sensors

MDPI

Article

DynDSE: Automated Multi-Objective Design Space Exploration for Context-Adaptive Wearable IoT Edge Devices

Giovanni Schiboni *, Juan Carlos Suarez, Rui Zhang and Oliver Amft

Digital Health, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), 91052 Erlangen, Germany; juan.c.suarez@fau.de (J.C.S.); rui.rui.zhang@fau.de (R.Z.); oliver.amft@fau.de (O.A.)

* Correspondence: giovanni@schiboni.it

Received: 22 September 2020; Accepted: 21 October 2020; Published: 27 October 2020; Corrected: 8 September 2022

Abstract: We describe a simulation-based Design Space Exploration procedure (DynDSE) for wearable IoT edge devices that retrieve events from streaming sensor data using context-adaptive pattern recognition algorithms. We provide a formal characterisation of the design space, given a set of system functionalities, components and their parameters. An iterative search evaluates configurations according to a set of requirements in simulations with actual sensor data. The inherent trade-offs embedded in conflicting metrics are explored to find an optimal configuration given the application-specific conditions. Our metrics include retrieval performance, execution time, energy consumption, memory demand, and communication latency. We report a case study for the design of electromyographic-monitoring eyeglasses with applications in automatic dietary monitoring. The design space included two spotting algorithms, and two sampling algorithms, intended for real-time execution on three microcontrollers. DynDSE yielded configurations that balance retrieval performance and resource consumption with an F1 score above 80% at an energy consumption that was 70% below the default, non-optimised configuration. We expect that the DynDSE approach can be applied to find suitable wearable IoT system designs in a variety of sensor-based applications.

Keywords: health monitoring; automatic dietary monitoring; physiological sensing; pattern spotting; energy saving; embedded machine learning

1. Introduction

Autonomous wearable IoT devices are being used for physiological and behavioural health-monitoring [1] and provide relevant health status information to their wearers [2,3]. Miniaturised electronics embedded in wearable accessories, garments, etc., provide the resources to retrieve pattern events from streaming sensor data and to interact with the wearer, which led to the concept of edge computing [4]. Edge computing aims to process data at the devices end, rather than the cloud to reduce network load and service response time. Furthermore, reducing communication bandwidth often lowers energy consumption, as well as privacy, and security concerns. For example, in medical IoT monitoring applications [5], a device may retrieve relevant events using embedded machine learning methods, thus sending only abstract event information to the cloud. Nevertheless, resource constrains are a key feature of IoT devices. A wearable IoT device typically consists of multiple sensors, a microcontroller (μC), which runs data processing algorithms, memory, and a radio module for data communication (Figure 1).

Therefore, the optimisation of resource-constrained IoT edge devices has high priority in the design process [6].

The design of an IoT device can be interpreted as the selection of an optimal hardware and software configuration from the wide design space of possible options. Certain system configurations may be not compatible with specific system requirements, e.g., energy saving and retrieval performance.

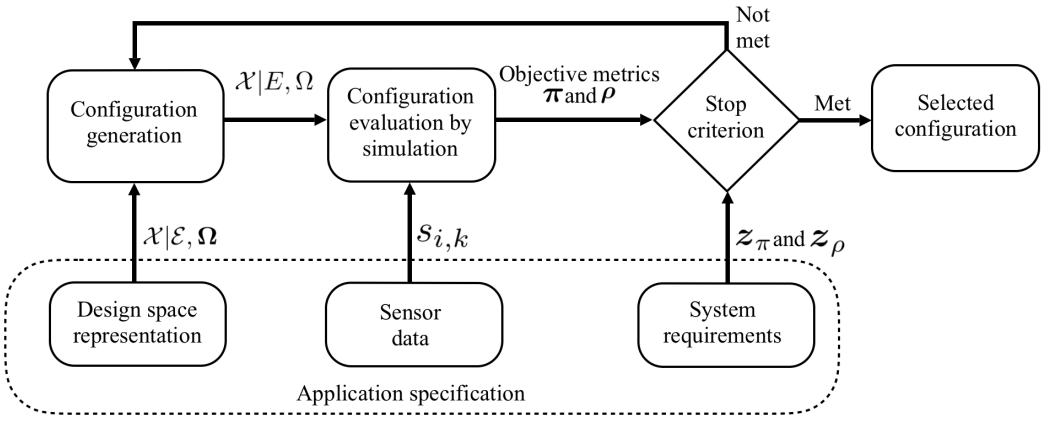

Figure 1. DynDSE procedure for wearable IoT edge devices. $\mathcal{X}|\mathcal{E},\boldsymbol{\Omega}$: design space; $\mathcal{X}|E,\Omega$: system configuration; $\boldsymbol{\pi}$ benefit metrics set; $\boldsymbol{\rho}$: cost metrics set; $\boldsymbol{z}_\pi$: benefit requirement set; $\boldsymbol{z}_\rho$: cost requirement set; $s_{i,k}$: data sample i from channel k.

Available μCs present differences in terms of resource consumption, execution time, and energy consumption. With the complex interplay of hardware and software components, it is a difficult task to quantify resource use and to manually identify optimal configurations that fulfil system requirements best. The size of the architectural design space often makes manual exploration of embedded systems unfeasible. Automated Design Space Exploration (DSE) [7,8] provides a computational framework to identify optimal configurations.

The design problem is exacerbated when the system does include functions that cannot be statically approximated, and by those with a dynamic effect on the resource-performance trade-off. Sampling strategies, e.g., context-adaptive sampling [9], are a basic dynamic function of wearable IoT devices. A context-adaptive sampling is a dynamic sampling strategy that tunes the sensor's sampling rate based on a context measure, thus aiming at minimising energy consumption. The stochastic and variable nature of human behavioural patterns induces variability into context-adaptive sampling behaviour, which, in turn, drastically affects the resource-perfomance trade-off. Therefore, the main challenge for the design of a context-adaptive wearable IoT device lies in the identification of viable configurations that fulfil the system requirements under dynamically varying conditions. With DynDSE, we explicitly incorporate context-adaptive system behaviour in the design exploration and simulate systems with actual sensor data.

In this paper, we provide the following contributions:

1. We present a simulation-based Design Space Exploration (DynDSE) procedure for wearable IoT devices that employ context-adaptive pattern recognition algorithms for event retrieval, see Figure 1. We provide a formal characterisation of the design space, given a set of system functionalities, components, and their parameters. An iterative search evaluates configurations according to system requirements. The inherent trade-off embedded in conflicting objectives are explored to find an optimal configuration.

2. We perform a wearable IoT application evaluation to analyse the resource-performance trade-off considering static and dynamic design aspects through simulations with actual data of Electromyography (EMG)-monitoring eyeglasses in automated dietary monitoring.

In the present work, we formally introduce the DynDSE exploration framework in Section 3 and relevant optimisation metrics in Section 4. Subsequently, a comprehensive wearable IoT application case is presented and analysed in Sections 5 and 6 to detail the potential of DynDSE.

2. Related Work

DSE frameworks are used for hardware/software co-design of heterogeneous multiprocessor and system-on-chip architectures [10], embedded systems [11], or Field Programmable Gate Array (FPGA) platforms [12]. In conventional DSE approaches, multiple metrics, e.g., energy consumption, memory demand, and cost, must be optimised concurrently according to some application requirements. The conflicting nature of objectives, which reflect many system characteristics, produces trade-offs inherent to the overall system performance. A decision on the most adequate system configuration needs to be taken according to a multi-objective optimisation process. The majority of DSE methods belongs to one of three analysis and evaluation categories, i.e., prototype-based, analytics-based, and simulation-based. The three categories differ in terms of a design time-modelling accuracy trade-off [8]. For example, the prototype-based evaluation provides adequate modelling accuracy, but requires development time with limited exploration capability. The analytics-based evaluation relies on analytical description of component interactions, which allows designers to explore larger design space portions in acceptable time. However, especially with complex architectures, the modelling accuracy of analytics-based evaluations is limited. The simulation-based evaluation is the most versatile approach, as time-modelling accuracy trade-offs of different designs can be achieved by tuning the simulation characteristics. For example, a lower abstraction simulation level, i.e., simulating digital signals between registers and combinational logic, yields higher accuracy but lowers simulation time for analysing software stacks. A higher abstraction simulation level, i.e., simulating system components at the cycle level, computes more efficient simulation, at the cost of averaging inter-cycle system state information. Moreover, simulation-based DSE enables dynamic profiling at run time, which allows the designer to quantify and optimise complex dynamic component interactions and workload.

In last two decades, many DSE approaches have been proposed to design wearable devices. For example, Bharatula et al. [13] proposed a design method to achieve a resource-performance trade-off for a highly-miniaturised multi-sensor context recognition system. Their evaluation showed that, through variations of system design space, the optimisation method was able to extend the battery lifetime by several hours. The same research group introduced multiple metrics to analyse the resource-performance trade-off of a wearable system, i.e., an accelerometer, a microphone, a light sensor, and a TI MSP430F1611 μC [14]. The authors presented an experimental validation in which a manual multi-objective optimisation was applied to find the best system configuration. In contrast to the works of Bharatula et al., we propose an analytical characterisation of the system architecture with the aim to automate the DSE modelling. Anliker et al. [15,16] presented an automatic design methodology based on abstract modules and task-dependent constraint sets. A multi-objective function incorporated recognition accuracy, energy consumption, and wearability, applied to classification of three modes of locomotion. In contrast, our simulation-based analysis is based on a realistic dataset collected in daily living. We evaluated the relevant effects of the free-living settings on the metrics. Beretta et al. [17] presented a model-based design optimisation to analytically characterise the energy consumption of a wearable sensor node. The authors described a multi-objective exploration algorithm to evaluate system configurations and relative trade-off. The method was application-driven with a fixed system architecture. Stäger et al. [18] took in account

several system configuration options, including sensor types, sensor parameters, features, and classifiers. A case study was described related to detection of interactions with household appliances by means of a wrist worn microphone and accelerometer. Evidences were presented for an improvement in battery lifetime by a factor 2–4 with only little degradation in recognition performance.

The aforementioned DSE approaches focused on the evaluation and exploration of wearable device architectures under static workloads. However, todays wearable systems may adopt opportunistic sensing strategies to balance energy consumption and information acquired by a wearable or mobile systems. For example, Rault et al. [19] provided an analysis of techniques for energy consumption reduction in wearable sensors for healthcare applications. Opportunistic sensing strategies consider dynamic effects on the resource consumption trade-off, not considered in a static DSE. For example, an adaptive sampling scheme may reduce sampling rate according to a detected lower signal entropy. For instance, Mesin [20] proposed an adaptive sampling scheme based on sample prediction, where a non-uniform schedule increased the sampling rate only during bursts of physical activity. A multi-layer perceptron predicted subsequent samples and their uncertainties, triggering a measurement when the uncertainty of the prediction exceeded a threshold. In contrast, our approach employs a transparent state-based reactive model to estimate relevance of future samples. Scarabottolo et. al. [21] presented a dynamic sampling strategy for low-power embedded devices. The sampling rate tuning was based on the analysis of the signal's spectral content. Rieger and Taylor [22] proposed a low-power analog system for real-time adaptive sampling rate tuning, proportional to the signal curvature. Different from our approach, a-priori knowledge was required. Moreover, in contrast to Rieger and Taylor's [22] low-power analog system, we consider a pattern spotting problem to analyse performance that is frequently required in wearable IoT systems.

