Journal of
Low Power Electronics and Applications

MDPI

Article

Hardware Solutions for Low-Power Smart Edge Computing

Lucas Martin Wisniewski [1,†], Jean-Michel Bec [1,†], Guillaume Boguszewski [1] and Abdoulaye Gamatié [2,*,†]

[1] CYLEONE S.A.S. Company, 34090 Montpellier, France
[2] LIRMM, University Montpellier, CNRS, 34095 Montpellier, France
* Correspondence: abdoulaye.gamatie@lirmm.fr
† These authors contributed equally to this work.

Abstract: The edge computing paradigm for Internet-of-Things brings computing closer to data sources, such as environmental sensors and cameras, using connected smart devices. Over the last few years, research in this area has been both interesting and timely. Typical services like analysis, decision, and control, can be realized by edge computing nodes executing full-fledged algorithms. Traditionally, low-power smart edge devices have been realized using resource-constrained systems executing machine learning (ML) algorithms for identifying objects or features, making decisions, etc. Initially, this paper discusses recent advances in embedded systems that are devoted to energy-efficient ML algorithm execution. A survey of the mainstream embedded computing devices for low-power IoT and edge computing is then presented. Finally, CYSmart is introduced as an innovative smart edge computing system. Two operational use cases are presented to illustrate its power efficiency.

Keywords: smart edge computing; energy-efficiency; Internet-of-Things; low-power embedded systems; machine learning; CYSmart

Citation: Martin Wisniewski, L.; Bec, J.-M.; Boguszewski, G.; Gamatié, A. Hardware Solutions for Low-Power Smart Edge Computing. *J. Low Power Electron. Appl.* **2022**, *12*, 61. https://doi.org/10.3390/jlpea12040061

Academic Editor: Orazio Aiello

Received: 28 October 2022
Accepted: 21 November 2022
Published: 25 November 2022

1. Introduction

The edge computing paradigm [1] is an emerging paradigm for Internet-of-Things systems where computations are distributed across a variety of compact devices in order to bring computing capability closer to data sources, such as environmental sensors and cameras. We can mention the following advantages of edge computing over the traditional centralized computing paradigm found in cloud systems:

- reduced communication bandwidth and power costs as a result of reduced data transfers to centralized cloud servers;
- physical proximity of data and devices facilitates real-time data processing, such as for self-driving cars;
- in-situ processing at the edge devices ensures privacy regarding sensitive data, and prevents their offloading to remote locations;
- as the system is distributed, failure of some nodes can be easily overcome with a minimal impact on the global system and new devices can be added in a modular fashion to increase computing power.

Figure 1 illustrates the hierarchical layers of a typical edge computing infrastructure. Sensors in Layer 0 collect data from the environment first. In subsequent layers, the data are processed with appropriate devices based on the complexity of the processing. To meet the edge computing requirements, devices are placed close to sensors.

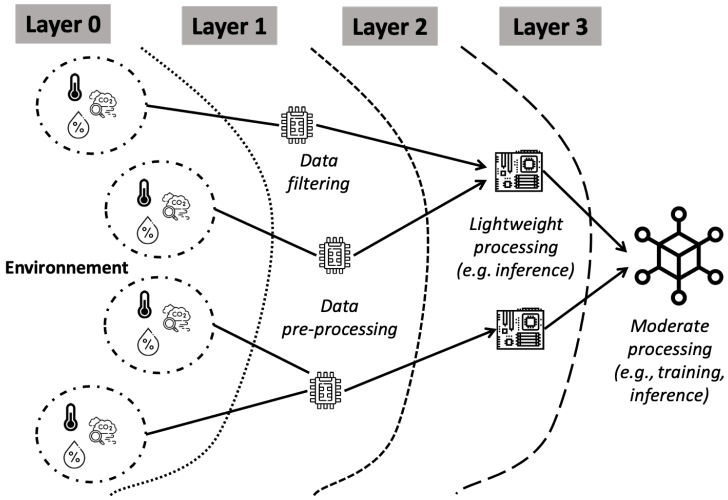


Figure 1. Hierarchical smart edge computing.

1.1. Machine Learning at the Edge

For edge computing nodes, implementing data analytics, particularly machine learning (ML), is a major challenge. Applications that leverage ML techniques at the edge are numerous. In essence, they deal with the inference problem. Real-time video analytics is a prominent application found in systems such as video surveillance, traffic control, and autonomous vehicles. Another notable application is feature extraction from images, for example, detecting areas and objects, identifying handwritten characters, and monitoring healthcare. To ensure people's safety (for instance, in the event of a fire) or to minimize energy consumption by utilizing renewable resources, smart homes and cities incorporate devices that use ML techniques for sensing and controlling the environment. Amazon's Alexa has made automatic speech recognition popular due to its success.

Based on the richness of ML techniques, three main families can be distinguished [2]. In *supervised* learning, classification and regression tasks are often achieved with algorithms such as support vector machines (SVM), artificial neural networks (ANNs), and linear regression. With the use of algorithms like k-means or x-means, *unsupervised* learning can be used for clustering and prediction tasks. *Reinforcement* learning focuses on decision-making through Q-learning. The majority of machine learning algorithms implemented on edge devices currently utilize *inference* (i.e., the process of directly solving ML problems with pre-trained ANNs) rather than *training* (i.e., the process of minimizing error as a function of ANN parameters, given an ML problem). One reason cited in [3] is the high bandwidth and latency costs involved in exchanging network updates across multiple edge devices (centralized model training might be more efficient since updated networks would be transferred directly to devices). Another concern in the design of edge nodes is energy and hardware costs. Structured labeled data is usually used for inference. The case is different for *deep learning* [4], which is used for training tasks in which precision is critical. In this process, complex multi-layer ANNs are applied along with huge amounts of raw data. Edge devices lack the computing power and data storage capacity needed for this.

