coefficients are transmitted, the full signal can be reconstructed without loss. However, if detail coefficients are omitted—such as in low-energy scenarios—they are replaced with zeros, leading to some data loss. The proposed energy-constrained (EC) approach allows these

TABLE 3. GCC compiler optimization levels.

Setting	Description
O0	No speed or size optimisation; Default compiler setting; Fast compiling
O1	Compiler reduces code size and execution time; Compiling is not time consuming
O2	Includes O1; Compiler reduces code size and execution time and optimizes without any time-space trade-off; Increases the execution speed and performance of the compiled code
O3	Includes O2; Includes time-space trade-off optimization tools; Compiling is highly time consuming compared to other optimisations
Os	Includes O1; Performs all functions in O2 except those which increase code size; Mainly focused on output code size

detail coefficients to be transmitted later, enabling signal refinement in the cloud when energy permits.

B. COMPILER OPTIMIZATION

High-level programming languages allow abstraction of the assembly language (ASM) to support more complex operations and efficient design in applications. However, a drawback of the compiling process is that the compiled program is not always as efficient as possible in terms of computational cost.

The compiling process can be executed with various parameters, especially to optimize computational cost or program size. Optimization tools can be used to reduce the number of steps and cycles or increase the number of cycles to minimize the number of instructions stored in program memory.

Table 3 shows the optimization variants of the GNU Compiler Collection (GCC). The GCC compiler's default setting is O0, which does not apply any optimization procedure. The advantage of this setting is fast compiling. The next settings are O1 and O2, both which reduce the size of the code and its computational costs. Compared to O1, setting O2 use additional tools to reduce size and execution time, but the compiling process is longer. Setting 03 achieves the most comprehensive optimization of execution time, using time-space optimization tools in addition to the tools of setting O2. The final optimization option is Os, which primarily decreases code size.

C. SIGNAL INSTRUCTION EXECUTION

WTs are computationally intensive operations that rely heavily on convolutions, requiring a significant number of multiply–accumulate (MAC) operations [24]. Efficient processing of MAC operations is critical to achieving high performance in applications involving WTs. Modern hardware solutions, such as digital signal controllers (DSCs), digital signal processors (DSPs), and system-on-chip (SoC) platforms, are specifically designed to optimize these operations. These systems often include dedicated instruction sets and hardware acceleration features for signal processing tasks, enabling faster execution of MAC operations. WTs can be implemented as two nested loops of MAC operations,

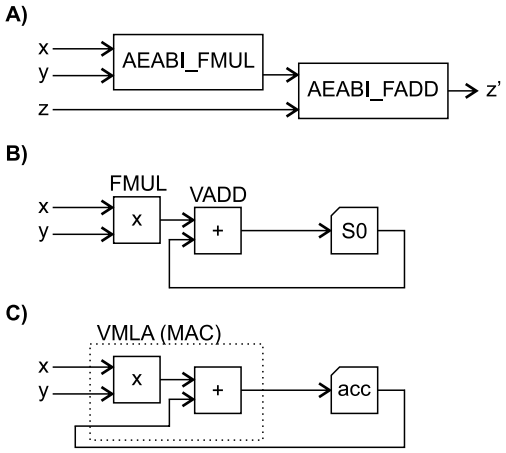

FIGURE 2. Compilation of the floating-point operation `z += x * y;` in three scenarios: A) Compiled C program without FPU; B) Compiled C program with FPU but without optimization; C) Compiled C program with O2 (or higher) optimization enabled.

where the inner loop performs the convolution, and the outer loop iterates over the signal segments, therefore, the efficient MAC instruction processing is crucial. Additionally, since WTs typically operate on real numbers, floating-point arithmetic is preferred for maintaining numerical accuracy and precision. The choice of hardware significantly affects overall performance. Its ability to efficiently handle floating-point MAC operations is crucial. This is especially important for applications that require real-time or large-scale signal processing.

Fig. 2 illustrates three variations of compiling the C language expression `float z += x * y;`. Fig. 2A depicts a block diagram of disassembled code when signal processing support is not present. The compiler uses two libgcc subroutines (AEABI_FMUL and AEABI_FADD). Each subroutine contains several instructions for processing in the loop. This approach is the most time consuming. Fig. 2B depicts the disassembly of compiled C code without optimization. Floating point instructions are used, and the results are accumulated into register S0 (generally SX). Fig. 2C depicts the disassembled compiled C code with O2 (or greater) optimization. The optimization routine recognizes that two floating point instructions can be replaced with VMLA, a single MAC instruction. This approach is also used in ASM implementations.

IV. EXPERIMENT

This section describes the DWT and FWT algorithms optimized in both C and ASM, the optimized convolution calculation, and the experimental procedure for performance analysis. WT performance was analyzed at the first level of decomposition; higher levels are linear in time and were calculated from first decomposition level values.

A. IMPLEMENTATION

This subsection describes the FWT and DWT algorithms optimized in both C and ASM. The source

codes of the proposed solution is available on GitHub: (https://github.com/CHO0178/FastWaveletTransform).

The implementation was debugged and verified against the FWT and DWT implementations available in Matlab software. The algorithms produced exactly the same results. To evaluate the correctness of the implementation, both white-box and black-box testing methods were used. Additionally, the implementation was validated according to the mathematical definition of the WT.

The mathematical definition of a WT is a convolution. Processing a convolution is generally time consuming, however the mother wavelet is a sparse matrix and allows this calculation to be optimized.

Algorithm 1 Wavelet Decomposition Algorithm for Execution in C

if FWT **then**
  out_len $= \frac{\text{in\_len}}{2}$
else if DWT **then**
  out_len $= \frac{\text{in\_len}}{2} + \frac{\text{WL\_len}}{2} - 1$
end if
for $i_d = 0$ **to** out_len $- 1$ **do**
  low_out$[i_d] = 0$, hi_out$[i_d] = 0$
  for $i_{WL} = 0$ **to** WL_len $- 1$ **do**
    if FWT **then**
      addr $= (2 \cdot i_d + i_{WL})$ mod in_len
    else if DWT **then**
      addr $=$ dS$(2 -$ WL_len $+ 2 \cdot i_d + i_{WL},$ in_len$)$
    end if
    low_out$[i_d] +=$ data[addr] $\cdot$ **H**$[i_{WL}]$
    hi_out$[i_d] +=$ data[addr] $\cdot$ **G**$[i_{WL}]$
  end for
end for

Algorithm 1 describes the execution in C of the DWT and FWT. It produces output data separated into two vectors (low_{out} and hi_{out}). To calculate a convolution, the wavelet decomposition algorithm applies two nested loops. To optimize computational costs, the inner loop iteration count is limited to a number of non-zero coefficients of the mother wavelet matrix. However, the appropriate non-zero coefficient address must be calculated differently because they are different in each algorithm (DWT and FWT).

The idea behind optimization is for the convolution to be calculated only with non-zero mother wavelet coefficients. The indices of these coefficients are generally not consecutive and should therefore be individually calculated but not be time consuming. For the FWT variant, the address is calculated from the equation

$$addr = (2 \cdot i_{\mathrm{d}} + i_{\mathrm{WL}}) \,\%\, in_{len}, \tag{4}$$

where addr is the input data address, i_{d} is the output data iteration i_{WL} mother wavelet iteration, and in_{len} is the input data length.