3. Exploration Framework

3.1. Design Space Representation

The design space configurations consist of set $\mathcal{X}$ of functionalities, realised by a set $\mathcal{E}$ of components, which are characterised by the set $\mathbf{\Omega}$ of parameters. An example is provided at the end of this section. Formally, the set of functionalities $\mathcal{X}$ is expressed as:

$$\begin{aligned} \mathcal{X} &= \{\Xi_\xi\}_{\xi \in \boldsymbol{\xi}}, \\ \boldsymbol{\xi} &= \{\xi | \xi \in \mathbb{N}_1, \xi \leq N_\Xi\}, \end{aligned} \tag{1}$$

where the functionality Ξ_ξ is the element of the set $\mathcal{X}$, $\boldsymbol{\xi}$ is the index set to $\mathcal{X}$, and N_Ξ is the number of elements of $\boldsymbol{\xi}$.

A functionality Ξ_ξ is carried out by one or more components grouped in the set $\boldsymbol{\epsilon}_\xi$, indexed by the index set $\boldsymbol{q}_\xi$, expressed as:

$$\begin{aligned} \boldsymbol{\epsilon}_\xi &= \{\epsilon_{\xi,q}\}_{q \in \boldsymbol{q}_\xi} \text{ and} \\ \boldsymbol{q}_\xi &= \{q | q \in \mathbb{N}_1, q \leq N_\xi\}, \end{aligned} \tag{2}$$

where the component $\epsilon_{\xi,q}$ is the element of the set $\boldsymbol{\epsilon}_\xi$, $\boldsymbol{q}_\xi$ is the index set to $\boldsymbol{\epsilon}_\sim$, and N_ξ is the number of components associated to the functionality ξ.

Overall, the design space consists of a collection $\mathcal{E}$ of system components sets, indexed by the collection $\boldsymbol{Q}$ of index sets, expressed as:

$$\begin{aligned} \mathcal{E} &= \{\boldsymbol{\epsilon}_{\xi}\}_{\xi \in \boldsymbol{\xi}} \text{ and} \\ \boldsymbol{Q} &= \{\boldsymbol{q}_{\xi} | \xi \in \boldsymbol{\xi}\}. \end{aligned} \tag{3}$$

A component $\epsilon_{\xi,q}$ is characterised by one or more component parameters grouped in the set $\boldsymbol{\omega}_{\xi,q}$, indexed by the index set $\boldsymbol{w}_{\xi,q}$, expressed as:

$$\begin{aligned} \boldsymbol{\omega}_{\xi,q} &= \{\omega_{\xi,q,w}\}_{w \in \boldsymbol{w}_{\xi,q}} \text{ and} \\ \boldsymbol{w}_{\xi,q} &= \{w | w \in \mathbb{N}_1, w \leq N_{\xi,q}\}, \end{aligned} \tag{4}$$

where the component parameter $\omega_{\xi,q,w}$ is the element of the set $\boldsymbol{\omega}_{\xi,q}$, $\boldsymbol{w}_{\xi,q}$ is the index set to $\boldsymbol{\omega}_{\xi,q}$, and $N_{\xi,q}$ is the number of component parameters associated to the functionality ξ and the component q. Overall, the design space consists of a collection $\boldsymbol{\Omega}$ of component parameters sets, indexed by the collection $\boldsymbol{W}$ of index sets, expressed as:

$$\begin{aligned} \boldsymbol{\Omega} &= \{\boldsymbol{\omega}_{\xi,q}\}_{(\xi,q) \in \bigcup_{\xi \in \boldsymbol{\xi}} (\xi \times \boldsymbol{q}_{\xi})} \text{ and} \\ \boldsymbol{W} &= \{\boldsymbol{w}_{\xi,q} | (\xi, q) \in \bigcup_{\xi \in \boldsymbol{\xi}} (\xi \times \boldsymbol{q}_{\xi})\}. \end{aligned} \tag{5}$$

For example, a spotting algorithm Ξ_1 is represented by a component $\epsilon_{1,1}$, e.g., a FFT-based algorithm, characterised by a component parameter $\omega_{1,1,1}$, e.g., the data frame size. Data sampling Ξ_2 is represented by a component $\epsilon_{2,1}$, e.g., an uniform sampling. A processing unit Ξ_3 is represented by a component $\epsilon_{3,2}$, e.g., a Texas Instrument μC, characterised by a component parameter $\omega_{3,2,1}$, e.g., the μC's clock frequency. In a compact form, the design space is expressed as:

$$\mathcal{X} | \mathcal{E}, \boldsymbol{\Omega}. \tag{6}$$

3.2. Configuration Generation

The configuration generation selects a design candidate to be evaluated in the simulation, i.e., see Section 3.3. The configuration generation is composed by two main stages. In the first stage, for each functionality Ξ_{ξ}, a component set $\boldsymbol{\epsilon}^c_{\xi}$, indexed by the index set $\boldsymbol{q}^c_{\xi}$, is selected as:

$$\begin{aligned} \boldsymbol{\epsilon}^c_{\xi} &= \{\epsilon_{\xi,q}\}_{q \in \boldsymbol{q}^c_{\xi}} \text{ and} \\ \boldsymbol{q}^c_{\xi} &\subseteq \boldsymbol{q}_{\xi}, \end{aligned} \tag{7}$$

where $\boldsymbol{q}^c_{\xi}$ represents a subset of the the index set $\boldsymbol{q}_{\xi}$. Overall, a configuration consists of a collection E of system components sets, indexed by the collection $\boldsymbol{Q}^c$ of index sets, expressed as:

$$\begin{aligned} E &= \{\boldsymbol{\epsilon}^c_{\xi}\}_{\xi \in \boldsymbol{\xi}} \text{ and} \\ \boldsymbol{Q}^c &= \{\boldsymbol{q}^c_{\xi} | \xi \in \boldsymbol{\xi}\}, \end{aligned} \tag{8}$$

In the second stage, for each component $\epsilon_{\xi,q} \in \boldsymbol{\epsilon}^c_\xi$ a component parameters set $\boldsymbol{\omega}^c_{\xi,q}$, indexed by the index set $\boldsymbol{w}^c_{\xi,q}$, is selected as:

$$\begin{aligned} &\boldsymbol{\omega}^c_{\xi,q} = \{\omega_{\xi,q,w}\}_{w \in \boldsymbol{w}^c_{\xi,q}} \text{ and} \\ &\boldsymbol{w}^c_{\xi,q} \subseteq \boldsymbol{w}_{\xi,q}, \end{aligned} \tag{9}$$

where $\boldsymbol{w}^c_{\xi,q}$ represents a subset of the the index set $\boldsymbol{w}^c_{\xi,q}$. Overall, a system configuration consists of a collection Ω of system component parameters sets, indexed by the collection $\boldsymbol{W}^c$ of index sets, expressed as:

$$\begin{aligned} &\Omega = \{\boldsymbol{\omega}^c_{\xi,q}\}_{(\xi,q) \in \bigcup_{\xi \in \boldsymbol{\xi}} (\xi \times \boldsymbol{q}^c_\xi)} \text{ and} \\ &\boldsymbol{W}^c = \{\boldsymbol{w}^c_{\xi,q} | (\xi, q) \in \bigcup_{\xi \in \boldsymbol{\xi}} (\xi \times \boldsymbol{q}^c_\xi)\}. \end{aligned} \tag{10}$$

In a compact form, a system configuration is expressed as:

$$\mathcal{X}|E, \Omega \subseteq \mathcal{X}|\mathcal{E}, \boldsymbol{\Omega}. \tag{11}$$

3.3. Configuration Evaluation

A metric estimates benefits π or costs ρ of a configuration. The configuration evaluation is based on two sets of metrics. The benefit metric set is defined as:

$$\boldsymbol{\pi} = \{\pi^p(\mathcal{X}|E, \Omega)\}_{p \in \boldsymbol{p}} \text{ with} \tag{12}$$

$$\boldsymbol{p} = \{p | p \in \mathbb{N}_1, p \leq N_p\}, \tag{13}$$

where the element $\pi^p(\mathcal{X}|E, \Omega)$ is a benefit metric and N_p is the number of benefit metrics. The benefit metric set $\boldsymbol{\pi}$ is subjected to the benefit requirement set as:

$$\boldsymbol{z}_\pi = \{z^p_\pi\}_{p \in \boldsymbol{p}}, \tag{14}$$

Similarly, the cost metric set is defined as:

$$\boldsymbol{\rho} = \{\rho^r(\mathcal{X}|E, \Omega)\}_{r \in \boldsymbol{r}} \text{ with} \tag{15}$$

$$\boldsymbol{r} = \{r | r \in \mathbb{N}_1, r \leq N_r\}. \tag{16}$$

where the element $\rho^r(\mathcal{X}|E, \Omega)$ is a cost metric and N_r is the number of cost metrics. The cost objective set $\boldsymbol{\rho}$ is subjected to the cost requirement set as:

$$\boldsymbol{z}_\rho = \{z^r_\rho\}_{r \in \boldsymbol{r}}. \tag{17}$$

Each $\pi^p(\mathcal{X}|E, \Omega)$ and $\rho^r(\mathcal{X}|E, \Omega)$ maps a configuration $\mathcal{X}|E, \Omega$ to the real space $\mathbb{R}$, i.e., $\mathcal{X}|E, \Omega \to \mathbb{R}$.

The overall optimisation process is formally described as follows. Given a design space $\mathcal{X}|\mathcal{E}, \boldsymbol{\Omega}$, the goal of DynDSE is to find the configuration $\mathcal{X}|E, \Omega$, which maximises benefits $\boldsymbol{\pi}$ and minimises costs

$\boldsymbol{\rho}$, while respecting the respective set of requirements. The problem can be interpreted as a constrained multi-objective optimisation:

$$\begin{aligned} \max_{\mathcal{X}|E,\Omega\subseteq\mathcal{X},\mathcal{E}|\boldsymbol{\Omega}} \ & \sum_{p} \pi^p(\mathcal{X}|E,\Omega) - \sum_{r} \rho^r(\mathcal{X}|E,\Omega) \\ \text{s.t. } & z_\rho^r \leq 0 \ \ \forall r \in \boldsymbol{r}, \\ & z_\pi^p \geq 0 \ \ \forall p \in \boldsymbol{p}. \end{aligned} \tag{18}$$

The optimisation provides a set of mutually conflicting solutions, which reflects the trade-offs in the design. To define optimality, one can usually exploit the concept of Pareto-dominance, i.e., a decision maker prefers a configuration to another if it is equal or better in all objectives and strictly better in at least one. As Künzli et al. [23] pointed out, several approaches exists to solve the multi-objective optimisation, e.g., exploration by hand [24], exhaustive search [25], or reduction to a single objective [26], as done in this work.

4. Optimisation Metrics

Table 1 lists the metrics introduced in this Section and their respective requirements.

Table 1. Our metrics and their relative system requirements. The table also indicates the elements which affect the system requirements. T denotes the runtime duration and m the data frame length.

						The Requirement Is Imposed by		
	Objective	**Abbreviation**	**Description**	**Requirement**	**Value**	**Task**	**Battery life**	**μC**
$\pi(\mathcal{X}\|E,\Omega)$	$\pi^1(\mathcal{X}\|E,\Omega)$	P	Precision	z_π^1	$\geq$70%,	X		
	$\pi^2(\mathcal{X}\|E,\Omega)$	R	Recall	z_π^2	$\geq$80%	X		
$\rho(\mathcal{X}\|E,\Omega)$	$\rho^1(\mathcal{X}\|E,\Omega)$	ET	Ex. time in real-time	z_ρ^1	Real-time: $< m$	X		X
			Ex. time online		Online: $< T$	X		X
	$\rho^2(\mathcal{X}\|E,\Omega)$	EC	Energy Consumption	z_ρ^2	$\leq$52.8 mWh	X	X	
	$\rho^3(\mathcal{X}\|E,\Omega)$	MD	Memory Demand	z_ρ^3	Variable (see Table 5)			X

4.1. Retrieval Performance Metric

The retrieval performance of an event retrieval algorithm is expressed through Precision-Recall metric [27], as follows:

$$\begin{aligned} \text{Precision P} &= \pi^1(\mathcal{X}|E,\Omega), \\ \text{Recall R} &= \pi^2(\mathcal{X}|E,\Omega). \end{aligned} \tag{19}$$

From the application perspective, the algorithm must be able to keep an adequate level of retrieval performance. The retrieval performance requirements is usually defined by expert knowledge and we refer to it as z_π^1 and z_π^2.