1.2. Motivation and Contribution of This Study

Smart edge devices rely on embedded systems with limited resources to process sensor data. For ML workloads requiring high computing power, energy-efficient systems are necessary. In recent years, many research efforts have been focused on energy-efficient embedded system designs to solve ML problems [5–9]. These systems are primarily designed to make inferences. However, embedded systems, especially at edge nodes, are likely to require online learning, control, and optimization capabilities. In autonomous cars, for example, using remote cloud-based training could lead to long communication delays. Therefore, embedded training devices are preferred. However, after being trained offline,

ML inference models tend to diverge once in production. The retraining of such models will therefore require online training capabilities.

To understand how low-power embedded computing devices might help fill the aforementioned demand, this paper reviews the main design approaches for energy-efficient ML algorithm execution. In the following sections, it surveys candidates that meet both smart edge computing and Internet of Things requirements for low-power devices. CYSmart is a flexible and low-power smart edge computing system that we present as an example. A few working scenarios are used to evaluate the power efficiency of CYSmart.

1.3. Outline of the Paper

First, we discuss energy-efficient computing systems dedicated to executing machine learning algorithms in Section 2. Section 3 provides a classification of low-power devices for IoT and smart edge computing on the basis of hardware resources and power dissipation constraints. This classification is then used to present a panorama of popular devices. The CYSmart low-power edge computing system is described in Section 4. Moreover, it is briefly compared with selected industrial edge computing technologies. Finally, some concluding remarks are provided in Section 5.

2. A Quick Journey in the Landscape of Energy-Efficient Compute Systems for ML Tasks

Design principle-wise, embedded machine learning can be implemented through a central processing unit (CPU), graphics processing unit (GPU), application-specific integrated circuit (ASIC), and field-programmable gate array (FPGA). There is a wide range of outcomes in terms of power and accuracy of prediction for these implementations [10].

Figure 2 roughly illustrates this tradeoff. With the ASIC implementations [11,12], execution latency is optimized, but ML model accuracy is lower as the model is customized with approximations like quantization of ANN weights (i.e., reducing numerical precision by reducing the number of bits). Contrary to ASIC chip designs, CPUs and GPUs often support high-precision numerical representations, which improve prediction accuracy at the expense of power consumption. ASICs are more energy-efficient than CPUs and GPUs since they use less computing, I/O bandwidth, and memory resources. However, their development can be time-consuming and expensive. FPGAs offer a flexible and cost-effective implementation, allowing better balancing of power consumption, response latency, and prediction accuracy, as evidenced by recent studies [13,14]. Nevertheless, this comes at the expense of programmability, e.g., when compared to CPUs and GPUs.

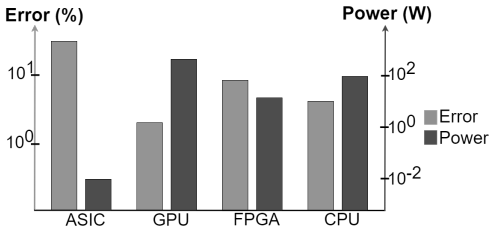


Figure 2. Power and prediction error for four hardware designs (based adapted from [10])—higher is worse.

In Figure 3, another perspective on existing machine learning systems is summarized [7]. Inference is addressed by systems with dissipation less than 100W. Among them are Google's Edge Tensor Processing Unit (TPUEdge) [15] and Intel's MovidiusX processors [16], embedded GPU-based neural engines such as the Apple A12 processor [17] and Huawei Kirin 980 [18], FPGA co-processors like the Zynq-020 [19] and the Stratix-V [20] chips, as well as mobile system-on-chips (SoCs) from Nvidia like Jetson-TX2 [21] and Xavier [22]. Training systems at high performance levels consume more than 100 watts. Typically, they consist of data center chips like the Google TPU3 [23] and the Intel Nervana2 [24] and data center systems like the Nvidia DGX-2 server system [25].

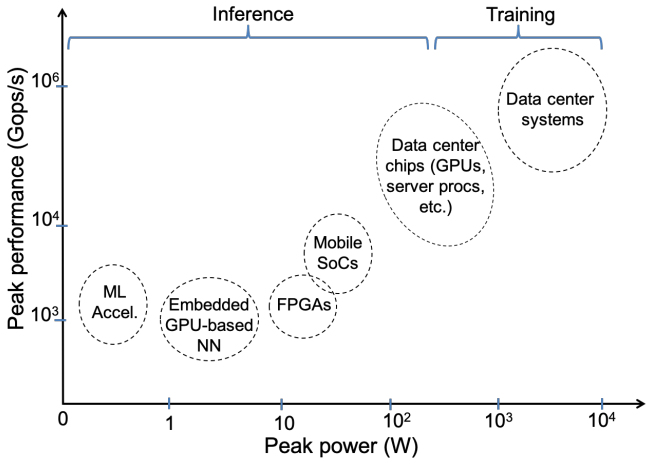


Figure 3. Computing landscape for ML: power vs. performance (adapted from [7]).