The DWT algorithm requires a more complex input data address calculation process than the FWT, thus significantly affecting computational costs. For the DWT calculation, extension of the input data is also required:

$$\mathrm{d}data_{ext} = \left\{ \begin{array}{c} \mathrm{D}_{\mathrm{W}L_{len}-3}, \ldots, \mathrm{D}_0, \\ \mathrm{D}_0, \ldots, \mathrm{D}_{N-1}, \\ \mathrm{D}_{N-1}, \ldots, \mathrm{D}_{N-\mathrm{W}L_{len}+2} \end{array} \right\}, \tag{5}$$

where $\mathrm{W}L_{len}$ is the mother wavelet length, D_i are input data, and N is the data length. For mother wavelets with two coefficients, data extension is not required. The result of data extension is that the input data are mirrored both before and after input. Data extension can be performed by copying the input data or virtually by extending the input address space which refers to the original input data. The calculation which converts the virtual data indices (negative indices and indices greater than the data length) to physical indices within the input data is expressed as

$$\mathrm{d}S(pos, \mathrm{d}l) = \left| \left(\left\lfloor \frac{pos}{\mathrm{d}l} \right\rfloor \,\%\, 2 \right) \cdot (\mathrm{d}l - 1) - pos \,\%\, \mathrm{d}l \right|, \tag{6}$$

where pos is the virtual position, $\mathrm{d}l$ is the input data length, and the dS function returns a physical address.

Algorithm 2 Wavelet decomposition Algorithm Executed in ASM

Initialize (and save) registers
Initialize $P =$ WL_len $- 3$
Initialize addr $= P$
for $i_d = 0$ **to** out_len $- 1$ **do**
  Calculate offset P address from i_d
  for $i_{\mathrm{WL}} = 0$ **to** WL_len $- 1$ **do**
    Clear FPU
    Calculate data address from i_d, i_{WL}, and P
    Copy current input sample into FPU
    Copy current **H** value and **G** value into FPU
    Perform FPU MAC instruction into $\mathrm{Acc_A}$
    Perform FPU MAC instruction into $\mathrm{Acc_B}$
  end for
  Copy $\mathrm{Acc_A}$ from FPU to low_out$[i_d]$
  Copy $\mathrm{Acc_B}$ from FPU to hi_out$[i_d]$
end for
(restore registers)

This subsection describes the DWT and FWT algorithms executed in ASM. Algorithm 2 describes wavelet decomposition (in ASM), and as with Algorithm 1 executed in C, calculates a convolution with non-zero mother wavelet coefficients only. Execution in ASM also uses an floating point unit (FPU) and thus MAC instructions. However, Cortex M7 does not support the modulo instruction, therefore (6) cannot be used, and the addresses should be calculated in another way. In addition to the modulo function, address calculation involves time consuming subtraction, addition, multiplication and division operations, and therefore another type of instruction (e.g., shift instruction) is useful.

Calculation of the input data address and the helper variable offset address P are discussed below. Data extension

for ASM execution is expressed by following:

$$data_{ext} = \left\{ \begin{array}{c} data_L : \{D_{WL_{len}-3}, \ldots, D_0, \} \\ data : \{D_0, \ldots, D_{N-1}, \} \\ data_R : \{D_{N-1}, \ldots, D_{N-WL_{len}+2}\} \end{array} \right\}, \quad (7)$$

where $data_L$ is the input data left extension and $data_R$ is the input data right extension.

Algorithm 3 Calculation of the Offset P

```
if P = 1 then
    P = 0
else if i_d ≤ WL_len/2 − 1 then
    P = P − 2
else
    P = P + 2
end if
```

The WT decomposition algorithm generally has four cases according to the mother wavelet length and algorithm type (FWT and DWT):

1) For FWT with a mother wavelet length equal to two, the offset P is initiated at zero and incremented by two in each iteration. The address (addr) starts at P in the outer loop iteration and is incremented by one in the inner loop iteration.
2) For DWT with a mother wavelet length equal to two, the calculation is the as same as in the first case.
3) For FWT with a mother wavelet length greater than two, the calculation is the same as the first case, but when the address exceeds the data length, it is reset to zero and the algorithm continues.
4) For DWT with a mother wavelet length greater than two, the offset P is calculated according to Algorithm 3; calculation of the address is described below.

To calculate an address, it is necessary to determine the membership of the P offset in the data set:

$$\begin{array}{ll} P \text{ is } odd : & data_L \\ P \text{ is } even \text{ and } in_{len} - WL_{len} + 2 < i_d : & data_R \\ Else : & data. \end{array} \quad (8)$$

The edge number E is then calculated according to the equation

$$E = \left\{ \begin{array}{ll} P \in data_L : & WL_{len} - (i_d + 1) \cdot 2 \\ P \in data_R : & WL_{len} - \left(i_d + 1 - \frac{in_{len}}{2} \right) \cdot 2 \end{array} \right. . \quad (9)$$

Finally, an address (addr) is calculated from the rules

$$\begin{array}{l} addr[i_{WL} = 0] = 0 \\ P \in data_L \text{ and } E = i_{WL} : addr[i_{WL}] = addr[i_{WL-1}] \\ P \in data_L \text{ and } E > i_{WL} : addr[i_{WL}] = addr[i_{WL-1}] + 1 \\ P \in data_L \text{ and } E < i_{WL} : addr[i_{WL}] = addr[i_{WL-1}] - 1 \\ P \in data_R \text{ and } E = i_{WL} : addr[i_{WL}] = addr[i_{WL-1}] \end{array}$$

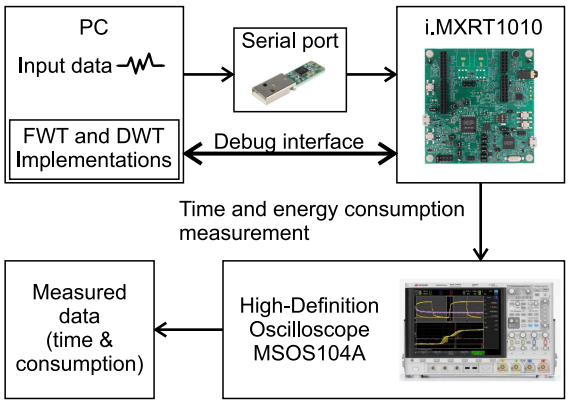

FIGURE 3. Block diagram of the experimental setup: the i.MXRT1010 platform executing wavelet-based computations.

$$\begin{array}{l} P \in data_R \text{ and } E > i_{WL} : addr[i_{WL}] = addr[i_{WL-1}] - 1 \\ P \in data_R \text{ and } E < i_{WL} : addr[i_{WL}] = addr[i_{WL-1}] + 1 \\ P \in data : addr[i_{WL}] = addr[i_{WL-1}] + 1 \end{array} \quad (10)$$

Determining the address is a crucial calculation that significantly affects the entire algorithm's performance. The algorithm therefore uses only primitive operations instead of complex instructions such as division, modulo or multiplication. Multiplication and division by two is therefore executed by shifting the register, which is also much faster than multiplication or division. Frequently repeated expressions are also processed only once and stored in a temporary register.

B. EXPERIMENTAL SETUP

This section describes the experimental setup for comparing execution in ASM and C. The experiment uses the 'arm-none-eabi-gcc' compiler, which is part of the MIMXRT1011xxxxx Software Development Kit (SDK). This compiler configuration was selected to meet the requirements of ARM MCU and processors. The compilation process is managed through the 'gnu make builder.' The toolchain used in the subsequent experiments is 'NXPMCU Tools,' specifically designed to provide dedicated support for NXP's ARM MCU platforms. In addition, no operating system was employed, and MCUXpresso IDE v11.6.1_8255 was used.