4.2. Execution Time Metric

A measure of computational complexity denotes the execution time of an algorithm, by pairing the algorithm module and the processing module. An algorithm's abstraction is the decomposition of an algorithm in distinct stages, in which each stage is composed by one or more functions. Each function f is broken down by counting additions and subtractions (Add), multiplications (Mult), divisions (Div), square roots (Root), exponentials (Exp), and comparisons (Comp).

The number of machine cycles n_f^{cyc} to compute a function f on a μC is defined as:

$$\begin{aligned} n_f^{cyc} =& (n_{f,\text{Add}}^{op} \times n_{\text{Add}}^{cyc}) + (n_{f,\text{Mult}}^{op} \times n_{\text{Mult}}^{cyc}) + \\ & (n_{f,\text{Div}}^{op} \times n_{\text{Div}}^{cyc}) + (n_{f,\text{Root}}^{op} \times n_{\text{Root}}^{cyc}) + \\ & (n_{f,\text{Exp}}^{op} \times n_{\text{Exp}}^{cyc}) + (n_{f,\text{Comp}}^{op} \times n_{\text{Comp}}^{cyc}), \end{aligned} \tag{20}$$

where $n_{f,x}^{op}$ is the number of executions related to an arithmetical operation x to compute a function f and n_x^{cyc} is the number of cycles to execute an arithmetical operation x on a μC. To estimate the execution time ET_f of a function f, the number of machine cycles n_f^{cyc} is divided by the clock frequency ν of the μC, as follows:

$$\text{ET}_f = \frac{n_f^{cyc}}{\nu}. \tag{21}$$

The definition of the execution time ET depends on the runtime mode. When the runtime mode is real time, ET is computed as:

$$\text{ET} = \rho^1(\mathcal{X}|E, \Omega) = \sum_f \text{ET}_f. \tag{22}$$

When the runtime mode is online, ET is computed as:

$$\text{ET} = \sum_{fr} \sum_f \text{ET}_f, \tag{23}$$

where fr is a data frame process. We refer to the system requirements for ET as z_ρ^1.

4.3. Energy Consumption Metric

μC energy consumption: Most μCs support an active state and a stand-by state. The average energy consumption in active state $\text{EC}_f^{\mu C}$, related to the computation of a function f, is proportional to the ET_f and defined as:

$$\text{EC}_f^{\mu C} = \text{ET}_f \times P_{\text{act}}^{\mu C}, \tag{24}$$

where $\text{EC}_f^{\mu C}$ is expressed in Wh, $P_{\text{act}}^{\mu C}$ is the power consumption of the μC in active mode expressed in W, and ET_f is expressed in hours.

The average energy consumption in stand-by state $\text{EC}_{\text{stb}}^{\mu C}$ is modelled as:

$$\text{EC}_{\text{stb}}^{\mu C} = T_{\text{stb}} \times P_{\text{stb}}^{\mu C}, \tag{25}$$

where T_{stb} is the time period of inactivity expressed in hours, and $P_{\text{stb}}^{\mu C}$ is the power consumption in stand-by mode expressed in W.

The μC energy consumption was calculated as:

$$\text{EC}^{\mu C} = \sum_{fr} \sum_f \text{EC}_f^{\mu C} + \text{EC}_{\text{stb}}^{\mu C}, \tag{26}$$

where $\sum_{fr} \sum_f \text{EC}_f^{\mu C}$ denotes the active state energy.

Sensor energy consumption: The average instantaneous energy consumption EC_t^s for a single sensing component is computed by applying the following equations:

$$\begin{aligned} I_t^s &= I_{\mathrm{act}}^s \times D_t + I_{\mathrm{stb}}^s \times (1 - D_t), \\ \mathrm{EC}_t^s &= I_t^s \times V \times t_r, \end{aligned} \tag{27}$$

where I_{act}^s is the sensor's average current in active state, I_{stb}^s is the sensor's average current in stand-by state, D_t is the instantaneous duty cycle rate, EC_t^s is expressed in Wh, V is the voltage level of the sensing component, and t_r is the temporal resolution expressed in hours.

The sensor energy consumption was calculated as:

$$\mathrm{EC}^s = \sum_t \mathrm{EC}_t^s. \tag{28}$$

fFlash/Non-Volatile Memory Energy Consumption: To estimate the flash and programmable memory, we formulated an energy model inspired by Konstantakos et al. [28]. Writing energy consumption was determined by:

$$\mathrm{EC}_w^m = \sum_b I_{\mathrm{write}}^m \times V \times t_{\mathrm{w}}, \tag{29}$$

where b indicates a block, I_{write}^m is the average current consumption in writing mode, and write time t_{w} is the time required to write a memory block. The energy required to read a memory block was neglected. A static memory energy consumption term was computed as:

$$\mathrm{EC}_s^m = I_{\mathrm{stb}}^m \times V \times (T - \sum_b t_{\mathrm{w}}), \tag{30}$$

where I_{b}^m is the stand-by state current value, and T is the total simulation time.

The memory energy consumption was calculated as:

$$\mathrm{EC}^m = \mathrm{EC}_w^m + \mathrm{EC}_s^m. \tag{31}$$

Radio transmission energy consumption measure: To estimate the energy consumption of the wireless communication, we relied on the energy model described by Prayati et al. [29]. The model considered the following three stages for transmission: (1) initialisation of transmission and transferring of the frame data from memory to the radio chip FIFO buffer, (2) back-off timeout, and (3) packet transmission via the wireless channel.

In order to calculate the energy consumption to transmit a packet, the following formula was applied:

$$EC_p^r = I_{\mathrm{trans}} \times V \times t_{\mathrm{trans}}, \tag{32}$$

where I_{trans} is the transmission current, p indicating packets, and t_{trans} is the time requested to prepare and send a packet. As EC_p^r is expressed in Wh, t_{trans} must be converted in hours. In this work we considered $I_{\mathrm{trans}} = 21.7$ mA, when a transmission power threshold 0 dBm is chosen, and $t_{\mathrm{trans}} = 16$ ms is the time requested to prepare and send a packet size of 114 Bytes.

The radio transmission energy consumption was calculated as:

$$\mathrm{EC}^r = \sum_p EC_p^r. \tag{33}$$

Total energy consumption: The energy consumption metric EC is computed as follows:

$$\begin{aligned} &\mathrm{EC} = \rho^2(\mathcal{X}|E,\Omega) = \\ &\mathrm{EC}^{\mu C} + \mathrm{EC}^{s} + \mathrm{EC}^{m} + \mathrm{EC}^{r}. \end{aligned} \tag{34}$$

The application context imposes an energy budget requirement to the behavioural monitoring. For example, while monitoring dietary behaviour, the wearable system should be able to work uninterruptedly for the entire day, in order to not miss relevant activities. The system requirement z_ρ^2 for EC indicates the energy required to deploy the application for the entire runtime. We defined z_ρ^2 as the quantity in mW calculated as:

$$z_\rho^2 = \frac{\mathrm{BC}}{\mathrm{RT}} \cdot \phi, \tag{35}$$

where BC is the battery capacity in mWh, RT is the required runtime of the application expressed in hours, e.g., 16 h, and the factor $0 < \phi \leq 1$ considers the effect of external factors, which can affect the battery life. In this work we considered $\phi = 0.9$.

4.4. Memory Demand Metric

The memory demand MD is an upper bound of the memory required by the system to execute an event retrieval. MD is computed by considering four terms, i.e., the code memory m_c, the data memory m_d, the processing memory m_f, and the event memory m_e.

The memory demand is defined as:

$$\begin{aligned} &\mathrm{MD} = \rho^3(\mathcal{X}|E,\Omega) = m_c + m_d + \sum_f m_f + m_e, \\ &\text{with } m_f = m_{f,\textbf{Int}} + m_{f,\textbf{Float}}, \\ &\text{and } m_e = m_{e,\textbf{Int}} + m_{e,\textbf{Float}}, \end{aligned} \tag{36}$$

where $m_{f,\textbf{Int}}$ and $m_{e,\textbf{Int}}$ are the memory required to store integer values, and $m_{f,\textbf{Float}}$ and $m_{e,\textbf{Float}}$ are the memory required to store float values. We refer to the system requirement for MD as z_ρ^3, which represents the maximum amount of memory available to store information on a certain μC.

4.5. Communication Latency Metric

The communication latency is the time span between the event-related raw sensor data measurements and the delivery moment at the receiver of the retrieved event information. The communication latency metric CL is computed as follows:

$$\mathrm{CL} = \rho^4(\mathcal{X}|E,\Omega) = \sum_f \mathrm{ET}_f + \frac{m_e}{\nu_{tr}}, \tag{37}$$

$$\text{with } \nu_{tr} = \frac{\mathrm{MPS}}{connInterval}, \tag{38}$$

where MPS is the transmitter's maximum payload size, e.g., 216 bits for Bluetooth Low Energy (BLE), *connInterval* defines the time of connection events, i.e., ranges from 7.5 ms to 4.0 s with steps of 1.25 ms for BLE, and ν_{tr} is the transmission data rate. We refer to the system requirement for CL as z_ρ^4, which represents the communication latency tolerance for the application.

5. IoT Application Evaluation

5.1. Smart Eyeglasses to Monitor Eating in Free-Living

We implemented our DynDSE for the design optimisation of 3D-printed regular-looking eyeglasses, which accommodate processing electronics, EMG electrodes, antenna, and power supply. Smart eyeglasses are particularly suited for automated dietary monitoring to unobtrusively detect intake and eating events from activities around the head throughout everyday life, thus replacing classic food journaling and supporting disease management [30,31]. As typical wearable IoT devices, smart eyeglasses could process data locally and provide estimates to other body-worn devices, e.g. smartphones. Furthermore, the monitoring task is a typical example of wearable IoT applications in remote health assistance.

Two pairs of electrodes were symmetrically integrated at the eyeglasses frame, located around the temple ear bends. Contraction of temporalis muscles was monitored bilaterally resulting in two EMG signal channels.

The EMG sensor data stream was segmented into eating and non-eating periods by pattern spotting algorithms. The spotting algorithms were designed to extract features in continuous EMG sensor data and perform one-class classification to identify eating events (i.e., time span between the start and the end of an eating activity), see Section 5.2.1.

5.2. Design Space Representation

We considered $N_{\xi} = 4$ system functionalities: Ξ^1, an algorithm for event retrieval, Ξ^2, a data sampling strategy, Ξ^3, a μC and Ξ^4, a runtime mode.

Table 2 presents the design space considered in this work. Our design space included two spotting algorithms, i.e., $\epsilon_{1,1}$ and $\epsilon_{1,2}$, which were considered for execution on three μCs, i.e., $\epsilon_{3,1}, \epsilon_{3,2}$ and $\epsilon_{3,3}$. Moreover, two data sampling strategies, i.e., $\epsilon_{2,1}$ and $\epsilon_{2,2}$, were considered, while applying two runtime modes, i.e., $\epsilon_{4,1}$ and $\epsilon_{4,2}$.

Table 2. Design space for the EMG-monitoring eyeglasses.