2.1. Focus on ML Accelerators, GPUs and FPGAs

Mobile phone SoCs, for example, use ML acceleration to address vector and matrix operations [26]. Various neural processing units and GPUs may be combined to achieve this, as in Qualcomm Snapdragon [27], HiSilicon 600 and 900 series chips [18] and MediaTek Helio P60 [28].

A typical approach toward efficient edge devices is to design hardware accelerators for machine learning models. This is already the case for ANNs for improved energy efficiency and throughput. By minimizing data access costs across the memory hierarchy, these accelerators can enable specialized processing dataflow that better exploits the memory characteristics. In [29], authors highlight several key design specializations tailored to machine learning accelerators: instruction sets that perform linear algebra operations like matrix multiplication and convolutions; on-chip buffers and on-board high-bandwidth memory to efficiently feed data; and high-speed interconnects that enable efficient communication between multiple cores. Additional hardware specializations for inference-only designs include Winograd convolution [30] and non-digital computing [12]. Although accelerators improve the execution performance of individual ML kernels, they may have some negative impact on the overall ML model performance because of costly communications between them and the associated system-on-chip (SoC).

Embedded GPUs and FPGAs are further alternatives for accelerating ML algorithms. As shown in Figure 4, several solutions exist. We only report devices with maximum power consumption of 50W, from the exhaustive list presented in [7]. The selected devices are compared w.r.t. their performance, power consumption and computational precision levels. Accelerators generally offer better precision and power consumption tradeoff, e.g., Google's tensor processing unit for edge computing (TPUEdge) [15] and Eyeriss [31]. On the other hand, GPUs and FPGAs globally provide better performance. The higher their computing precision, the higher their consumption, e.g., Xavier GPU [22] and ZCU102 FPGA [32].

For accelerating ML algorithms, embedded GPUs and FPGAs can also be used. Figure 4 shows several solutions. As part of the exhaustive list presented in [7], we report only devices with a maximum power consumption of 50W. We compare the selected devices in terms of performance, power consumption, and computational precision. There is generally a better tradeoff between precision and power consumption with accelerators, such as Google's tensor processing unit for edge computing (TPUEdge) [15] and Eyeriss [31]. In contrast, GPUs and FPGAs provide better performance globally. In general, the higher the precision of the computation, the more power it consumes, as in Xavier GPU [22] and ZCU102 FPGA [32].

A comprehensive survey on hardware accelerators has been proposed very recently in [33]. The reader can refer to this survey for a full coverage of the state of the art.

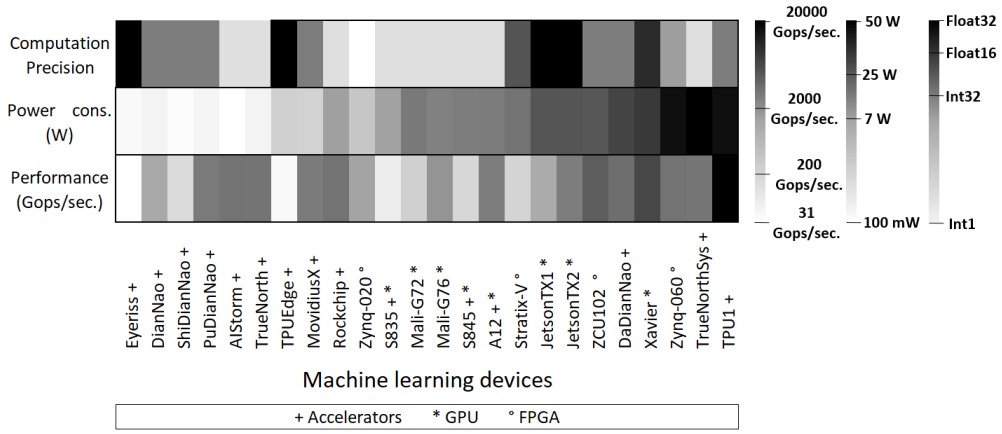


Figure 4. Accelerators, GPUs and FPGAs for embedded ML (adapted from [7])—he darker color, the higher the metric value.

2.2. From Software-Hardware Codesign to Emerging Computing Paradigms

Weight compression [34], parameter pruning, and weight quantization [35] are well-known ML optimization techniques for ANNs. Their goal is to improve energy efficiency by lowering computing complexity, data volume, and hardware resources used during the execution of the networks. Pruning involves gradually suppressing the connections between neurons in an ANN. Quantization involves reducing the number of bits in binary words. It is similar to approximate computing [36], where floating-point representations are converted to fixed-point representations. This reduces the precision of weight values in connections but speeds up execution. Another key aspect is developing compilers and runtime systems [37] that abstract away hardware details. This makes it easier to deploy and train ML models on mobile devices. The extensive software development environment made available to users by Nvidia contributes to the success of Nvidia GPUs for ML.

In the widely adopted von Neumann architectures, ML workloads based on ANNs frequently perform multiply-accumulate operations, which generate multiple data movements between memory and processors. As a result of these exchanges, there is a high execution time and power consumption, and this is known as the "memory wall". Modern ML commodity chips combine CPUs with High-Bandwidth Memory (HBM) via efficient interconnects to address this problem. In parallel, an emerging paradigm, called *near-data processing* [38], has been studied to address the memory wall issue. Computing capability is built into the memory or storage, enabling data stored there to be processed. Mixed-signal circuit design and advanced memory technologies were used to accomplish this. Other near-data processing techniques include in-memory processing [39] and in-storage processing [40]. Integrated 3D technologies and emerging Non-Volatile Memory (NVM) technologies enable such realizations. In comparison to DRAM, NVMs [41,42] like Spin Torque Transfer RAM (STT-RAM) and Resistive RAM (ReRAM) have lower leakage and higher cell density. By using them, edge nodes can mitigate their idle power draw concerns. In Hybrid Memory Cube (HMC) [43], several DRAM dies are stacked above the logic layer using Through-Silicon-Vias (TSV) to address the memory access issue.