Fig. 3 illustrates the experimental setup. All WT processes were performed on the NXP MIMXRT1010-EVK hardware development kit, which includes an ARM Cortex-M7 MCU clocked at 500 Mhz, 128 kB RAM, and 64 kB flash memory. This type of MCU supports signal instruction processing and includes MAC instructions on the FPU. This platform allowed testing of all the scenarios described in this study.

All versions of the FWT and DWT were tested in binary executable code prepared on a computer and flashed to the target i.MXRT1010 platform through a debugging interface. The flashed programs operated with input data copied to RAM through a serial port. The computational cost performance of the FWT and DWT algorithms was

independent of the data characteristics, therefore synthetic data was sufficient for application in this setup.

The testing procedure also measured time and energy consumption for the purposes of estimating the effect on computational cost. All measurements were performed with a high-definition, 10-bit Keysight MSOS104A oscilloscope with voltage and double-ranged current probes. Time was measured by changing the logical level on a general purpose I/O peripheral. The presented algorithms are fully deterministic and it is expected that each run provides the same results. To increase measurement accuracy, the program was run for 100,000 iterations, and the results were calculated according to the average time per program run, which exactly corresponds with the actual one run time. Current and voltage were measured with the oscilloscope in all tested scenarios. Energy consumption was calculated by multiplying voltage and current over the measurement period.

V. RESULTS

This section provides an analysis of the computational costs of the WTs executed in both C and ASM, comparing two types of WT algorithms (FWT and DWT) which used four mother wavelets and various non-zero coefficients. Ten variants of C language compiling parameters were also incorporated. Furthermore, this section presents a comparison of data loss analysis performed on real-life industrial data.

Table 4 compares the computation times of the FWT and DWT algorithms according to wavelet type. Four sections indicate the mother wavelet type. Each mother wavelet was evaluated with three variants of input data length (in_{len}). The columns represent the execution type and compiler settings. C language was compiled with a GCC compiler with FPU support, no FPU support, compiler optimization (O1, O2, O3 and OS), and no compiler optimization (O0).

Fig. 4 graphs the executions in C and ASM of the FWT and DWT algorithms. The coloured bars represent variants of these executions (ASM, C language, with an FPU, without an FPU), and hashed bars represent input data length. Execution in C language is presented without compiler optimization for the purpose of comparing the computational costs of the default algorithms without any additional optimizations.

The results indicate that execution in ASM achieved faster processing times than any configuration compiled in C. In terms of performance according to the algorithm type, processing time with ASM by the DWT was approximately 33% longer than with the FWT. The results also indicate that WT complexity was strongly dependent on the mother wavelet's size and input data length. The reasons for the higher computational cost of the DWT algorithm are straightforward: calculating the address is more complex with the DWT than with the FWT, and data expansion results in a longer output vector and a larger number of concurrent outer loop iterations. The only exception was the Haar wavelet with size two, where no data expansion was performed, achieving similar results with the DWT and FWT both executed in ASM. Execution in C, however, performed with a higher computational cost because of its different method of calculating addresses.

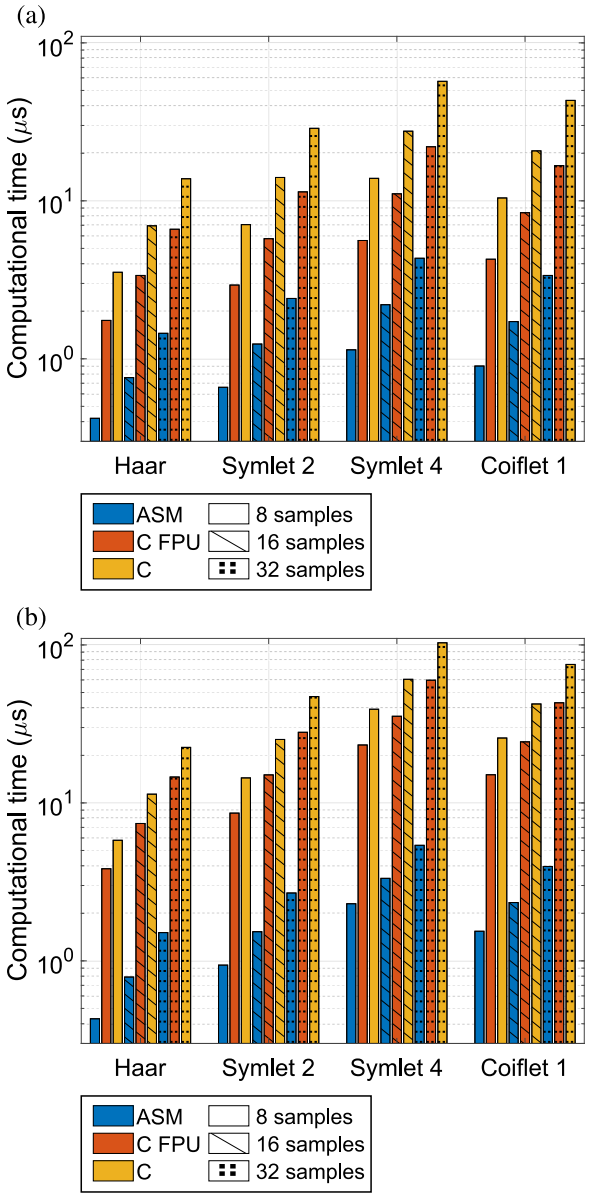

FIGURE 4. Comparison of computational time across three implementations for wavelets using ASM, C with FPU (O0 optimization), and C without FPU (O0 optimization): (a) FWT and (b) DWT.

Computational cost should have a linear dependence on the wavelet length and input data length. However, before the convolution is calculated, an initialization is performed to prepare the data, and therefore the results for various wavelet lengths and input data lengths are not strictly linear. The results indicate four types of mother wavelet according to wavelet length. The first three (Haar, Symlet 2 and Symlet 4) had wavelength powers of two, which potentially simplifies WT calculations. However, some mother wavelets did not fulfil this assumption, for example Coiflet 1. The WT method proposed here therefore allows the use of wavelets with

TABLE 4. Comparison of overall computation time (μs) (↓) of the FWT and DWT algorithms according to wavelet type.