Functionality (Ξ_{ξ})	**Algorithm** (Ξ_1)	**Data Sampling** (Ξ_2)	**Microcontroller** (Ξ_3)	**Runtime Mode** (Ξ_4)
Component ($\epsilon_{\xi,q}$)	FFT-based ($\epsilon_{1,1}$) WPD-based ($\epsilon_{1,2}$)	Uniform ($\epsilon_{2,1}$) Adaptive ($\epsilon_{2,2}$)	PSoC1 M8C ($\epsilon_{3,1}$) TI MSP430F1611 ($\epsilon_{3,2}$) ARM CortexM3 ($\epsilon_{3,3}$)	Real time ($\epsilon_{4,1}$) Online ($\epsilon_{4,2}$)
Parameter ($\omega_{\xi,q,w}$)	Data frame size m ($\omega_{1,1,1}$) Data frame size m ($\omega_{1,2,1}$) Dim. feature space d ($\omega_{1,2,2}$)	Context measure's upper bound θ_h ($\omega_{2,2,1}$)		

To deal with the requirements in this case study, we relaxed the optimisation problem of Equation (18) into one of maximisation as follows:

$$\begin{aligned} \max_{\mathcal{X}|E,\Omega\subseteq\mathcal{X}|\mathcal{E},\boldsymbol{\Omega}} & \sum_{p} \pi^{p}(\mathcal{X}|E,\Omega) \\ \text{s.t. } & z_{\rho}^{r} \leq 1 \;\; \forall r \in \boldsymbol{r}, \\ & z_{\pi}^{p} \geq 1 \;\; \forall p \in \boldsymbol{p}. \end{aligned} \tag{39}$$

The above constrained optimisation problem was solved by a grid search-based approach, evaluating a grid of possible configurations with an exhaustive search. At each iteration, a system configuration was generated and the sensor data processed through the simulation. The metrics were estimated and compared with the respective requirements.

5.2.1. Algorithm (Ξ_1)

FFT-based spotting: The first pattern spotting method was introduced by Zhang and Amft [32] in order to identify eating moments. An online non-overlapping sliding window segmentation with length m expressed in seconds was used to extract features in continuous EMG data. A one-class classification was performed by a one-class SVM (oc-SVM). Details of the feature extraction and one-class classification can be found in [32]. The ocSVM was trained applying the leave-one-participant-out (LOPO) cross-validation strategy. Hyperparameter optimisation was performed using grid search approach.

WPD-based spotting: We designed the second pattern spotting method inspired by the classification task presented in [33]. An online non-overlapping sliding window segmentation with length m expressed in seconds was used. From each segmented frame, the maximum sample value was extracted and compared to a threshold experimentally found. When the sample value was lower than the threshold, the frame was not fed to the spotting pipeline. In the pre-processing module, the EMG signals were passed through a notch filter of 50 Hz to remove power line's interferences, likely to occur in free-living, also de-trended by a digital high pass filter of 20 Hz and rectified. In the feature extraction module, the signal was passed through a Wavelet Packet Decomposition (WPD), to extract c features, i.e., the WPD coefficients, in the time-frequency domain. The depth level of the tree decomposition was kept constant, i.e., $l = 2$. A principal component analysis (PCA) was subsequently used to reduce the number of features from c to d. After normalisation, the features were used as discriminant of the target class by using a ocSVM, with number of support vectors $v = 1500$.

The ocSVM was trained applying the LOPO cross-validation strategy. Hyperparameter optimisation was performed using grid search approach.

Table 3 presents a breakdown of the spotting algorithms.

Table 3. Analytical breakdown of the two spotting algorithms. Each function f belongs to an algorithm stage κ. The variable m denotes the length of the window size. For the WPD-based feature extraction, the variable c denotes the number of WPD coefficients, l denotes the depth level of the WPD decomposition tree, and d denotes the dimensionality of the feature space after applying PCA. For spotting, the variable v is the number of support vectors. The operations for pre-processing, feature extraction, and spotting are calculated on each sliding window's instance.

Algorithm	Stage (κ)	Function (f)	N. Arithmetical Operations						Memory Demand	
			Add	Mult	Div	Root	Comp	Exp	Integers	Floats
Both	Pre-processing	Low-pass filtering	m	-	-	-	-	-	-	1
FFT-based	Feature extraction	Standard deviation	$3m-1$	m	2	1	-	-	1	-
		Fast Fourier trasform	$3m \cdot log_2 m$	$2m \cdot log_2 m$	-	-	-	-	-	k
		Maximum	-	-	-	-	$m-1$	-	1	-
		L2-norm	$d-1$	d	-	1	-	-	1	-
WPD-based	Feature extraction	WPD	$\sum_i^l \frac{(m/2^i)-1}{2+1} \cdot 2^i$	$\sum_i^l \frac{(m/2^i)+1}{4} \cdot 2^i$	-	-	-	-	-	$4+d$
		PCA	$d \cdot (c-1)$	$d \cdot c$	-	-	-	-	-	$c \cdot d$
		L2-norm	$d-1$	d	-	1	-	-	1	-
Both	Spotting	Kernel SVM	$v+1$	$2v$	-	-	1	-	v	$v \cdot (d+1)$
		+ Radial basis kernel	$v \cdot (2d-1)$	$v \cdot (d+1)$	-	-	1	v	-	1

5.2.2. Data Sampling (Ξ_2)

A context-adaptive sampling algorithm needs two main components, i.e., a context measure and a response model, to adapt the sampling rate depending on an estimation of relevance of future samples. As a context measure, we employed a basic representation of EMG signal energy. A feedforward state-based model that alternates between attentive and sleep states was adopted as response model. Our sampling strategy was based on the n-shots measure paradigm: the sensor wakes up, takes n samples, and goes to sleep again. The energy content from the n samples was used to compute the context measure as:

$$e_k = \frac{\sum_{i=1}^{n} s_{i,k}}{n}, \qquad \theta_t = \max\left(e_1, \ldots, e_k, \ldots, e_K\right), \tag{40}$$

where e_k is the signal energy for the k^{th} channel, $s_{i,k}$ is the i^{th} value sampled from the n-shots measurement, θ_t is the context measure, t is the time-step, K is the number of channels that connect to the same system. A linear mapping function converted θ_t to a candidate duty cycle rate D^*_{t+1}:

$$D^*_{t+1} = D_l + \left[\frac{D_h - D_l}{\theta_h - \theta_l} \cdot \theta_t - \theta_l\right], \tag{41}$$

where D_l is the minimum duty rate set to θ_l, which was estimated from the signals noise. A maximum duty rate D_h was set to θ_h. We adjusted the model's sensitivity by tuning θ_h.

The behaviour of the response model is described by Equation (42).

The computation of the duty cycle rate for the next period was based on a comparison between the candidate duty cycle rate D^*_{t+1} and the threshold value D_{TH}.

As the D^*_{t+1} exceeds the threshold D_{TH}, the response model switches from inattentive state to attentive state. The attentive state is characterised by a monotonically increasing duty cycle rate. As the D^*_{t+1} drops

below the threshold D_{TH} and the attention time τ expires, the response model switches back from attentive state to inattentive state. The duty cycle rate's decision rules for the two states were computed as follows:

$$\begin{aligned} &(1)\ \text{Inattentive state} \\ &\quad \text{If } (D^*_{t+1} < D_{TH}) \text{ and } \tau \text{ elapsed}) : \\ &\quad D_{t+1} = D^*_{t+1} \\ &(2)\ \text{Attentive state} \\ &\quad \text{If } (D^*_{t+1} > D_{TH}) : \\ &\quad D_{t+1} = \begin{cases} D_t, & D^*_{t+1} \leq D_t \\ D^*_{t+1}, & D^*_{t+1} > D_t. \end{cases} \end{aligned} \tag{42}$$

Table 4 presents a break down of the context-adaptive sampling algorithm. More details can be found in our previous work [34].

Table 4. Analytical breakdown of the context-adaptive sampling algorithm. The variable n denotes the number of samples taken during the n-shots measure, and g denotes the number of channels. The function is executed at any n-shot measure.

Function (f)	**N. Arithmetical Operations**						**Memory Demand**	
	Add	**Mult**	**Div**	**Root**	**Comp**	**Exp**	**Integers**	**Floats**
Context measure	$2n \cdot g$	-	2	-	$n-1$	-	-	1
Response output	4	1	1	-	-	-	-	4
Attention time	-	-	-	-	2	-	-	2

5.2.3. μC (Ξ_3)

The processing module simulates the behaviour of an energy-efficient μC while computing the algorithm's functions at a certain clock frequency. We considered three commonly widely used μCs, i.e., PSoC1 M8C, TI MSP430F1611, and ARM CortexM3. A Li-Ion polymer battery provided energy to the system. Each μC was provided with memory for local data processing, i.e., Flash and non-volatile memory. Table 5 shows the number of machine cycles n_x^{cyc} related to the arithmetical operation x for different μCs.

Table 5. Number of machine cycles n_x^{cyc} for the arithmetical operation x and memory specifications for the considered μCs.

		Number of Machine Cycles per Operation						**Memory spec.** [kB]	
		n^{cyc}_{Add}	n^{cyc}_{Mult}	n^{cyc}_{Div}	n^{cyc}_{Root}	n^{cyc}_{Comp}	n^{cyc}_{Exp}	ROM/Flash	RAM
μC	PSoC1 M8C	544	560	912	1344	80	2672	32	2
	TI MSP430F1611	177	153	405	668	37	334	48.25	10
	ARM CortexM3	60	50	80	380	12	210	512	96

5.2.4. Runtime mode (Ξ_4)

Two runtime modes were considered in this work, i.e., online and real time. In our application, real time mode implies that as soon as a data frame has been recorded, the output of the data processing must

be available. With online data processing, the temporal requirement is more relaxed, as the output of the data processing must be available by the end of the runtime.

5.3. Configuration Evaluation

5.3.1. Sensor Dataset

Ten healthy volunteers (4 females, 6 males) aged between 20 and 30 years wore the EMG-monitoring eyeglasses for one day. The application data used in the simulation were sensor data collected at uniform sampling rate, i.e., 256 Hz. The eyeglasses were attached after getting up in the morning and kept on till bed time. When a risk of contamination with water existed, the participants were allowed to remove the eyeglasses. Participants manually logged the occurrence of eating events in a diet journal with a one minute resolution.

5.3.2. Multi-Objective Computation

Our framework is based on a simulation which reproduces the functionalities of a wearable IoT system. The sensor's and μC's behaviour can be emulated with a finite-state machine approach, e.g., Buschhoff et al. [35], in order to reproduce dependencies between the system components and between hardware and software.

We evaluated the algorithm's retrieval performance considering the total number of samples of all eating events according to ground truth labels, the total number of samples of all retrieved eating events, and the sum of the number of samples correctly retrieved belonging to eating events segments.

The number of machine cycles for an instance of a oc-SVM for the WPD-based spotting algorithm, with hyper-parameters $(m = 256, d = 20)$, applying Equation (20), is computed as:

- Kernel SVM:
  $[v \cdot 60] + [2v \cdot 50] + [1 \cdot 12] \approx 24 \times 10^4$ cycles, where $v = 1500$, is the number of support vectors;
- Radial basis function:
  $[v(2d-1) \cdot 60] + [v(d-1) \cdot 50] + [1 \cdot 12] + [v \cdot 210] \approx 525 \times 10^4$ cycles.

For simplicity, we assumed that all operations were performed using float data type.

Accordingly to Equation (21), the ET_f is computed from the number of machine cycles n_f^{cyc}, as inversely proportional to the clock frequency ν of a considered μC. Table 6 lists the clock frequencies of the three candidate μCs.

Time ET_f to execute the oc-SVM on a ARM CortexM3 was 114.3 ms.