3. Classification of Low-Power Devices for IoT and Smart Edge Computing

As IoT and edge computing grow in popularity, multiple sophisticated tiny embedded computing devices have emerged over the last decade. A general and systematic way of assisting designers in choosing low power IoT and smart edge computing devices does not exist. In a recent paper, Neto et al. [5] proposed a classification for IoT devices aimed at smart cities and smart buildings. We revised this classification to better reflect a broader class of edge computing devices encountered beyond smart cities and smart buildings. This includes hardware architectures used by mobile devices such as smartphones. The enhanced classification takes into account the hardware characteristics, including both

computing and memory components (which reflect the potential device performance), and the total power dissipated. The resulting classes are accompanied by some typical target algorithms that the corresponding device family can handle.

Our proposed extension of the classification from Neto et al. [5] is represented in Table 1. A total of six device classes are distinguished. The Class 0 devices are based on microcontrollers with limited memory capacity and power consumption. In general, the processed dataset is very small; for example, temperature and humidity measurements. Nevertheless, such devices can perform lightweight inference tasks using simple pre-trained models. Hence, all the subsequent device classes can be used for inference. A Class 1 device is one that can store data in addition to collecting and processing data. Such devices generally run on monocore microcontrollers or application cores with larger storage and memory capacities. Typically, such devices process only some basic statistics, such as noise reduction.

Table 1. Device classification for smart and low-power embedded systems (adapted with permission from [5]).

Class	Storage	Memory	Compute Unit Types	Power	Typical Algorithms
0	≤512 MB	≤512 kB	Microcontrollers	≤1 W	Basic computations (lightweight inference)
1	≤4 GB	≤512 MB	Microcontrollers/ Application cores	≤2 W	Basic statistics (inference)
2	≥4 GB	≤2 GB	Application cores	≤4 W	Classification/Regression (inference)
3	≥4 GB	≤8 GB	Application cores	≤16 W	Prediction/Decision-making (inference)
4	≥4 GB	≤16 GB	Application cores	≥16 W	Deep learning, auto-encoders, etc. (inference & training)
5	≥4 GB	≥16 GB	Application cores	≥16 W	

From Class 2, all devices are considered to have one or more application cores. The presence of SD card slots in the majority of these devices makes storage capacity more scalable. Devices of Class 2 are powerful enough to enable CNN inference, such as in image analysis. Their performance is good enough to execute lightweight IoT and edge workloads, as well as more intensive workloads such as training and inference.

In class 3, embedded GPUs make it possible to run lightweight training tasks. It is the first class with sufficient resources to enable real-time video analysis without any special ML accelerators.

In class 4 and class 5, we find devices that can be used in (quasi)autonomous systems such as smartphones or self-driving cars. The devices should be able to withstand environmental changes, while delivering the performance to process large datasets using high-performance accelerators, such as Nvidia GPUs found in server-class systems. A class 4 device is often intended for smartphones and is often more energy-efficient than a class 5 device, which is mostly designed for training and inference purposes.

Quick Survey of Typical Embedded Devices

Following the above classification, we now look at the most popular low-power devices used for IoT and edge computing recently. As shown in Tables 2–4, the devices identified fall into three application families:

- the first application family includes ultra-low-power devices with limited resources suitable for lightweight IoT and edge applications. This applies to all class 0 and class 1 devices;

- the second application family consists of the most popular devices encountered in average edge computing and IoT applications. All devices in class 2 and part of devices in class 3 are included in this class;
- the third application family includes devices with the most powerful hardware resources for performing machine learning and inference tasks. It covers a significant portion of devices in classes 3 and 4 and 5.

We categorize each application family by its device classes, its execution unit (CPU, GPU, accelerator, etc.), the most appropriate ML task (inference vs. training), and some domain application examples. Following the three application families outlined above, we will briefly discuss an application-driven panorama of key IoT and edge devices. We rely on [6] in part for this survey.

Table 2. Ultra low-power devices for IoT and edge computing.

Device	Class	GPU/Accel.	CPU	ML Usage	Application Examples
Arduino Mega	0	-	Microcontroller ATmega 8-bit @16 MHz	inference (ANN)	domotic [44], robotics [45]
Arduino RP2040	0	-	2xARM Cortex-M0+ @133 MHz (RP2040)	inference (ANN)	parking traffic [46], virtual reality [47]
MSP430G2553 LaunchPad	0	-	MSP430 16-Bit RISC Architecture @16 MHz	inference (ANN)	activity recognition [48]
Sony Spresense	0	-	6xARM Cortex-M4F @156 MHz	inference (ANN)	object detection [49]
SparkFun Edge	0	-	32-bit ARM Cortex-M4F @48 MHz	inference (ANN)	speech recognition [50]
STM32F103	0	-	ARM Cortex-M3 @72 MHz	inference (CNN)	image recognition [51]
STM32F765VI	0	-	ARM Cortex-M7 @216 MHz	inference (CNN)	image recognition [52]
Tiny Eats	0	-	ARM Cortex-M0+ @48 MHz	inference (DNN)	audio recognition [53]
Beaglebone Black	1	PowerVR SGX530 GPU	ARM Cortex-A8 single-core @1 GHz	inference (ANN)	robotics [54], camera drones [55]
Hello-Edge	1	-	ARM Cortex-M7 (STM32F746G)	inference (DNN)	keyword spotting [56]
MAX78000	1	Deep CNN Accelerator	ARM Cortex-M4 @100 MHz RISC-V coprocessor @60 MHz	inference (DNN)	object detection [57]
ZedBoard Dev. Board	1	FPGA accel.	2x ARM Cortex-A9 @667 MHz	inference (DNN,CNN)	image recognition [58]