FWT												
				With FPU					Without FPU			
Wavelet	WL_{len}	in_{len}	ASM	C O0	C O1	C O2	C O3	C OS	C O0	C O1	C O2	C O3	C OS
Haar	2	8	0.42	1.75	0.60	0.57	0.57	0.67	3.53	2.49	2.70	2.67	2.48
		16	0.76	3.37	1.08	1.03	1.03	1.21	6.95	4.87	5.30	5.25	4.84
		32	1.45	6.61	2.15	1.99	1.99	2.35	13.80	9.63	10.51	10.41	9.59
Symlet 2	4	8	0.66	2.93	0.95	0.92	0.90	1.07	7.07	5.02	5.48	5.41	5.04
		16	1.24	5.76	1.78	1.75	1.72	2.03	14.06	9.95	10.89	10.74	10.00
		32	2.41	11.40	3.44	3.41	3.35	3.94	28.78	20.56	22.43	22.14	20.68
Symlet 4	8	8	1.14	5.61	1.70	1.71	1.67	1.93	13.88	9.79	10.74	10.60	9.89
		16	2.20	11.08	3.27	3.30	3.24	3.74	27.60	19.43	21.36	21.08	19.65
		32	4.33	22.00	6.40	6.50	6.37	7.35	57.10	40.81	44.46	43.91	41.25
Coiflet 1	6	8	0.90	4.27	1.32	1.31	1.29	1.50	10.43	7.37	8.00	7.90	7.43
		16	1.72	8.41	2.52	2.53	2.48	2.88	20.72	14.61	15.91	15.70	14.75
		32	3.37	16.70	4.92	4.96	4.86	5.65	43.24	31.04	33.52	33.10	31.32
DWT													
				With FPU					Without FPU				
Wavelet	WL_{len}	in_{len}	ASM	C O0	C O1	C O2	C O3	C OS	C O0	C O1	C O2	C O3	C OS
Haar	2	8	0.43	3.83	3.50	2.77	2.77	2.79	5.80	4.55	13.49	13.49	4.64
		16	0.79	7.41	5.49	5.30	5.30	5.34	11.36	8.86	39.28	39.30	9.05
		32	1.51	14.59	10.81	10.37	10.37	10.44	22.47	17.48	129.14	129.00	17.87
Symlet 2	4	8	0.94	8.62	6.27	6.06	6.06	6.15	14.40	11.34	55.73	55.42	11.49
		16	1.53	15.05	10.82	10.47	10.47	10.62	25.22	19.86	90.20	90.20	20.26
		32	2.69	28.00	19.99	19.35	19.35	19.63	47.12	36.85	170.98	170.69	37.62
Symlet 4	8	8	2.30	23.27	16.72	16.38	16.38	16.56	39.18	30.96	93.48	93.49	31.59
		16	3.33	35.37	25.08	24.57	24.57	24.84	60.61	47.68	180.58	180.62	48.69
		32	5.39	59.82	42.05	41.19	41.19	41.66	103.22	80.80	355.82	356.08	82.57
Coiflet 1	6	8	1.54	15.08	10.88	10.61	10.61	10.74	25.76	20.50	86.23	86.23	20.83
		16	2.34	24.36	17.34	16.92	16.92	17.13	42.31	33.42	115.82	116.82	34.09
		32	3.96	43.05	30.41	29.67	29.67	30.04	75.28	59.21	245.02	243.90	60.49

various lengths. As mentioned in Section III-A, selecting the appropriate mother wavelet is crucial. From the point of view of the proposed approach, only wavelet length and input data length impact the computational cost.

Fig. 5 shows the average performance improvements of various C language compiler optimization levels. The biggest performance improvement was achieved by the FWT with FPU optimization: the compiler recognized that MAC instructions can be applied instead of two floating point operations, which are used by compilers with no optimization. The advanced compiling levels (O2 and O3) also increased performance, although it was minor compared to O1.

The optimization process was able to increase the computational performance in three cases: both FWT configurations and the DWT with an FPU. The DWT configured without an FPU and O2 and O3 produced a decrease in performance, likely caused by the time-space trade-off optimization tools, which reduce code size at the expense of speed.

The performance optimizations of the FWT and DWT algorithms configured with an FPU clearly indicate a significant difference. In Algorithm 1 described in Section IV, address calculation for the FWT is significantly simpler than the calculation for the DWT and has a major benefit on computational performance.

Fig. 6 compares the best performing implementations, which were the FWT and DWT executed in ASM with FPU support and optimization with O3. Of these, the fastest configuration was FWT executed in ASM; the DWT

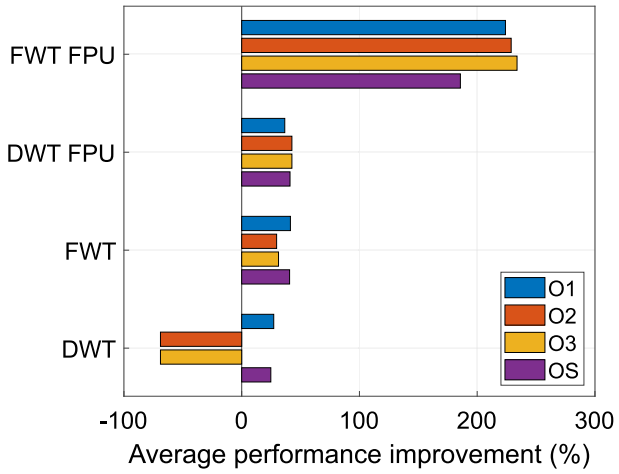

FIGURE 5. Comparison of average performance improvements across various levels of compiler optimization. A 100% performance improvement indicates the optimized code runs twice as fast as unoptimized code, while a −50% performance change means the optimized code runs at half the speed of the unoptimized code.

ASM and FWT C configured with an FPU also achieved satisfactory results. However, the DWT executed in C had much higher computational costs than the other three configurations.

Table 5 provides a breakdown of the experimentally measured energy consumption and computational costs of the FWT and DWT calculations in an Arm Cortex-M7 MCU. The table shows the estimated power consumption for executions

TABLE 5. Energy consumption analysis for executions in ASM and C, with/without an FPU and O3 compiler optimization for various wavelets and input data lengths.

		FWT					DWT				
		Energy consumption (µJ) (↓)			ASM energy saved (%) (↑)		Energy consumption (µJ) (↓)			ASM energy saved (%) (↑)	
Wavelet	WL_{len}	ASM	C-O3-FPU	C-O3	C-O3-FPU	C-O3	ASM	C-O3-FPU	C-O3	C-O3-FPU	C-O3
Haar	8	0.095	0.129	0.777	26.7	83.3	0.098	0.626	3.923	84.4	84.1
	16	0.172	0.232	1.525	25.8	84.8	0.179	1.197	11.426	85.0	89.5
	32	0.328	0.450	3.027	27.2	85.1	0.341	2.344	37.504	85.4	93.8
Symlet 2	8	0.149	0.203	1.572	26.7	87.1	0.213	1.371	16.113	84.4	91.5
	16	0.280	0.388	3.123	27.8	87.6	0.345	2.367	26.223	85.4	91.0
	32	0.545	0.756	6.437	28.0	88.2	0.609	4.374	49.625	86.1	91.2
Symlet 4	8	0.258	0.378	3.082	31.7	87.8	0.519	3.703	27.179	86.0	86.4
	16	0.497	0.732	6.128	32.1	88.0	0.752	5.555	52.512	86.5	89.4
	32	0.979	1.441	12.766	32.1	88.7	1.218	9.312	103.523	87.0	91.0
Coiflet 1	8	0.203	0.290	2.297	30.0	87.4	0.347	2.399	25.070	85.5	90.4
	16	0.389	0.560	4.564	30.6	87.7	0.530	3.824	34.0	86.1	88.7
	32	0.762	1.098	9.623	30.6	88.6	0.895	6.706	70.909	86.7	90.5