Accordingly to Equation (24), the $\mathrm{EC}_f^{\mu C}$ is computed multiplying the ET_f and the μC energy consumption $P_{\mathrm{act}}^{\mu C}$. The μC switching behaviour is regulated by the time constraints given by the ET and the sensor data frequency. Table 6 lists the current consumptions $I_{\mathrm{act}}^{\mu C}$ in active state, $I_{\mathrm{stb}}^{\mu C}$ in stand-by state, and the voltage level V, of the three candidate μCs. For example, the energy consumption $\mathrm{EC}_f^{\mu C}$ to execute an instance of the oc-SVM on a ARM CortexM3 was 0.732 µWh.

Table 6. Component electrical characteristics to estimate the EC and MD measures via simulation. All values were extracted from datasheets. The following symbology is used. I_{act}: current consumption in active mode; I_{stb}: current consumption in stand-by mode; V: voltage; ν: frequency; Res.: resolution; Cap.: capacity; M_b: data block's size; I^m_{write}: current consumption for memory writing; t_{write}: time for writing a data block.

Functionality	Component	Electric Characteristics								
								Memory Write Operation		
		I_{act}[mA]	I_{stb}[mA]	V	ν[MHz]	Res.[bit]	Cap.[mWh]	M_b[Byte]	I^m_{write}[µA]	t_{write}[ms]
μC (Ξ^3)	PSoC1 M8C ($\epsilon^{3,1}$)	8.0	0.025	3.3	24	8	-	64	619.5	1.5
	TI MSP430F1611 ($\epsilon^{3,2}$)	0.57	0.05	3	8	16	-	60	2300	23.0
	ARM CortexM3 ($\epsilon^{3,3}$)	7.0	0.55	3.3	48	32	-	256	500	3.28
Data Sampling (Ξ^2)	EMG sensing	4.0	0.008	3.3	256×10^{-6}	-	-			
Power supply	Li-Ion polymer battery	-	-	3.7	-	-	925			

Figure 2a,b show the linear relation between $\sum_f \mathrm{ET}_f$ and $\sum_f \mathrm{EC}^{\mu C}_f$, for the individual spotting stage κ. A log scale for the y axes was defined in order to make the plot more readable. A tuning effect of the spotting parameters on the execution time and on the energy consumption, is evident. The higher complexity of a parameter combination, the more machine cycles n^{cyc}_f, execution time ET_f, and energy consumption EC_f.

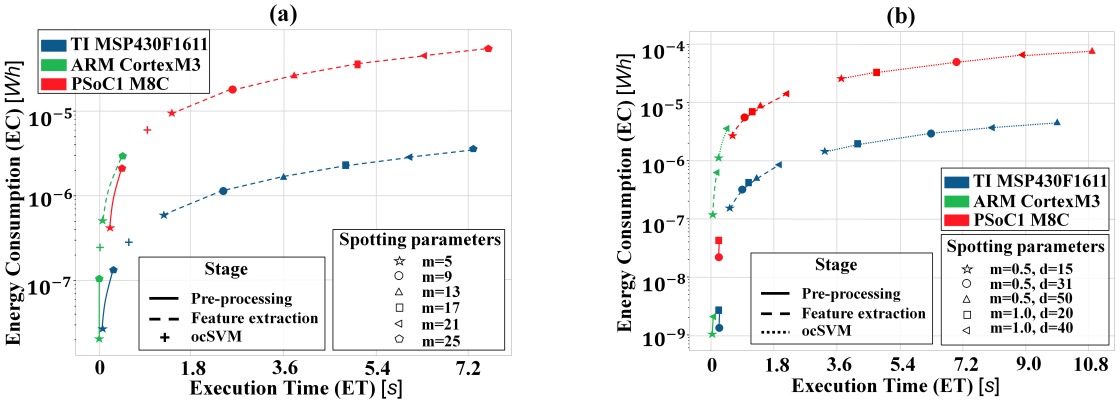

Figure 2. (**a**) FFT-based spotting: Relation between execution time and energy consumption. Some of the spotting parameters were omitted for readability. The spotting parameter m represents the data frame size. (**b**) WPD-based spotting. Relation between execution time and energy consumption. The spotting parameter m represents the data frame's size and d represent the reduced feature space's dimension.

Tables 5 and 6 list the memory specifications and the data resolution for the three candidate μCs. Each μC had a RAM memory and a larger flash support, and their capacity represented a system constraint.

By switching runtime mode, i.e., real-time mode or online mode, m_d was defined as follows. In real-time mode, the m_d was defined as the amount of data memory for a window size m. In online mode, the m_d was estimated as the peak of memory footprint required to process the data stream continuously without data loss, using a ring buffer.

Depending on their characteristics, the considered μCs had a certain latency that shaped the distribution of their memory footprint. For example, Figure 3 shows the distribution of the memory m_d footprint related to a daylong processing. The BLE transmission's specifications defined the maximum ν_{tr} as 305 kbps and the MPS as 215 bits [36]. Equation (38) defines the actual ν_{tr} by tuning the $connInterval$ parameter, which ranged from 7.5 ms to 4.0 s with steps of 1.25 ms. For example, a retrieved eating event

was represented by two time stamps, which represent the start and end of an eating event. A time stamp can be represented as an unsigned short variable type in 2 bytes (i.e., 5 bits for day, 5 bits for hour, and 6 bits for minute). The communication latency CL to deliver an event retrieved by the FFT-based spotting (m = 13) on a ARM CortexM3 was 886 ms, according to Equation (37).

Considering the typical event frequencies in human daily behaviour and the negligible memory footprint for event information, communication latency was omitted from the following analyses.

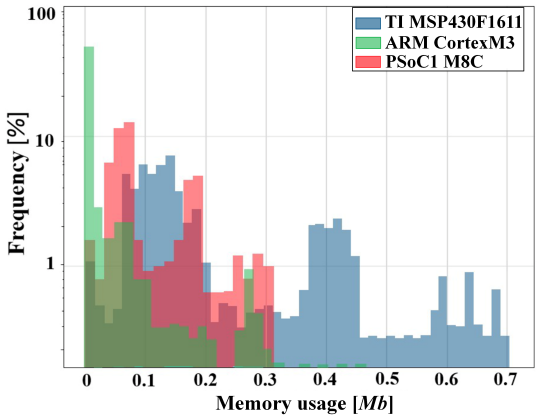

Figure 3. Memory m_d foot-print for the online data buffer while executing the WPD-based spotting. Each colour is related to a specific μC.

5.3.3. Multi-Objective Visualisation

To visualise whether the metrics lie within the system requirements' boundary conditions or not, we exploited radar plots. Radar plots support the representation of the trade-off across the design space. Each y-axes is related to an objective, i.e., P, R, EC, ET, and MD, and normalised for the respective requirements, yielding a feasibility region within the unitary axes. Thus, a radar plot point beyond the unitary axes indicates unacceptable configurations. The unitary axes represents an upper bound regarding the requirements.

5.3.4. Multi-Objective Analysis

For FFT-based, by tuning the parameter m, the precision's variation ranged from 48.8% for $m = 1$, to 95.7% for $m = 33$. The recall R variation ranged from 62.9% for $m = 29$, to 99.2% for $m = 1$. The F1-score variation ranged from 63.3% for $m = 1$, to 85.2% for $m = 9$. For WPD-based, by tuning the parameters m and d the precision P variation ranged from 38.7% for $m = 0.25, d = 50$, to 95.7% for $m = 0.5, d = 15$. The recall R variation ranged from 52.7% for $m = 0.25, d = 28$, to 86.2% for $m = 1, d = 40$. The F1-score variation ranged from 49.1.0% for $m = 0.25, d = 50$, to 85.9% for $m = 1, d = 40$. The system requirements z_π^1 and z_π^2 are indicated as horizontal lines. It is evident that only a subset of spotting parameters respects the requirements.

Figure 4a,b show the results of the cross-validated FFT-based and WPD-based spotting algorithms in uniform sampling mode, when tuning their spotting parameters. Specifically, precision P, recall R, and F1-score, are reported for all spotting parameters combinations. The degree of sampling reduction was changed by tuning θ_h in order to find a balance between retrieval performance and resource consumption. Figure 5a,b show the trade-off between F1-score and sampling reduction for the FFT-based and WPD-based spotting algorithms. It is clear that we can keep the retrieval performance up over 80% while performing a sampling reduction over 70%, in both methods. Comparing the two methods, the WPD-based spotting

appears to be more robust against the down-sampling effect, presenting a lower degradation of retrieval performance for a given sampling reduction degree.

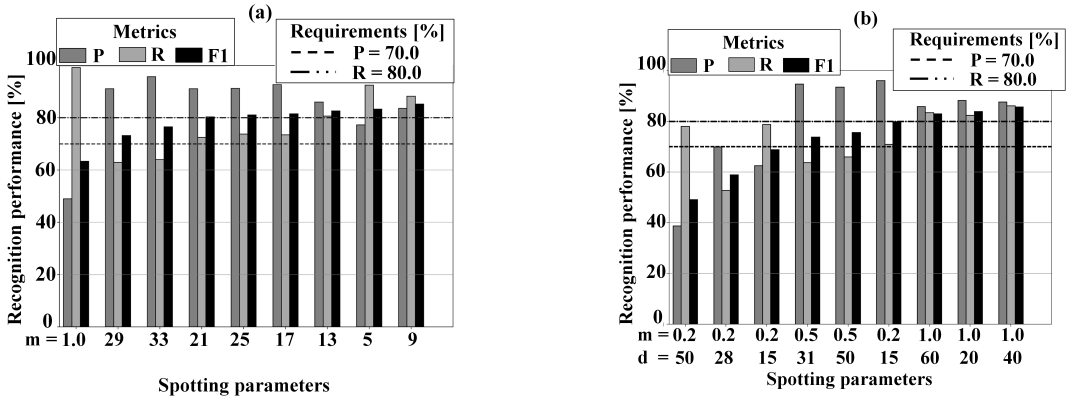

Figure 4. (**a**) FFT-based spotting. Average retrieval performance when varying the spotting parameters in uniform sampling mode. Bars were sorted by increasing retrieval performance. The spotting parameter m represents the data frame size. (**b**) WPD-based spotting. Average retrieval performance when varying the spotting parameters in uniform sampling mode. Bars were sorted by increasing retrieval performance. The spotting parameter m represents the data frame's size and d represent the reduced feature space's dimension.

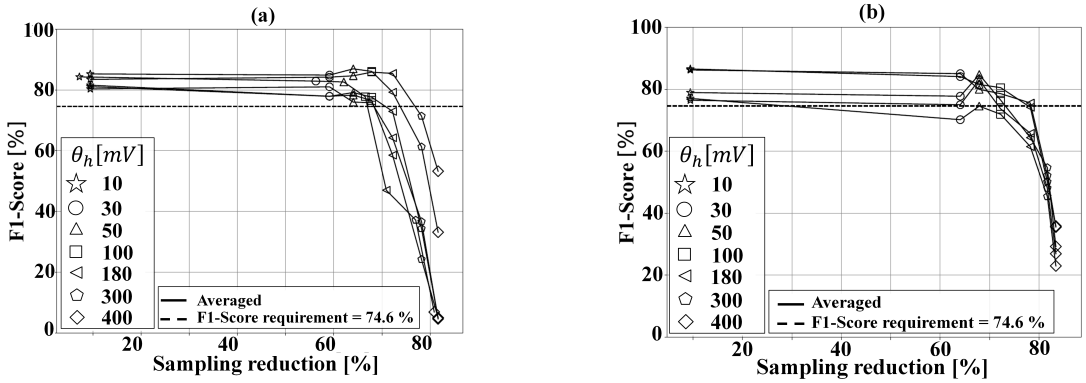

Figure 5. (**a**) FFT-based spotting. Average retrieval performance vs. sampling reduction when varying θ_h. Individual lines correspond to the spotting parameters, which respect P and R requirements in Table 1. The F1-score requirement is derived by the same P and R requirements. The used parameters for the context-adaptive sampling were: $D_h = 1$, $D_l = 0.1$, $D_{TH} = 0.6$, $n = 4$, $\tau = 3\,\text{s}$, $\theta_l = 10\,\text{mV}$. (**b**) WPD-based spotting. Average retrieval performance vs. sampling reduction when varying θ_h. Individual lines correspond to the spotting parameters that respect P and R requirements in Table 1. The F1-score requirement is derived from the same P and R requirements. The used parameters for the context-adaptive sampling were: $D_h = 1$, $D_l = 0.1$, $D_{TH} = 0.6$, $n = 4$, $\tau = 3\,\text{s}$, $\theta_l = 10\,\text{mV}$.