The first application family, shown in Table 2, deals with lightweight data processing. Display devices are usually located close to data sources and are used primarily for domotics. The Arduino board is typically found in smart houses to monitor lightning [44]. Meanwhile, the most powerful devices may come with a compute accelerator integrated into them. Using its CNN accelerator, the MAX78000 device executes a light and optimized object detection algorithm on camera data [57]. Generally, the devices in Table 2 are affordable. They use programming models that are often closer to the hardware, i.e., at a low abstraction level, like assembly code.

Due to its inherent energy-efficiency, ARM technology is often adopted for embedded systems. Cortex-M microcontrollers and Cortex-A application processors are examples. In addition to ARM cores, a few designs use Intel technology, such as the Movidius Myriad 2 vision processing unit (VPU). Using them, deep neural networks (DNNs) can be run in smart cameras, for example. Also worth noting is the emerging GAP8 device, based on

the RISC-V open Instruction Set Architecture (ISA) [59]. It paves the way for a new era of processor innovation sustained by a collaborative and dynamic ecosystem.

There are a number of devices that combine ARM CPUs with embedded GPUs (see Tables 3 and 4), except for the Beaglebone Black system (see Table 2). In the powerful Jetson TX series and Tegra X1 systems, Nvidia's Pascal and Maxwell GPUs are combined with ARM Cortex-A57 cores. In both cases, the resulting systems can consume up to 15W of power. Since GPUs are present, they provide higher computing performance at the expense of more power consumption.

GPUs such as those reported in Table 3 are capable of executing the second application family. With Raspberry Pi, this involves image processing to detect tomato disease [60] or image super-resolution [61] or image classification [62] with smartphones. Other applications involve robotics, such as robotic perception [63] implemented using the Robot Operating System (ROS). As a result, the devices used here can support higher abstraction levels of programming.

According to Table 4, all devices in the third application family combine a GPU with at least four application cores, except for the RZ/V2M board. Applications primarily focus on image and video processing. With the Jetson Nano, real-time vehicle object detection is possible [64]. Furthermore, these devices are used in smartphones, such as the Huawei P40 Pro, which is equipped with powerful video super-resolution. Additionally, they can be used with ROS in the robotics field [65] for navigation, perception, and control.

Table 3. Low-power devices at the frontier of IoT and edge computing.

Device	Class	GPU/Accel.	CPU	ML Usage	Application Examples
BeagleBone AI	2	-	2x ARM Cortex-A15 @1,5 GHz 2x ARM Cortex-M4 SoC with 4 EVEs	inference (CNN)	computer vision [66]
Intel Movidius	2	-	Myriad-2 VPU	inference (SVM)	computer vision [67]
Raspberry Pi 3	2	400 MHz VideoCore IV GPU	4xARM A53 @1.2 GHz	inference (SVM, CNN)	video analysis [68] medical data processing [69]
Raspberry Pi Z2 W	2	400 MHz VideoCore IV GPU	4xARM Cortex-A53 @1 GHz	inference (CNN)	object detection [70]
RISC-V GAP8	2	-	8 RISC-V 32-bit @250 MHz + HW ConvolutionEngine	inference (CNN)	image, audio processing [71]
Samsung Galaxy S3 (Exynos 4412 SoC)	2	Mali-400 MP GPU	4xARM Cortex-A9 quad-core @1.4 GHz	inference (CNN)	image classif. [72]
Khadas VIM 3	2,3	4xARM Mali-G52 @800 MHz	4xARM Cortex-A73 @2.2 GHz 2xARM Cortex-A53 @1.8 GHz	inference (CNN)	robotics [63]
Raspberry Pi 4	2,3	500 MHz VideoCore VI GPU	4xARM Cortex-A72 @1.5 GHz	inference (SVM, CNN)	image analysis [60]
Motorola Z2 Force (Snapdragon 835 SoC)	2, 3	Qualcomm Adreno 540 GPU	4x Kryo 280 @ 2.45 GHz 4x Kryo 280 @ 1.9 GHz	inference (CNN)	image classif. [62] recognition [73]
Xiaomi Redmi 4X (Snapdragon 435 SoC)	2, 3	Qualcomm Adreno 505 GPU	8xARM Cortex-A53 @1.4 GHz	inference (CNN)	image super resolution [61]

Table 4. Powerful embedded devices for ML at the edge.