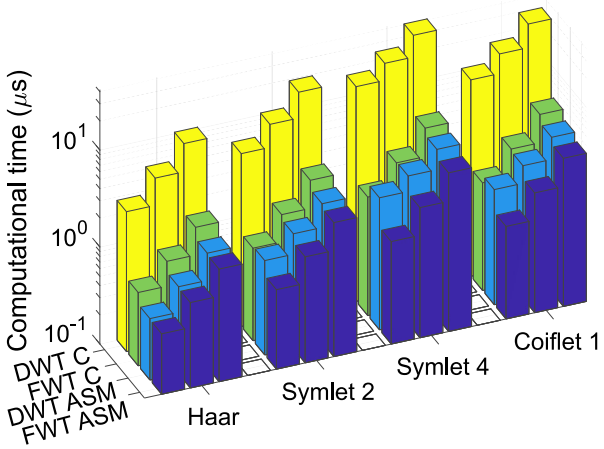

FIGURE 6. Comparison of computation time for different wavelet implementations: FWT in ASM, DWT in ASM, FWT in C with FPU (O3 optimization), and DWT in C with FPU (O3 optimization).

in ASM and C optimized with O3, with/without an FPU, by the FWT and DWT algorithms. Executed in ASM, FWT saved up to 32.1% energy and DWT even up to 87%. The difference in energy consumption between the FWT algorithm executed in ASM with an FPU and the FWT algorithm executed in C with optimization but no FPU was 88.7%; for the DWT algorithm, this difference was 93.8%. Fig. 7 presents a comparison of energy consumption between the ASM implementation and the C implementation, which uses the FPU and is optimized with O3 optimization. It is evident that the ASM implementation achieves significantly lower power consumption, particularly for the DWT implementation.

Table 6 presents a comprehensive analysis of the signal quality at different compression levels in comparison to the original, uncompressed signal. The signal was decomposed (and composed) in 32-sample frames. The compression level, denoted as L, corresponds to the decomposition level achieved using the WT. At level zero, the signal is decomposed using WT, and all coefficients are transmitted without any loss of data. At level one, the detailed coefficients (d_1) are excluded from transmission, while at level two, both d_1 and d_2 are omitted. For level three, only the approximation coefficients (a_3) are transmitted. It is worth noting that data decomposed using the FWT algorithm maintains the same length as the original data. However, in the case of DWT, the length of decomposed data can exceed that of the original data when the choice of mother wavelet is inappropriate. This makes DWT-based compression unsuitable for uncompressed signal transmission since it necessitates the transmission of a higher volume of data than the original signal. Compression variants unsuitable for this purpose are indicated in italic style.

In this study, an accelerometer dataset acquired from real-life measurements on an industrial washing machine was employed as the input data. Overall, the Haar wavelet emerges as the most suitable choice for the analysis of this particular dataset. Results show, that in case of haar wavelet, there is no discernible distinction in terms of data loss between the FWT and the DWT. However, the results also demonstrated that DWT exhibits a more favorable performance, characterized by lower Root Mean Square Error (RMSE) and a higher Goodness of Fit (GoF) when employing the db2, db4, and coif1 wavelets. Notably, the diminished data loss for these wavelets in conjunction with DWT is associated with the decomposition process, which yields larger data sizes and consequently mitigates data loss.

VI. DISCUSSION

The current study introduces several areas for discussion. The first covers the use of WTs in data transmission for predictive maintenance purposes; the second introduces the key parameters of the proposed solution; the last deals with a comparative performance analysis of the proposed methods executed in C and ASM and their performance potential in MCUs.

Industrial monitoring using IIoT, for example for predictive maintenance purposes, needs high quality data to ensure a sufficiently reliable output for predicting potential failure. Deterioration in data quality adversely impacts this expected reliability, therefore lossless mechanisms instead

TABLE 6. Comparison of compressed signal quality at a specific level (L) with the uncompressed (original) signal. Italics indicate unsuitable variants where the data length after the WT exceeds that of the original signal.

FWT				DWT			
Wavelet	L	RMSE (↓)	GoF (↑)	Wavelet	L	RMSE (↓)	GoF (↑)
Haar	0	0.0000	1.0000	Haar	0	0.0000	1.0000
	1	0.8635	0.8745		1	0.8635	0.8745
	2	1.3886	0.7981		2	1.3886	0.7981
	3	1.7920	0.7395		3	1.7920	0.7395
Symlet 2	0	0.0000	1.0000	*Symlet 2*	*0*	*0.0000*	*1.0000*
	1	0.9097	0.8678		*1*	*0.7671*	*0.8885*
	2	1.5178	0.7793		*2*	*1.2614*	*0.8166*
	3	2.0373	0.7038		*3*	*1.5695*	*0.7718*
Symlet 4	0	0.0000	1.0000	*Symlet 4*	*0*	*0.0000*	*1.0000*
	1	0.9879	0.8564		*1*	*0.6610*	*0.9039*
	2	1.5973	0.7678		*2*	*1.2715*	*0.8151*
	3	2.0396	0.7035		*3*	*1.5138*	*0.7799*
Coiflet 1	0	0.0000	1.0000	*Coiflet 1*	*0*	*0.0000*	*1.0000*
	1	0.9307	0.8647		*1*	*0.7357*	*0.8930*
	2	1.5166	0.7795		*2*	*1.2719*	*0.8151*
	3	2.1640	0.6854		*3*	*1.5351*	*0.7768*

TABLE 7. Comparison of key parameters between the proposed solution and SOTA EC methods.

Parameter	Proposed solution	SOTA EC solutions
IIoT node price	Lower	Higher
Cloud service price	Higher	Lower
Adaptability to predictive maintenance models	Yes, in the cloud	IIoT node firmware must be upgraded
Dynamic update in the cloud	Yes	No
Suitable for energy harvesting	Yes	No

of lossy compression are more appropriate solutions for data transmission. A promising solution is the use of WTs for compression scaling. WTs allow rough data to be transmitted with a certain level of data loss, followed by detailed data once sufficient energy is available for that process. Rough data and detailed data together contain the original information without any compromise in its quality.

Table 7 compares key parameters of the proposed solution with state-of-the-art (SOTA) energy consumption (EC) methods. The proposed approach processes data in the cloud, allowing the use of low-cost microcontrollers (MCUs) at the measurement nodes [25]. There is a trade-off between the node price and the quality of the collected data. However, this solution offers several advantages: all data are stored in the cloud, which simplifies the adaptation of next-generation prediction models and facilitates long-term industrial process analysis [26]. Additionally, the low computational cost at the node enables the use of energy-efficient devices and supports energy harvesting [27]. The parameters in the table show that the proposed approach allows for optimal transmission control, significantly reducing the power consumption of IIoT nodes. While reducing the amount of transmitted data during energy shortages may impact the immediate quality of the information, it can be enhanced later when sufficient energy is available. Moreover, the transmission of approximation coefficients ensures data immediacy, even under limited energy conditions.

For IIoT nodes, it is possible to use either low-cost MCUs or MCUs that support signal processing instructions. Naturally, MCUs lacking support for signal processing instructions handle the wavelet transform (WT) less efficiently than those with such support. Executing data processing in assembly (ASM) with signal processing instructions or in C with compiler optimizations are both viable options. However, for a low-cost MCU without signal processing support, ASM is not ideal unless a simplified algorithm is employed, such as one using fixed-point numbers. While fixed-point calculations introduce inaccuracies, execution in C allows for concurrent optimization of both the algorithm and the compiler.

Power for autonomous devices is supplied by energy harvesting systems that produce limited energy. Therefore, a critical factor in autonomous operation is energy consumption. To ensure reliable functioning, the device must be energy neutral, meaning that the energy harvested must meet or exceed the energy consumed. Efficient data processing significantly contributes to minimizing energy consumption. In this experiment, GCC compiler optimization reduced computation time by up to 71% for the fast wavelet transform (FWT) and 31% for the discrete wavelet transform (DWT). Additionally, ASM execution saved 31% more energy with FWT and 87% more with DWT compared to the fully optimized GCC-compiled code.