From Figures 6–11, the trade-off across the metrics are shown for different system configurations. Figures 6 and 7 show results for the FFT-based spotting in real-time for uniform and context-adaptive sampling. In uniform sampling, i.e., Figure 6, the largest requirement breach on any μC was due to the energy consumption EC. In context-adaptive sampling, i.e., Figure 7, the reduction of the energy consumption EC determined a set of feasible configurations on the ARM Cortex M3. In both sampling modalities, the execution time ET fulfilled the real-time requirements on any μC, although the memory demand MD compromised the feasibility of the configurations on the PSoC1 M8C.

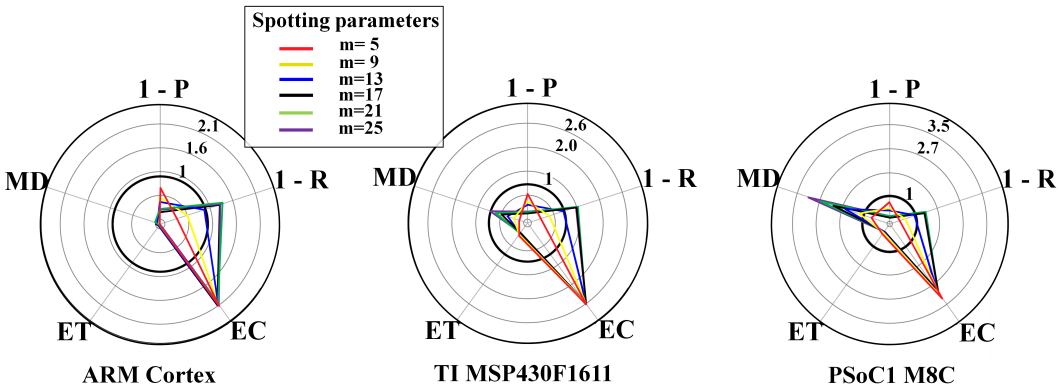

Figure 6. FFT-based spotting and uniform sampling. Resource-performance trade-off for real-time mode, including different μCs: ARM CortexM3 (**left**), TI MSP430F1611 (**center**), PSoC1 M8C (**right**). List of objectives: P = precision, R = recall, EC = energy consumption, ET = execution time, MD = memory demand.

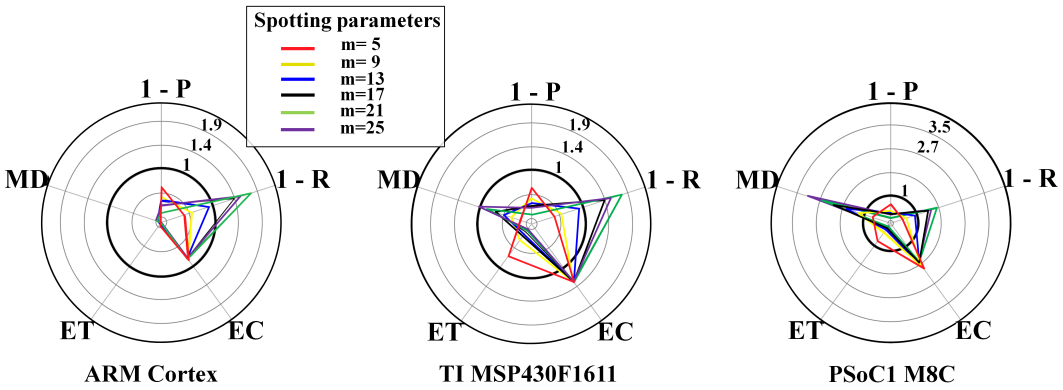

Figure 7. FFT-based spotting and context-adaptive sampling. Resource-performance trade-off for real-time mode, including different μCs: ARM CortexM3 (**left**), TI MSP430F1611 (**center**), PSoC1 M8C (**right**). List of metrics: P = precision, R = recall, EC = energy consumption, ET = execution time, MD = memory demand. Context-adaptive sampling parameters are: $D_h = 1$, $D_l = 0.1$, $D_{TH} = 0.6$, $n = 4$, $\tau = 3\,s$, $\theta_l = 10$ mV, $\theta_h = 180$ mV.

Figures 8 and 9 show results for the WPD-based spotting in real-time for uniform and context-adaptive sampling. In uniform sampling, i.e., Figure 8, the largest requirement breaches on any μC were due to the execution time ET and the energy consumption EC. In context-adaptive sampling, i.e., Figure 9, the reduction of the execution time ET determined a set of feasible configurations on the ARM Cortex M3. Overall, the memory demand MD was neglectable in all configurations, as only a data frame had to be stored.

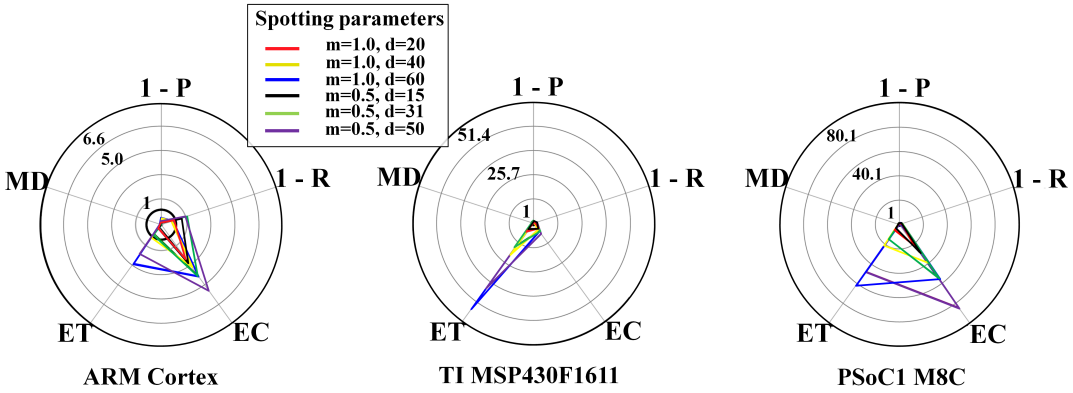

Figure 8. WPD-based spotting and uniform sampling. Resource-performance trade-off for real-time mode, including different μCs: ARM CortexM3 (**left**), TI MSP430F1611 (**center**), PSoC1 M8C (**right**). List of metrics: P = precision, R = recall, EC = energy consumption, ET = execution time, MD = memory demand.

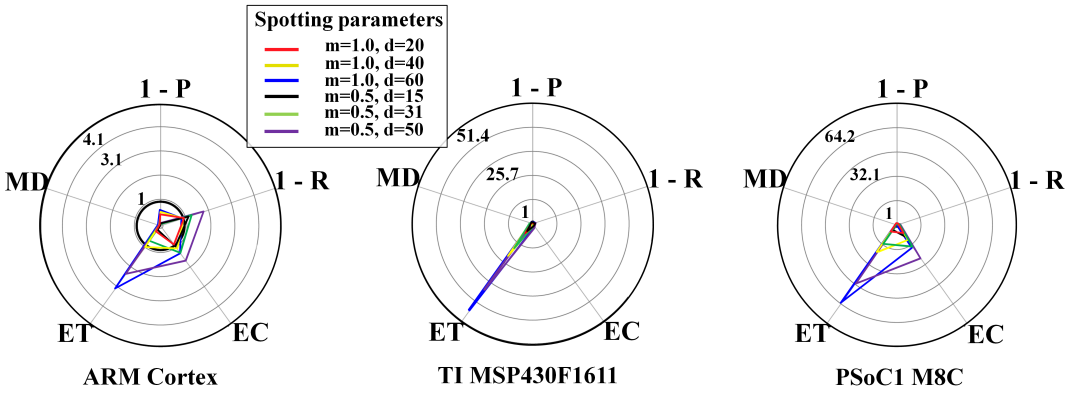

Figure 9. WPD-based spotting and adaptive sampling. Resource-performance trade-off for real-time mode, including different μCs: ARM CortexM3 (**left**), TI MSP430F1611 (**center**), PSoC1 M8C (**right**). List of metrics: P = precision, R = recall, EC = energy consumption, ET = execution time, MD = memory demand. The used parameters are: $D_h = 1$, $D_l = 0.1$, $D_{TH} = 0.6$, $n = 4$, $\tau = 3\,\text{s}$, $\theta_l = 10\,\text{mV}$, $\theta_h = 180\,\text{mV}$.

Figures 10 and 11 show results for the WPD-based spotting in online mode for uniform and context-adaptive sampling. The online mode required a more intensive memory use, due to the longer ring buffer, that in turn made the z^3_ρ a more stringent requirement. In uniform sampling, i.e., Figure 10, the largest requirement breaches on any μC were due to the energy consumption EC and the memory demand MD. In context-adaptive sampling, i.e., Figure 11, the reduction of the energy consumption EC determined a set of feasible configurations on the ARM Cortex M3.

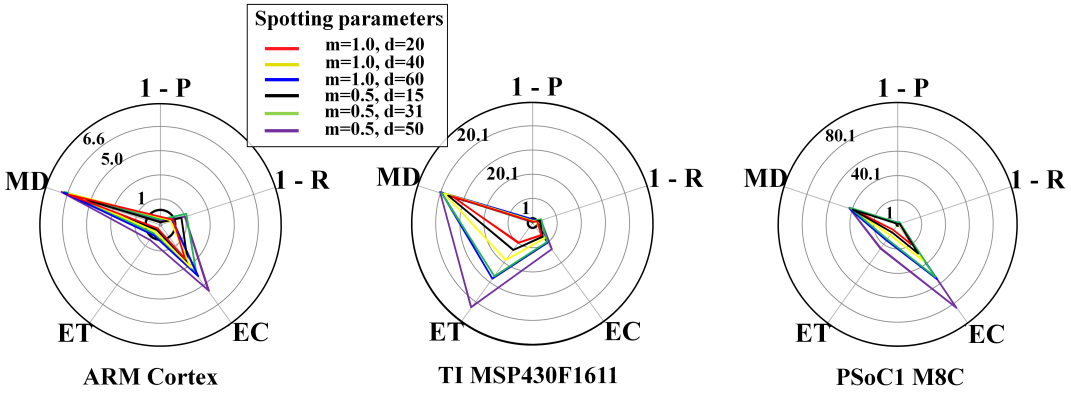

Figure 10. WPD-based spotting and uniform sampling. Resource-performance trade-off for online mode, including different μCs: ARM CortexM3 (**left**), TI MSP430F1611 (**center**), PSoC1 M8C (**right**). List of metrics: P = precision, R = recall, EC = energy consumption, ET = execution time, MD = memory demand.