Device	Class	GPU / Accel.	CPU	ML Usage	Application Examples
Coral Development Board	3	GC7000 Lite GPU + TPUEdge accel.	NXP i.MX 8M SoC (4x ARM Cortex-A53 + Cortex-M4F)	inference (CNN)	image processing [74]
Google Pixel C (Tegra X1 SoC)	3	256-core Maxwell GPU	4x ARM Cortex-A57 + 4x ARM Cortex-A53	inference (SVM)	pedestrian recognition [75]
Jetson Nano	3	128-core Maxwell GPU	4x ARM Cortex-A57	inference (CNN)	video image recognition [64]
Jetson TX1	3	256-core Maxwell GPU	4x ARM Cortex-A57 2x MB L2	inference (CNN)	video, image analysis [76], robotics [77]
Odroid-XU4 (Exynos 5422 SoC)	3	ARM Mali-T628 MP6 GPU	4x ARM Cortex-A15 + 4x ARM Cortex-A7	inference/ training (ANN)	urban flooding, automobile traffic [78]
RZ/V2M Evaluation Board	3	DRP-AI	1x ARM Cortex-A53 @996 MHz	inference (CNN)	image processing [79]
Samsung Galaxy S8 (Exynos 8895 SoC)	3	ARM Mali-G71 GPU	4x ARM Cortex-A53 @ 1.7 GHz 4x Exynos M2 @2.5 GHz	inference (CNN)	image recognition [73]
Odroid-M1	3,4	4xARM Mali-G52 @ 650 MHz	4xARM Cortex-A55 @2 GHz	inference (CNN)	video image recognition [80]
Huawei P40 PRO (Kirin 990)	4	16xARM Mali-G76 @600 MHz	2x ARM Cortex-A76 @2.86 GHz 2x ARM Cortex-A76 @2.09 GHz 4x ARM Cortex-A55 @1.86 GHz	inference (CNN)	video super resolution [81]
Jetson TX2	4	256-core Pascal GPU	2x Denver2, 2 MB L2 + 4x ARM Cortex-A57, 2 MB L2	inference (CNN, DNN, SVM)	video, image analysis [76], robotics [82]
One Plus 9 Pro (Snapdragon 888)	4	Adreno 660 GPU	1x ARM Cortex-X1 @ 2.84 GHz 3x ARM Cortex-A78 @2.42 GHz 4x ARM Cortex-A55 @1.80 GHz	inference (CNN)	image classification [83]
Jetson AGX Orin	5	2048xCUDA cores 64xTensor cores @1.3 GHz	12xARM Cortex-A78 @2.2 GHz	inference (CNN)	robotics [65]
Jetson AGX Xavier	5	512xVolta GPU	8xNVIDIA Carmel	inference (CNN)	real-time object detection [84]

A few devices, however, combine CPUs with specific ML accelerators. ZedBoard and Coral Dev Board both integrate an FPGA-based accelerator and Google's TPUEdge [15]. Based on the computing complexity of algorithm execution, this diversity of cores enables maximizing the efficiency of the combined processing elements.

Homogeneous multicore devices appear mostly in devices for the first application family (Table 2), and in a few cases for the second application family (Table 3). These devices dissipate only a few milliwatts or a few Watts, such as SparkFun Edge and Hello-Edge. They, however, deliver less performance than heterogeneous devices.

Lastly, it is worth noting that most of the devices reported in Tables 2 and 3 are used for inference tasks rather than training (which is more expensive) due to their limited computing resources. Only some devices listed in Table 4 has been considered for lightweight training tasks, e.g., Odroid-XU4 board.

4. Low-Power Smart Edge Computing with CYSmart Solution

CYSmart is an edge computing system that gathers, processes, and displays locally measured data with minimal power consumption. It is capable of providing real-time feedback to domain experts. There are a number of low-power devices in this system, called

CYComs, which collect data from sensors at points of interest, pre-process it, and transmit it to the CYEdge via LoRa networks, as illustrated in Figure 5.

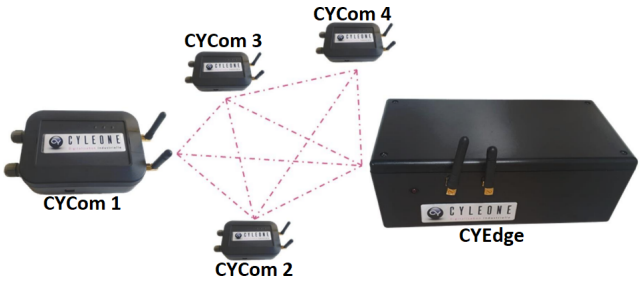


Figure 5. Overview of the CYSmart solution.

A CYCom implements the services provided by Layers 0 and 1 in Figure 1. As a result, CYSmart is able to perform some preliminary lightweight analyses on the collected data. This analysis can be performed to filter it before sending the result to the other components of the system. The outputs CYComs can then be processed and displayed by the CYEdge component, which typically implements Layers 2 and 3 of Figure 1. Data processing algorithms can determine which device class is appropriate for implementation of a CYEdge. For energy-efficient and secure computations, the latter is deployed close to CYComs.

- Measurement identifier and name of the point of interest
- Type of measurement performed
- Unit of measurement used
- Range of measurement desired
- Operating mode of the module (continuous measurement, on demand, sleep…)
- Time range of system activity
- Battery level of CYComs
- Limit ranges of expected values
- Alert generation
- Transmission signal strength

Every measure is stored in the CYEdge internal memory and remains accessible at any time through:

- a visualization tool that displays the measurement curves versus time;
- a download of all the information stored on a USB flash drive, computer, or server. Depending on the needs of the customer or the third party software used, the file type and format are adapted.

Figure 6 presents a more detailed technical presentation of CYSmart. The CYCom and CYEdge components are detailed in the next two sections, followed by some use case scenarios.