The case study presents the results of data compression applied to an industrial vibration dataset, where compression using the Haar wavelet method produced the lowest data loss. These outcomes depend heavily on the characteristics of the dataset. The cluster size (32 samples in this case) can be adjusted based on data characteristics, sampling frequency, and required transmission periods. Larger clusters enable higher levels of decomposition, resulting in greater compression efficiency.

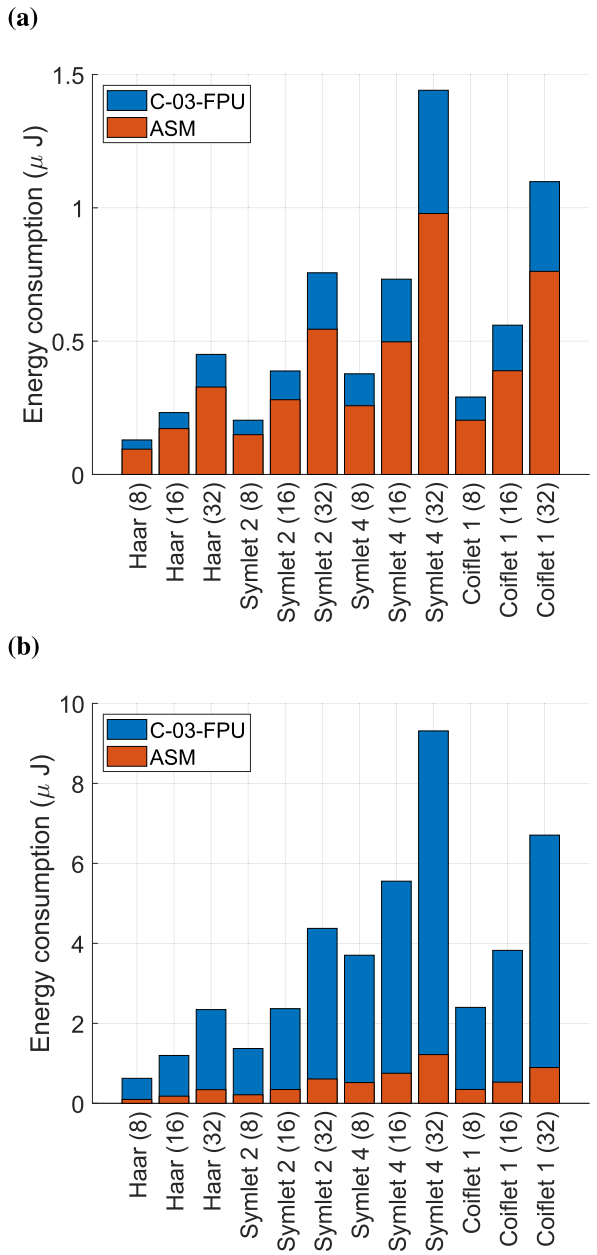

FIGURE 7. Comparison of energy consumption between ASM and C with FPU (O3 optimization) for: (a) FWT and (b) DWT.

Our method demonstrates substantial reductions in energy consumption, with savings of up to 87% when using DWT and 32.1% with FWT compared to traditional C-based implementations. These results suggest that industries adopting this approach could extend the lifespan of IIoT sensors, reduce operational costs, and enhance system reliability, particularly in energy-sensitive applications like predictive maintenance.The main difference between FWT and DWT lies in computational efficiency and flexibility. FWT is an optimized, faster implementation of DWT, designed to reduce computational complexity. However, this speed can come at the cost of accuracy and flexibility. DWT, on the other hand, offers more precise decomposition of signals, allowing for a finer analysis across multiple decomposition levels, and provides greater flexibility in choosing wavelet functions, which can be crucial for handling specific signal characteristics. This makes DWT better suited for applications requiring detailed signal analysis, such as noise reduction or pattern recognition, where precision is critical, whereas FWT prioritizes speed over detail.

While deep learning-based methods may achieve superior compression ratios, they come with significantly higher computational requirements, making them less suitable for the low-power environments targeted in this study. For instance, Zonzini et al. [16] propose an Autoregressive Model with Moving Average that processes 2565 samples in 129 seconds on an STM32L5 platform. Zhang et al. [17] introduce a quantized deep compressed sensing technique, which compresses 20,000 samples in 14.02 seconds with a compression ratio of 1/128. Li et al. [19] present an image compression algorithm that compresses a 128 × 128 RGB image in approximately 50 milliseconds using a GPU. It is clear that our proposed solution offers a distinct advantage in terms of computational cost.

In practical terms, the energy savings demonstrated by our method could have a significant impact on industries that rely on IIoT for long-term monitoring, such as manufacturing, energy, and transportation. By reducing the frequency of battery replacements or the need for external power sources, industries can lower maintenance costs and improve operational sustainability. Furthermore, the method's adaptability to varying energy inputs and flexible data transmission make it ideal for applications requiring continuous data monitoring, even with intermittent energy availability—such as in remote environmental monitoring or smart city infrastructure.

The trade-offs between data compression levels and resulting data loss have significant practical implications, particularly in predictive maintenance applications. Higher compression levels reduce the data volume transmitted, which is crucial for energy-constrained IoT devices like those powered by energy-harvesting systems. However, increased compression often introduces data loss, potentially degrading the accuracy of predictive algorithms. In predictive maintenance, this could result in delayed or missed detection of critical fault conditions, compromising system reliability and leading to unplanned downtime or safety hazards.

As highlighted in the referenced study, the proposed method balances this trade-off by dynamically adjusting the compression level based on available energy and data priority. The method prioritizes the transmission of high-information-density approximate coefficients, ensuring that essential data reaches the cloud with minimal latency. Detailed coefficients, which refine the accuracy of predictive models, are transmitted later when energy conditions improve, mitigating the risk of significant information loss. This approach enables adaptive data management tailored

to the operational constraints of IoT sensors, enhancing their practical deployment in predictive maintenance without sacrificing critical system functionality [9].

The proposed methodology demonstrates potential for generalizability across various data types and IoT applications beyond predictive maintenance. By leveraging a modular and adaptable wavelet transform framework, this method supports scalable compression ratios and dynamic adjustments based on energy availability. These characteristics make it suitable for diverse IoT applications such as environmental monitoring, where real-time data on air quality, soil conditions, or weather patterns requires both immediate and deferred data transmission. Additionally, the method's ability to operate on resource-limited microcontrollers enhances its applicability in smart agriculture, where it could manage data from soil moisture sensors, temperature sensors, and irrigation control systems. In smart cities, this methodology could optimize data from energy-harvesting sensors in public infrastructure, such as structural health monitoring of bridges or traffic flow management. The lightweight implementation tailored for ARM Cortex-M microcontrollers further ensures compatibility with other MCU families after suitable optimization, allowing deployment in numerous IoT scenarios requiring high energy efficiency. The flexibility of wavelet transform in capturing and reconstructing features in different signal types underscores its utility across varied domains, from audio and image processing to biomedical signal analysis, demonstrating the method's broad applicability in IoT ecosystems.