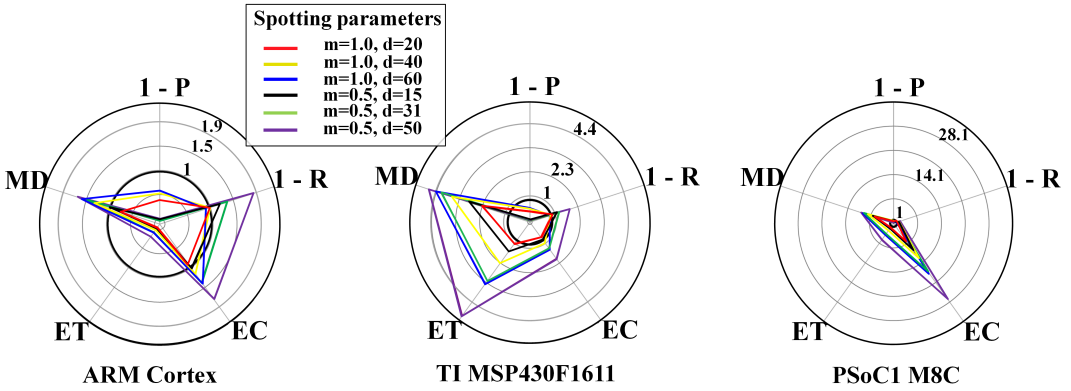

Figure 11. WPD-based spotting and context-adaptive sampling. Resource-performance trade-off for online mode, including different μCs: ARM CortexM3 (**left**), TI MSP430F1611 (**center**), PSoC1 M8C (**right**). List of metrics: P = precision, R = recall, EC=energy consumption, ET = execution time, MD = memory demand. The used parameters are: $D_h = 1$, $D_l = 0.1$, $D_{TH} = 0.6$, $n = 4$, $\tau = 3\,\text{s}$, $\theta_l = 10\,\text{mV}$, $\theta_h = 180\,\text{mV}$.

Figure 12 depicts the energy consumption estimated from the best system configuration, related to the individual participants. For three participants, the boundary conditions are not respected, implying a reduced application's runtime with respect to the system requirements.

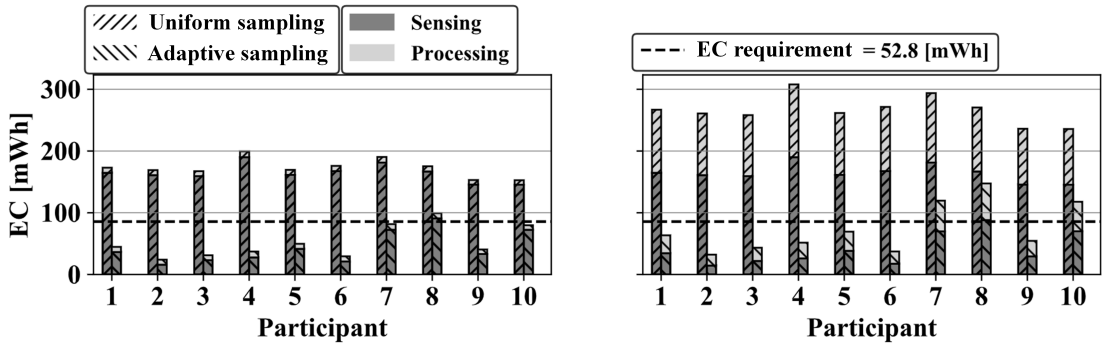

Figure 12. Estimated energy consumption (EC) for the individual participants on the TI MSP430F1611. **Left**: FFT-based spotting with $m = 13$. **Right**: WPD-based spotting with $(m = 256, d = 20)$.

6. Discussion

The simulation-based DynDSE presented here targets wearable IoT device design, which run time-variable recognition algorithms on the device. Processing a large volume of data locally and enabling local inference is key to a scalable IoT network. Furthermore, manual tuning of hardware and algorithms in a physical implementation is tedious. The DynDSE simulation-based approach can cover a wide configuration space to identify a balance between resources and event retrieval performance. Our case study demonstrated interactions and dependencies among hardware and algorithm components, and justified the need for co-designing and developing of associated functionalities. We chose two example retrieval algorithms to illustrate different effects on the optimisation result (cf. Figures 9–12). FFT-based oc-SVM spotting was published before [32]. The WPD-based spotting showed performance improvements P and R, but also implications for execution time ET, thus further illustrating the design trade-off features of our methodology. The processing functionality (Ξ_3) affected the execution time ET, due to the μC type and speed, and the memory demand MD, due to its memory capacity. The μC's energy consumption EC heavily depended on the algorithmic complexity, see Figure 12. Executing the WPD-based spotting, the μC's energy consumption EC was comparable with the sensor's energy consumption EC. As the computational complexity was shrunk by employing the FFT-based spotting, the energy consumption EC became neglectable. The algorithm (Ξ_1) and data sampling (Ξ_2) functionality were the most influencing configuration elements on the system's metrics. Tuning the algorithm parameters affected precision P and recall R, and the required memory demand MD. Data sampling had the highest impact on the energy consumption EC and the tuning of the algorithm parameters had the lowest.

We found that our context-adaptive sampling strategy kept the performance of the spotting pipeline at a average F1-score over 80% while reaching almost 70% reduction in resource consumption.

DynDSE requirements in the application evaluation were set according to Table 1. While in our analysis, retrieval performances (P, R) larger than 80% were reached for optimised parameter settings, our requirements z_π^1 and z_π^2 followed literature recommendations [37] suggesting that even 70% in retrieval performance has relevant application value. Energy consumption requirement z_ρ^2 was set considering the capacity and size of standard lithium-ion batteries, and the application runtime, according to Equation (35). Execution time and memory demand requirements z_ρ^1 and z_ρ^3 were dictated by the algorithm and the μC characteristics, respectively. While the retrieved system configurations and their performances appear relevant, a direct comparison to prior work is not feasible, due to the diversity in analysis goals, applications, and dataset characteristics. First, many investigations optimise for a fraction of the DynDSE metrics only, e.g., recognition performance. Second, sensor and algorithm choice span a wide value space for performance metrics. Our investigation aimed at defining a generalisable procedure, which provides trade-off indicators across a variety of design space options and could thus assist designers in taking decisions and investigate details depending on application relevance.

Figures 7 and 8 show the analysis of the variance in the resource-perfomance trade-off. Under the same P and R, higher sampling reduction can be achieved, which corresponds to lower energy consumption EC. In context-adaptive sampling mode, the resource consumption is proportional to the event frequency and the duration of event patterns. Consequently, for one configuration, resource saving varies according to individual behaviour. Population-averaged models do not guarantee to fulfil the system requirements for every individual. In our case study, a homogeneous study group of university students was included, however the resource consumption estimation did not respect the boundary conditions for all participants, as shown in Figure 12.

Personalising models increases computational complexity and entails more complicated deployment. A reasonable approach is to take into account the heterogeneity of the population by defining subpopulations having similar behaviour and include a safety margin.

We derived approximate machine cycle numbers, which limit accuracy of execution time estimation. The exact number of cycles is highly dependent on the algorithm implementation and compiler. Thus, our analysis could be integrated with extended target-dependent hardware and machine instruction simulators. Another source of inaccuracy are the energy consumption measures, as we did not consider overhead of the electronic circuits. We considered for simplicity only floating point operations. Differentiating between integer and floating point operations would result in higher modelling accuracy. Also, differentiating the range of variables and datatypes may improve the modelling.

The metric set serves as mapping between the design space and the application specifications, whose definition largely depends on the application requirements. Therefore, a direct comparison of metric outcomes is limited. However, depending on the defined metric set, well-determined functional implications and system properties can be identified and compared.

For example, Bharatula et al. [14] defined four conflicting metrics and analysed the inherent trade-off on an activity recognition task: Flexibility, which included estimation of memory demand and μC's operating frequency, electronic packaging, relative recognition performance, and energy consumption measures. The conflicting nature of the four metrics was highlighted as orthogonal, meaning that optimising all the four metrics at the same time is not feasible. Similar to our work, the authors embedded recognition algorithm characteristics in the trade-off analysis, namely classification accuracy. However, we included execution time ET to highlight dynamic design aspects that appear during runtime. The ET metric linked the algorithmic computational complexity with the hardware μC characteristics in time. Understanding of the system temporal constraints enables DynDSE to leverage dynamic system behaviour for context-adaptivity. Azariardi's [38] DSE included ET and classification accuracy in the metric set but omitted energy consumption. The authors were able to estimate temporal constraints for SVM processing and investigated how the DSE solution matches with application requirements and free-living user. However, assumptions were needed to compensate for the missing energy consumption (EC) metric. The application of Beretta et al. [17] consisted of a wearable node transmitting sensor data using compressive sensing. Objectives were the node's energy consumption, the percentage root-mean-square difference (PRD) to approximate the information loss due to compression, the communication delay and the packet error rate (PER) of the radio transmission. The solution space was compared with the one reported by Kumar et al. [39], which optimised only energy consumption and communication delay. Under the same energy consumption and communication delay solution, the PRD and PER were significantly higher. Moreover, Kumar et al. were able to discover only the 2.3% of the solution space with respect to Beretta's work.

From the above comparison, it appears that the descriptive power of the trade-off analysis in DSE depends on a careful selection of the metrics. Neglecting metrics may result in misleading results. The same conclusion can be drawn with regard to our work. For example, consider Figures 9 and 10: When omitting EC, it may seem that the choice of sampling mode does not affect the system behaviour. As the EC is included in our trade-off analysis, it becomes evident how the uniform sampling mode does not provide any feasible configuration. DSE frameworks that consider hardware-software co-design, in principle, achieve higher system performances as a consequence of the flexibility given by a finer model granularity. For example, Shoaib et al. [40] optimised the individual processing stages of a SVM pipeline by exploring hardware architectures based on custom instructions and coprocessor computations. The authors reported a reduction in energy consumption of almost three orders of magnitude compared to that of a low-power μC, as targeted by our work. The energy consumption metric was computed as the sum of several real measurements related to hardware components involved in the SVM processing stage. The design space solutions included specific hardware to run kernel-based classification in varying contexts. The optimisation potential of hardware-software co-design comes at the cost of an expensive

design, which includes custom-made platforms, and design space and metrics definition that rely on hardware-specific knowledge.

Overall, dynamic system configurations have been rarely considered in DSE for wearable systems. The inclusion of data sampling strategies into the design space enabled us to adapt system designs to context. Moreover, memory demand has been infrequently included into the DSE objective set, although memory limitations are common in μCs and represent a bottleneck for embedded recognition algorithm deployment, as evident from Figures 9 and 10. We argue that memory demand should be considered in the design phase.

This work focuses on one typical wearable IoT application in order to derive a detailed analysis of the design space spanned by two retrieval algorithms, three μCs, and two sampling procedures introducing dynamic variations. Nevertheless, we kept the DynDSE design space formalism general, such that a wide variety of other components, system architectures, metrics, and IoT applications could be explored, including other hardware, data, and recognition algorithms. Thus, the DynDSE approach does not depend on the particular application considered nor does the method require modifications for other applications. Rather, we deem it essential to match the DynDSE approach with appropriate sensor data to drive the simulation.

For larger design spaces than the one considered here, DynDSE may require approximate rules. Nevertheless, the exhaustive search deployed here remains a suitable option for coarse design selections before investigating further design variables in subsequent, local explorations.

7. Conclusions and Future Work

We introduced a general methodology for multi-objective DynDSE applied to context-adaptive wearable IoT edge devices, which retrieve events from streaming sensor data using pattern recognition algorithms. We provided a formal characterisation of the configuration space given a set of system functionalities, components and their parameters. A constrained optimisation problem was formulated to identify an optimal system configuration according to application-dependent system requirements. The simulation can provide crucial information about the compatibility of system components. The method is particularly suitable to analyse design options at an early stage of the development process, to approximate key system design aspects, e.g., size of wireless battery powered devices, to confirm software and hardware choices under given design constraints, and to review designs under varying data patterns.