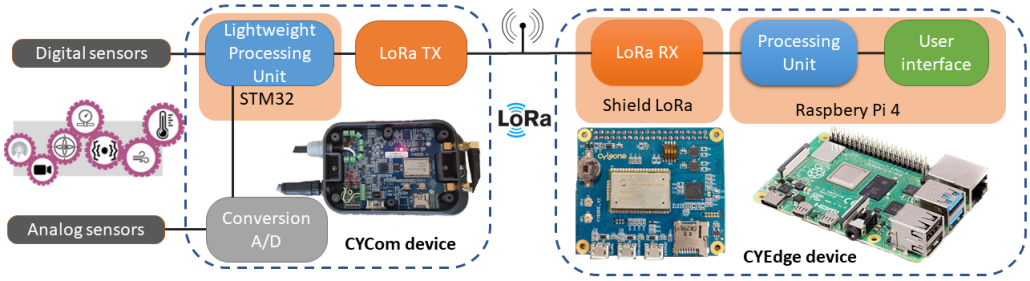


Figure 6. Detailed view of CYSmart solution.

4.1. Data Acquisition Device: CYCom

A CYCom is a device used to acquire data in the CYSmart system. Physical data can be recovered using this low-power technology running on an external battery. Data can come from digital sensors with serial communication (SPI, UART, I2C) or from analog sensors with a 16-bit Analog to Digital Converter (0–10V, 4–20 mA, or thermal resistance input). An STM32 microcontroller allows the CYCom to pre-process data (threshold detection, filtering, conversion…) before sending it by LoRa to the CYEdge. Each device can be physically configured to communicate with the CYEdge unit using one of two frequency bands (433 MHz or 866 MHz), each with four transmission channels. In the event the receiving device is not reachable, the sent frame can be stored on the receiving device's internal flash memory (8 MB) or SD card and sent back when the connection is restored. The device can also make use of other modules, such as a micro-USB port (currently used for CYCom updates), Bluetooth module, 3-axis inertial measurement unit (IMU) module (acceleration, angular, magnetic), or GPS module directly integrated within the device.

4.2. Centralized Early Data Processing: CYEdge

Data centralisation and setup of the sensor network is achieved with the CYEdge technology. It is a box that can be connected to an external battery or to the socket. The technology consists of two parts. The first one is a Raspberry Pi 4 (Processing Unit) and the second one is a proprietary shield developed by CYleone that enables LoRa communication with CYComs. This shield is a LoRa gateway that allows the processing of AT commands sent by the Raspberry Pi. These commands are sent to setup the CYComs but also to retrieve the data frames from the CYComs. The Raspberry Pi acts mainly as a data processing and graphic display unit. It reads, processes, saves and displays data from the shield on the graphical user interface. The interface allows the configuration of the sensor network and the retrieval of all the data and configurations of each CyCom. It can be accessed by connecting via WiFi or Ethernet to the local network created by the board itself, or to an existing network. The Raspberry can also be used for other parallel tasks such as GPS measurements, digital data retrieval and synchronization of this data with that received via LoRa.

4.3. Use Case Evaluation of CYSmart

One use case application of CYSmart is to collect data from different points of interest every interval of ten minutes, in a critical environment. As shown in Figure 6, the CYComs collect data from sensors where it is difficult to take and transmit measurements, such as aeraulic measurements in basements or bunkers. Data can be stored in the CYComs and sent to the CYEdge after lightweight processing, such as threshold detection or filtering. By sending only useful and ready-to-use data, this application minimizes LoRa communication. On the CYEdge, complementary data processing can be performed. A diagram of the CYCom's operations is shown in Figure 7.

A second use case application relies on LoRa to wirelessly transmit raw data from a CYCom directly to the CYEdge. The latter performs all data processing and displays the evolution of the values. This scenario is used to track the evolution of a process or a physical value in time, such as a refrigerator temperature in catering. Here, in most cases, the interval between two measures is around a few seconds, e.g., four seconds in our case. In addition, a CYEdge can only be paired with one to five CYComs. Figure 8 shows the corresponding flow diagram.

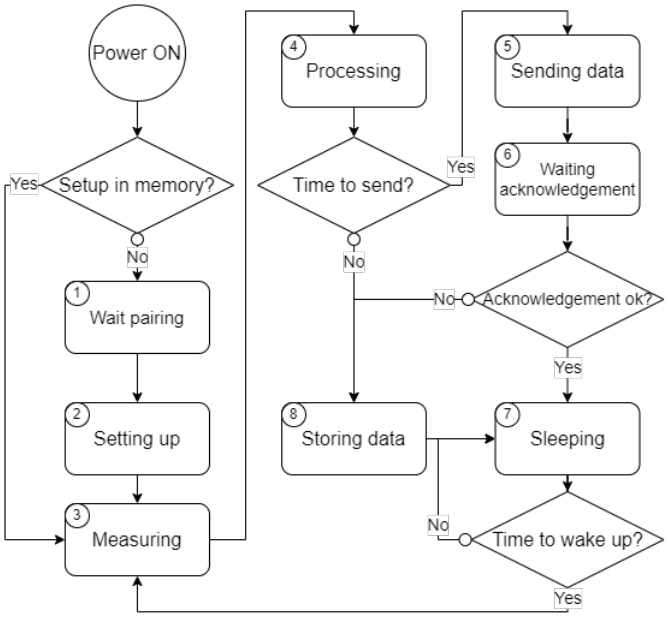


Figure 7. Operational diagram of the first application use case (Case I).