The study has several limitations. The implementation is designed specifically for ARM Cortex-M MCUs, and the ASM implementation would need to be adjusted for different instruction sets used in other core types. Another potential drawback could be the cluster size. The signal needs to be clustered according to a specific cluster size, and it is not appropriate to compress clusters of different sizes. Additionally, the method allows for dynamic changes in the compression level, which may result in a signal with varying compression levels over time, potentially causing a disadvantage. Another limitation could be the energy consumption of the proposed solution. The additional energy required for compression must be considered as a trade-off between reducing the data size and the energy overhead introduced by the compression process. Striking a balance between these factors is critical, particularly in energy-constrained environments, where the benefits of data reduction need to outweigh the cost of increased energy consumption.

VII. CONCLUSION

The study introduced an implementation methodology suitable for IIoT solutions with limited energy and computational resources, providing the option to customize the level of information loss during data transmission. It demonstrated an analysis of the computational costs on an MCU with and without hardware support for floating-point instructions and presented a case study on an industrial vibration dataset illustrating data loss.

A computational cost analysis was performed to compare the execution of the code in C and ASM, as well as the performance of the DWT and FWT algorithms. GCC compiler optimization tools were also incorporated. The results of the experiment showed that execution of the FWT in ASM saved up to 32.1% energy, while execution of the DWT in C saved up to 87%.

The proposed implementation methodology offers practical advantages in industrial settings by seamlessly integrating with cloud-based edge computing platforms. Its dynamic data compression adapts to energy availability, ensuring essential information is transmitted promptly while detailed data is deferred for later. Optimized for low-cost microcontrollers, the method significantly reduces energy consumption and computational demands, making it ideal for energy-harvesting IIoT systems. This hybrid approach supports real-time monitoring and long-term analytics, enhancing predictive maintenance capabilities while enabling scalability and sustainability in diverse industrial applications.

The main benefit of this study is that the lightweight WT implementation was evaluated on a particular ARM Cortex-M7 central processing unit, indicating that the results can be transferable to other microcontrollers (MCUs) in the ARM Cortex-M family. A limitation, however, is that for other central processing unit families, such as RISC-V, only the C implementation is directly transferable due to the different architecture of the instruction set. Even though the C implementation can be used, the compilation process will produce different assembly code, which may result in varying computational performance.

Future research could explore implementation on the RISC-V architecture, which uses a different instruction set, potentially combined with hardware accelerators such as digital signal processing (DSP) engines, direct memory access (DMA) acceleration, or System on Chips (SoCs) with Field Programmable Gate Array (FPGA) integration, where part of the algorithm could be realized at the hardware level. This would further enhance performance and energy efficiency, making the method even more adaptable for different use cases.

Additionally, the approach could be integrated into systems using machine learning, where the signal variability is critical. These systems could be designed to adjust dynamically to the incoming information, ensuring that signal processing adapts to changing conditions in real-time.

This research has a major impact on systems that use energy harvesting, as it allows for dynamic data transmission. IoT sensors can schedule transmissions based on the available energy, optimizing operational efficiency and extending the lifespan of the sensors in energy-constrained environments.

The study also opens up future research possibilities, such as examining the impact of data loss on monitoring

quality, particularly for predictive maintenance applications. Another direction for research could involve exploring state-of-the-art or future MCUs with specialized computational units that enable faster processing and reduced energy consumption.

DECLARATION OF GENERATIVE AI AND AI-ASSISTED TECHNOLOGIES IN THE WRITING PROCESS

During the preparation of this work, the authors used GPT-4 for language correction. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

REFERENCES

[1] A. Bousdekis, K. Lepenioti, D. Apostolou, and G. Mentzas, "A review of data-driven decision-making methods for industry 4.0 maintenance applications," *Electronics*, vol. 10, no. 7, p. 828, Mar. 2021.

[2] A. Jimenez-Cortadi, I. Irigoien, F. Boto, B. Sierra, and G. Rodriguez, "Predictive maintenance on the machining process and machine tool," *Appl. Sci.*, vol. 10, no. 1, p. 224, Dec. 2019.

[3] S. Namuduri, B. N. Narayanan, V. S. P. Davuluru, L. Burton, and S. Bhansali, "Review—Deep learning methods for sensor based predictive maintenance and future perspectives for electrochemical sensors," *J. Electrochemical Soc.*, vol. 167, no. 3, Feb. 2020, Art. no. 037552.

[4] H. Nizam, S. Zafar, Z. Lv, F. Wang, and X. Hu, "Real-time deep anomaly detection framework for multivariate time-series data in industrial IoT," *IEEE Sensors J.*, vol. 22, no. 23, pp. 22836–22849, Dec. 2022.

[5] S. F. Ahmed, M. S. B. Alam, M. Hoque, A. Lameesa, S. Afrin, T. Farah, M. Kabir, G. Shafiullah, and S. M. Muyeen, "Industrial Internet of Things enabled technologies, challenges, and future directions," *Comput. Electr. Eng.*, vol. 110, Sep. 2023, Art. no. 108847. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790623002719

[6] R. Sahal, J. G. Breslin, and M. I. Ali, "Big data and stream processing platforms for industry 4.0 requirements mapping for a predictive maintenance use case," *J. Manuf. Syst.*, vol. 54, pp. 138–151, Jan. 2020.

[7] T. Hafeez, L. Xu, and G. Mcardle, "Edge intelligence for data handling and predictive maintenance in IIOT," *IEEE Access*, vol. 9, pp. 49355–49371, 2021.

[8] F. Calabrese, A. Regattieri, M. Bortolini, M. Gamberi, and F. Pilati, "Predictive maintenance: A novel framework for a data-driven, semi-supervised, and partially online prognostic health management application in industries," *Appl. Sci.*, vol. 11, no. 8, p. 3380, Apr. 2021.

[9] J. Konecny, M. Prauzek, and M. Borova, "Fuzzy controlled wavelet based edge computing method for energy harvesting IoT sensors," *IEEE Internet Things J.*, vol. 10, no. 21, pp. 18909–18918, Nov. 2023.

[10] T. Sanislav, G. D. Mois, S. Zeadally, and S. C. Folea, "Energy harvesting techniques for Internet of Things (IoT)," *IEEE Access*, vol. 9, pp. 39530–39549, 2021.

[11] S. Zeadally, F. K. Shaikh, A. Talpur, and Q. Z. Sheng, "Design architectures for energy harvesting in the Internet of Things," *Renew. Sustain. Energy Rev.*, vol. 128, Aug. 2020, Art. no. 109901.

[12] M.-U.-R. Ashraf Virk, M. F. Mysorewala, L. Cheded, and A. Aliyu, "Review of energy harvesting techniques in wireless sensor-based pipeline monitoring networks," *Renew. Sustain. Energy Rev.*, vol. 157, Apr. 2022, Art. no. 112046.

[13] T. Kucova, M. Prauzek, J. Konecny, D. Andriukaitis, M. Zilys, and R. Martinek, "Thermoelectric energy harvesting for Internet of Things devices using machine learning: A review," *CAAI Trans. Intell. Technol.*, vol. 8, no. 3, pp. 680–700, Sep. 2023.

[14] X. Chen, H. Hu, J. Zhou, Y. Li, L. Wan, Z. Cheng, J. Chen, J. Xu, and R. Zhou, "Indoor photovoltaic materials and devices for self-powered Internet of Things applications," *Mater. Today Energy*, vol. 44, Aug. 2024, Art. no. 101621.

[15] Q. He and J. Briscoe, "Piezoelectric energy harvester technologies: Synthesis, mechanisms, and multifunctional applications," *ACS Appl. Mater. Inter.*, vol. 16, no. 23, pp. 29491–29520, Jun. 2024.