Further investigations may consider automated, on-demand resource distribution between functions of an embedded system that incorporates the DynDSE methodology. Dynamic resource management may result in wearable IoT systems that reconfigure themselves at runtime according to dynamic conditions. Furthermore, the increasing ubiquity and interconnection among wearable IoT devices rise concerns about security and privacy, as malicious interactions are more likely to happen. System security objectives could be incorporated into the dynamic optimisation to represent varying privacy concerns. Nevertheless, further research is needed to effectively quantify security and privacy concerns in metrics.

Author Contributions: G.S. and O.A. devised the methodology. R.Z. performed data curation. G.S. and J.C.S. implemented the algorithms. O.A. provided feedback throughout the implementation phase. G.S. and O.A. prepared the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially funded by the EU H2020 MSCA ITN ACROSSING project (GA no. 616757).

Acknowledgments: The present study was performed in (partial) fulfillment of the requirements for obtaining the degree “Dr. rer. biol. Hum”.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results’.

Abbreviations

The following abbreviations are used in this manuscript:

μC	Microcontroller
BLE	Bluetooth Low Energy
CL	Communication latency
DSE	Design space exploration
EC	Energy consumption
ECG	Electrocardiography
EEG	Electroencephalography
EMG	Electromyography
ET	Execution time
FFT	Fast Fourier transform
FPGA	Field Programmable Gate Array
GPU	Graphics Processing Unit
IoT	Internet of things
IP	Intellectual Property
LOPO	Leave-one-participant-out
MPS	Maximum payload size
ocSVM	One-class support vector machines
P	Precision
PCA	Principal component analysis
R	Recall
SVM	Support vector machines
WPD	Wavelet packet decomposition

References

1. Amft, O. How wearable computing is shaping digital health. *IEEE Pervasive Comput.* **2018**, *17*, 92–98. [CrossRef]
2. Verma, P.; Sood, S.K. Fog assisted-IoT enabled patient health monitoring in smart homes. *IEEE Internet Things J.* **2018**, *5*, 1789–1796. [CrossRef]
3. Tokognon, C.A.; Gao, B.; Tian, G.Y.; Yan, Y. Structural health monitoring framework based on Internet of Things: A survey. *IEEE Internet Things J.* **2017**, *4*, 619–635. [CrossRef]
4. Sittón-Candanedo, I.; Alonso, R.S.; Corchado, J.M.; Rodríguez-González, S.; Casado-Vara, R. A review of edge computing reference architectures and a new global edge proposal. *Future Gener. Comput. Syst.* **2019**, *99*, 278–294.
5. Selvaraj, S.; Sundaravaradhan, S. Challenges and opportunities in IoT healthcare systems: A systematic review. *SN Appl. Sci.* **2020**, *2*, 139. [CrossRef]
6. Jayakumar, H.; Raha, A.; Kim, Y.; Sutar, S.; Lee, W.S.; Raghunathan, V. Energy-efficient system design for IoT devices. In Proceedings of the 2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC), Macau, China, 25–28 January 2016; pp. 298–301.
7. Belwal, M.; Sudarshan, T. A survey on design space exploration for heterogeneous multi-core. In Proceedings of the 2014 International Conference on Embedded Systems (ICES), Coimbatore, India, 3–5 July 2014; pp. 80–85.
8. Pimentel, A.D. Exploring exploration: A tutorial introduction to embedded systems design space exploration. *IEEE Des. Test* **2016**, *34*, 77–90. [CrossRef]
9. Schiboni, G.; Amft, O. Saving energy on wrist-mounted inertial sensors by motion-adaptive duty-cycling in free-living. In Proceedings of the 2018 IEEE 15th International Conference on Wearable and Implantable Body Sensor Networks (BSN), Las Vegas, NV, USA, 4–7 March 2018; pp. 197–200.
10. Brandolese, C.; Fornaciari, W.; Pomante, L.; Salice, F.; Sciuto, D. Affinity-driven system design exploration for heterogeneous multiprocessor SoC. *IEEE Trans. Comput.* **2006**, *55*, 508–519.

11. Streichert, T.; Glaß, M.; Haubelt, C.; Teich, J. Design space exploration of reliable networked embedded systems. *J. Syst. Archit.* **2007**, *53*, 751–763. [CrossRef]
12. Haubelt, C.; Schlichter, T.; Keinert, J.; Meredith, M. SystemCoDesigner: Automatic design space exploration and rapid prototyping from behavioral models. In Proceedings of the 45th Annual Design Automation Conference. ACM, Anaheim, CA, USA, 8–13 June 2008; pp. 580–585.
13. Bharatula, N.B.; Stäger, M.; Lukowicz, P.; Tröster, G. Empirical study of design choices in multi-sensor context recognition systems. In Proceedings of the IFAWC 2nd International Forum on Applied Wearable Computing, Zurich, Switzerland, 17–18 March 2005; pp. 79–93.
14. Bharatula, N.B.; Anliker, U.; Lukowicz, P.; Tröster, G. Architectural tradeoffs in wearable systems. In Proceedings of the International Conference on Architecture of Computing Systems, Anaheim, CA, USA, 8–13 June 2008; pp. 217–231.
15. Anliker, U.; Beutel, J.; Dyer, M.; Enzler, R.; Lukowicz, P.; Thiele, L.; Troster, G. A systematic approach to the design of distributed wearable systems. *IEEE Trans. Comput.* **2004**, *53*, 1017–1033.
16. Anliker, U.; Junker, H.; Lukowicz, P.; Tröster, G. Design methodology for context-aware wearable sensor systems. In Proceedings of the International Conference on Pervasive Computing, Berlin, Germany, 8 May 2005; pp. 220–236.
17. Beretta, I.; Rincon, F.; Khaled, N.; Grassi, P.R.; Rana, V.; Atienza, D.; Sciuto, D. Model-based design for wireless body sensor network nodes. In Proceedings of the 2012 13th Latin American Test Workshop (LATW), Quito, Ecuador, 10–13 April 2012; pp. 1–6.
18. Stäger, M.; Lukowicz, P.; Tröster, G. Power and accuracy trade-offs in sound-based context recognition systems. *Pervasive Mob. Comput.* **2007**, *3*, 300–327.
19. Rault, T.; Bouabdallah, A.; Challal, Y.; Marin, F. A survey of energy-efficient context recognition systems using wearable sensors for healthcare applications. *Pervasive Mob. Comput.* **2017**, *37*, 23–44. [CrossRef]
20. Mesin, L. A neural algorithm for the non-uniform and adaptive sampling of biomedical data. *Comput. Biol. Med.* **2016**, *71*, 223–230. [CrossRef] [PubMed]
21. Scarabottolo, I.; Alippi, C.; Roveri, M. A spectrum-based adaptive sampling algorithm for smart sensing. In Proceedings of the 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), San Francisco, CA, USA, 4–8 August 2017; pp. 1–8.
22. Rieger, R.; Taylor, J.T. An adaptive sampling system for sensor nodes in body area networks. *IEEE Trans. Neural Syst. Rehabil. Eng.* **2009**, *17*, 183–189. [CrossRef] [PubMed]
23. Künzli, S.; Thiele, L.; Zitzler, E. Modular design space exploration framework for embedded systems. *IEEE Proc.-Comput. Digit. Tech.* **2005**, *152*, 183–192. [CrossRef]
24. Gajski, D.D.; Vahid, F.; Narayan, S.; Gong, J. System-level exploration with SpecSyn. In Proceedings of the 35th Annual Design Automation Conference, San Francisco, CA, USA, 15–19 June 1998; pp. 812–817.
25. Zhuge, Q.; Shao, Z.; Xiao, B.; Sha, E.H.M. Design space minimization with timing and code size optimization for embedded DSP. In Proceedings of the 1st IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, Newport Beach, CA, USA, 1–3 October 2003; pp. 144–149.
26. Rajagopal, S.; Cavallaro, J.R.; Rixner, S. Design space exploration for real-time embedded stream processors. *IEEE Micro* **2004**, *24*, 54–66. [CrossRef]
27. Powers, D.M. Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. *Mach. Learn. Technol.* **2011**, *2*, 37–63.
28. Konstantakos, V.; Chatzigeorgiou, A.; Nikolaidis, S.; Laopoulos, T. Energy consumption estimation in embedded systems. *IEEE Trans. Instrum. Meas.* **2008**, *57*, 797–804. [CrossRef]
29. Prayati, A.; Antonopoulos, C.; Stoyanova, T.; Koulamas, C.; Papadopoulos, G. A modeling approach on the TelosB WSN platform power consumption. *J. Syst. Softw.* **2010**, *83*, 1355–1363. [CrossRef]
30. Zhang, R.; Amft, O. Monitoring chewing and eating in free-living using smart eyeglasses. *IEEE J. Biomed. Health Inform.* **2017**, *22*, 23–32. [CrossRef]

31. Schiboni, G.; Amft, O. Automatic dietary monitoring using wearable accessories. In *Seamless Healthcare Monitoring*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 369–412.
32. Zhang, R.; Amft, O. Free-living eating event spotting using EMG-monitoring eyeglasses. In Proceedings of the 2018 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI), Las Vegas, NV, USA, 4–7 March 2018; pp. 128–132.
33. Chu, J.U.; Moon, I.; Lee, Y.J.; Kim, S.K.; Mun, M.S. A supervised feature-projection-based real-time EMG pattern recognition for multifunction myoelectric hand control. *IEEE/ASME Trans. Mechatron.* **2007**, *12*, 282–290. [CrossRef]
34. Schiboni, G.; Suarez, J.C.; Zhang, R.; Amft, O. Attention-Based Adaptive Sampling for Continuous EMG Data Streams. In Proceedings of the 2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), Leicester, UK, 19–23 August 2019; pp. 1178–1183.
35. Buschhoff, M.; Günter, C.; Spinczyk, O. A unified approach for online and offline estimation of sensor platform energy consumption. In Proceedings of the 2012 8th International Wireless Communications and Mobile Computing Conference (IWCMC), Limassol, Cyprus, 27–31 August 2012; pp. 1154–1158.
36. Mikhaylov, K. Simulation of network-level performance for Bluetooth Low Energy. In Proceedings of the 2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC), Washington, DC, USA, 2–5 September 2014; pp. 1259–1263.
37. Zhang, R.; Bernhart, S.; Amft, O. Diet eyeglasses: Recognising food chewing using EMG and smart eyeglasses. In Proceedings of the 2016 IEEE 13th International Conference on Wearable and Implantable Body Sensor Networks (BSN), San Francisco, CA, USA, 14–17 June 2016; pp. 7–12.
38. Azariadi, D.; Tsoutsouras, V.; Xydis, S.; Soudris, D. ECG signal analysis and arrhythmia detection on IoT wearable medical devices. In Proceedings of the 2016 5th International Conference on Modern Circuits and Systems Technologies (MOCAST), Thessaloniki, Greece, 12–14 May 2016; pp. 1–4.
39. Kumar, G.S.A.; Manimaran, G.; Wang, Z. End-to-end energy management in networked real-time embedded systems. *IEEE Trans. Parallel Distrib. Syst.* **2008**, *19*, 1498–1510. [CrossRef]
40. Shoaib, M.; Jha, N.K.; Verma, N. Algorithm-driven architectural design space exploration of domain-specific medical-sensor processors. *IEEE Trans. Very Large Scale Integr. VLSI Syst.* **2012**, *21*, 1849–1862. [CrossRef]