[16] F. Zonzini, V. Dertimanis, E. Chatzi, and L. D. Marchi, "System identification at the extreme edge for network load reduction in vibration-based monitoring," *IEEE Internet Things J.*, vol. 9, no. 20, pp. 20467–20478, Oct. 2022.

[17] M. Zhang, H. Zhang, C. Zhang, and D. Yuan, "Communication-efficient quantized deep compressed sensing for edge-cloud collaborative industrial IoT networks," *IEEE Trans. Ind. Informat.*, vol. 19, no. 5, pp. 6613–6623, May 2023.

[18] V. Makarichev, V. Lukin, O. Illiashenko, and V. Kharchenko, "Digital image representation by atomic functions: The compression and protection of data for edge computing in IoT systems," *Sensors*, vol. 22, no. 10, p. 3751, May 2022.

[19] J. Li, X. Liu, Y. Gao, L. Zhuo, and J. Zhang, "BARRN: A blind image compression artifact reduction network for industrial IoT systems," *IEEE Trans. Ind. Informat.*, vol. 19, no. 9, pp. 9479–9490, Sep. 2023.

[20] F. M. Chache, S. Maxon, R. M. Narayanan, and R. Bharadwaj, "Effects of lossy compression on the age of information in a low power network," in *Proc. IEEE 24th Int. Symp. World Wireless, Mobile Multimedia Netw. (WoWMoM)*, Jun. 2023, pp. 382–387.

[21] X. Chen, Q. Yu, S. Dai, P. Sun, H. Tang, and L. Cheng, "Deep reinforcement learning for efficient IoT data compression in smart railroad management," *IEEE Internet Things J.*, vol. 11, no. 15, pp. 25494–25504, Aug. 2024.

[22] U. Serhii and K. Vasyl, "Optimizing data transmission in IoT networks through enhanced compression and edge computing techniques," in *Proc. IEEE Int. Conf. Inf. Telecommun. Technol. Radio Electron.*, Nov. 2023, pp. 76–79.

[23] J. Too, A. Rahim, and N. Mohd, "A comparative analysis of wavelet families for the classification of finger motions," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 4, pp. 1–6, 2019.

[24] A. Hazarika, S. Poddar, M. M. Nasralla, and H. Rahaman, "Area and energy efficient shift and accumulator unit for object detection in IoT applications," *Alexandria Eng. J.*, vol. 61, no. 1, pp. 795–809, Jan. 2022.

[25] C.-H. Chen, M.-Y. Lin, and C.-C. Liu, "Edge computing gateway of the industrial Internet of Things using multiple collaborative microcontrollers," *IEEE Netw.*, vol. 32, no. 1, pp. 24–32, Jan. 2018.

[26] Y. K. Teoh, S. S. Gill, and A. K. Parlikad, "IoT and fog-computing-based predictive maintenance model for effective asset management in industry 4.0 using machine learning," *IEEE Internet Things J.*, vol. 10, no. 3, pp. 2087–2094, Feb. 2023.

[27] F. Lauer, M. Schöffel, C. C. Rheinländer, and N. Wehn, "Exploration of thermoelectric energy harvesting for secure, TLS-based industrial IoT nodes," in *Internet of Things—ICIOT 2022* (Lecture Notes in Computer Science), vol. 13735. Cham, Switzerland: Springer, 2023. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-031-23582-5_7

JAROMIR KONECNY (Senior Member, IEEE) was born in Frýdek-Místek, Czech Republic, in 1986. He received the bachelor's degree in control and information systems, in 2008, the master's degree in measurement and control engineering, in 2010, and the Ph.D. degree in technical cybernetics, in 2014. Since 2012, he has been with the Department of Cybernetics and Biomedical Engineering, VSB—Technical University of Ostrava, Czech Republic, where he is currently an Associate Professor. He has authored more than 70 articles and conference papers and has four registered inventions. His research interests include embedded systems, electronics, environmental monitoring systems, and localization systems in robotics. He is an active IEEE Senior Member of the IEEE Systems, Man and Cybernetics Society, the IEEE Computational Intelligence Society, and the IEEE Internet of Things Community.

JAN CHOUTKA was born in Moravská Třebová, Czech Republic, in 1997. He received the bachelor's degree in control and information systems, in 2020, and the master's degree in measurement and control engineering, in 2022. He is currently pursuing the Ph.D. degree in cybernetics. Since 2024, he has been with the Department of Cybernetics and Biomedical Engineering, VSB—Technical University of Ostrava, Czech Republic. His research interests include embedded systems, electronics, and environmental monitoring systems.

RADIM HERCIK was born in Ostrava, Czech Republic, in 1987. He received the bachelor's degree in control and information systems, in 2009, the master's degree in measurement and control engineering, in 2011, and the Ph.D. degree in technical cybernetics, in 2014. His professional experience includes working as a Developer of ultrasonic automotive sensors at Continental Automotive Czech Republic s.r.o., until 2020. Since 2020, he has been with the Department of Cybernetics and Biomedical Engineering, VSB—Technical University of Ostrava, Czech Republic, where he currently holds the position of an Assistant Professor. He has authored more than 20 articles and conference papers and holds six registered inventions. His research interests include embedded systems, automation, industrial robotics, and mobile robotics.

JIRI KOZIOREK is currently a Full Professor in cybernetics at the VSB—Technical University of Ostrava (VSB-TUO), Czech Republic. Since 1998, he has been with the Department of Cybernetics and Biomedical Engineering, VSB-TUO, where he has also been the Head of the Department, since 2009. He is a coordinator of several national and international research projects, typically in cooperation with industrial partners. He is the author/co-author of more than 100 scientific publications. His research interests include industrial automation, control system design, industrial communications, digitization of industry, and sensors. His pedagogical practice relates to his research areas.

DANGIRUTIS NAVIKAS received the M.Sc. and Ph.D. degrees in electronics engineering, in 1994 and 1999, respectively. He is with the Department of Electronics Engineering, Faculty of Electrical and Electronics Engineering, Kaunas University of Technology, where he is currently the Head of the Department. His research interests include finding solutions for the issues related to the interactive design of microprocessor systems, integrated information systems, and WSN.

DARIUS ANDRIUKAITIS (Member, IEEE) received the M.Sc. and Ph.D. degrees in electronics engineering, in 2005 and 2009, respectively. He is with the Department of Electronics Engineering, Faculty of Electrical and Electronics Engineering, Kaunas University of Technology, where he is currently the Vice Dean of Research. His research interests include finding solutions for the issues related to interactive electronic systems, integrated information systems, and WSN.

MICHAL PRAUZEK (Senior Member, IEEE) was born in Ostrava, Czech Republic, in 1983. He received the bachelor's degree in control and information systems, the master's degree in measurement and control systems, and the Ph.D. degree in technical cybernetics from the VSB—Technical University of Ostrava (VSB—TUO), Czech Republic, in 2006, 2008, and 2011, respectively. Since 2010, he has been with the Department of Cybernetics and Biomedical Engineering, VSB—TUO, where he is currently a Full Professor. From 2013 to 2014, he was a Research Postdoctoral Fellow with the University of Alberta, Canada. He has authored more than 100 articles and conference papers and has 11 registered inventions. His research interests include embedded systems, data and signal analysis, control design, and machine learning. He is an active IEEE Senior Member in the IEEE Systems, Man and Cybernetics Society and the IEEE Engineering in Medicine and Biology Society.