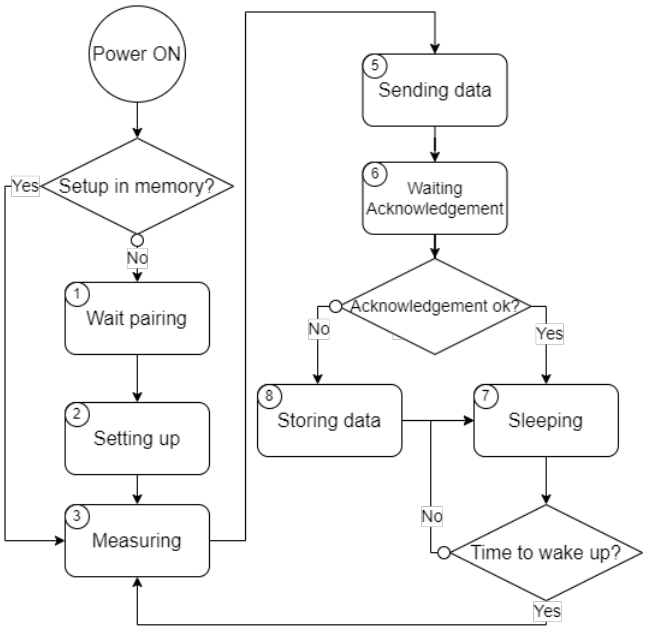


Figure 8. Operational diagram of the second application use case (Case II).

CYSmart devices are now ready to be evaluated according to the above use cases. As a processing unit, CYEdge utilizes a Raspberry PI 4 device with a LoRa shield. The CYCom uses a homogeneous architecture based on a STM32L496 microcontroller and ARM Cortex-M4 CPU (class 0 device hardware). The CYCom has a storage capacity of 1 MB from the STM32 and 8 MB from an external flash memory (class 0 device storage), as well as an SD card slot. There is also 320 KB memory in the STM32, corresponding to a class 0 memory.

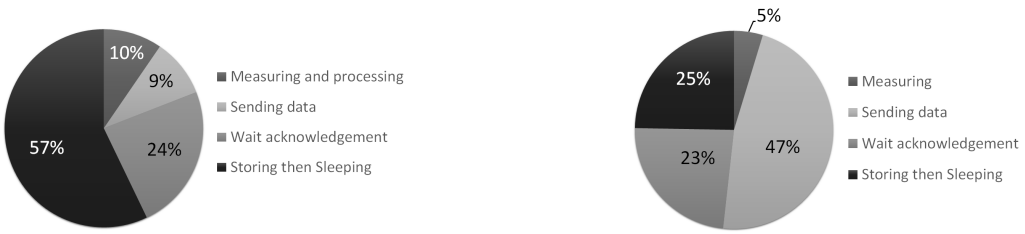
According to the different steps of the two operational diagrams, the evaluation of CYCom is presented in Table 5. During each step, the power is measured using the generator voltage (constant 15V) and the CYCom current draw. A CYCom is connected to a generator through an ampere-meter to measure the current draw with the greatest precision between the generator value and the ampere-meter display.

Table 5. Power consumption and duration of each steps in the use case scenarios.

Step Labels	Detailed of the Step	Power Consumption	Time Duration
1	Wait pairing and synchronizing from the CYEdge	412.5 mW	20 s
2	Setting up measurement parameters from the CYEdge	468 mW	7 s
3	Measuring digital and analog data from sensor	357 mW	10 s (Case I) 200 ms (Case II)
4	Data processing (filtering, conversion)	387 mW	50 ms
5	Sending stored and measured data to the CYEdge (LoRa)	377.5 mW	2–10 s
6	Waiting acknowledgement from the CYEdge	252 mW	1–25 s
7	Sleep until next measuring	177 mW	1 s–1 min
8	Storing not sent data in RAM memory	256.5 mW	50 ms

From Figures 7 and 8, we note two parts in the operational diagram. The first part, with steps 1 and 2, concerns the setup of a CYCom. Both steps are performed only once during the setup of the entire system and the CYComs. The latter retain their configuration in memory until they are reinitialized by the use of a hardware reset (push button inside). The second part relates to the execution routine of the device. In this routine, the device is woken up, a measurement is taken, data is processed and sent (depending on the use case), before returning to sleep until the next measurement. Steps 3 to 8 also belong to the routine and are executed in an infinite loop.

Figure 9 shows the duration distribution for the steps in one routine iteration, considering a worst-case scenario. In Case I (see Figure 9a), the worst case scenario occurs when there is 1 min between each data measure in the CYCom and there are frequent connection issues between the CYCom and CYEdge (i.e., low communication quality). In this worst-case scenario, (i) the CYCom transmits data to the CYEdge during 2 s and waits for an acknowledgement during 5 s; (ii) if no acknowledgment is received, the CYCom repeats phase (i) up to five times, otherwise it moves on to step 7. After five attempts (about 35 s), if no acknowledgement is received, the CYCom stores the data in its local memory, and proceeds to step 7.



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

Figure 9. Time distribution between the different steps in the worst-case scenario, for two use cases. (**a**) Use case I; (**b**) Use case II.

For Case II in Figure 9b, we assume that the CYEdge and CYCom are close to each other. This minimizes the loss of communication between both components during data exchanges. It enables the CYCom to send data to the CYEdge only once during 2 s and wait for the corresponding acknowledgement during 1 s. If no acknowledgement is received, the data is stored in the local memory of the CYCom. The worst-case scenario requires 3 s to reach step 7.

Despite their high power consumption, steps 1 and 2 only consist of system setup functions executed during installation. For this reason, as shown in Figures 10a,b, various components are activated during these steps to set them up.

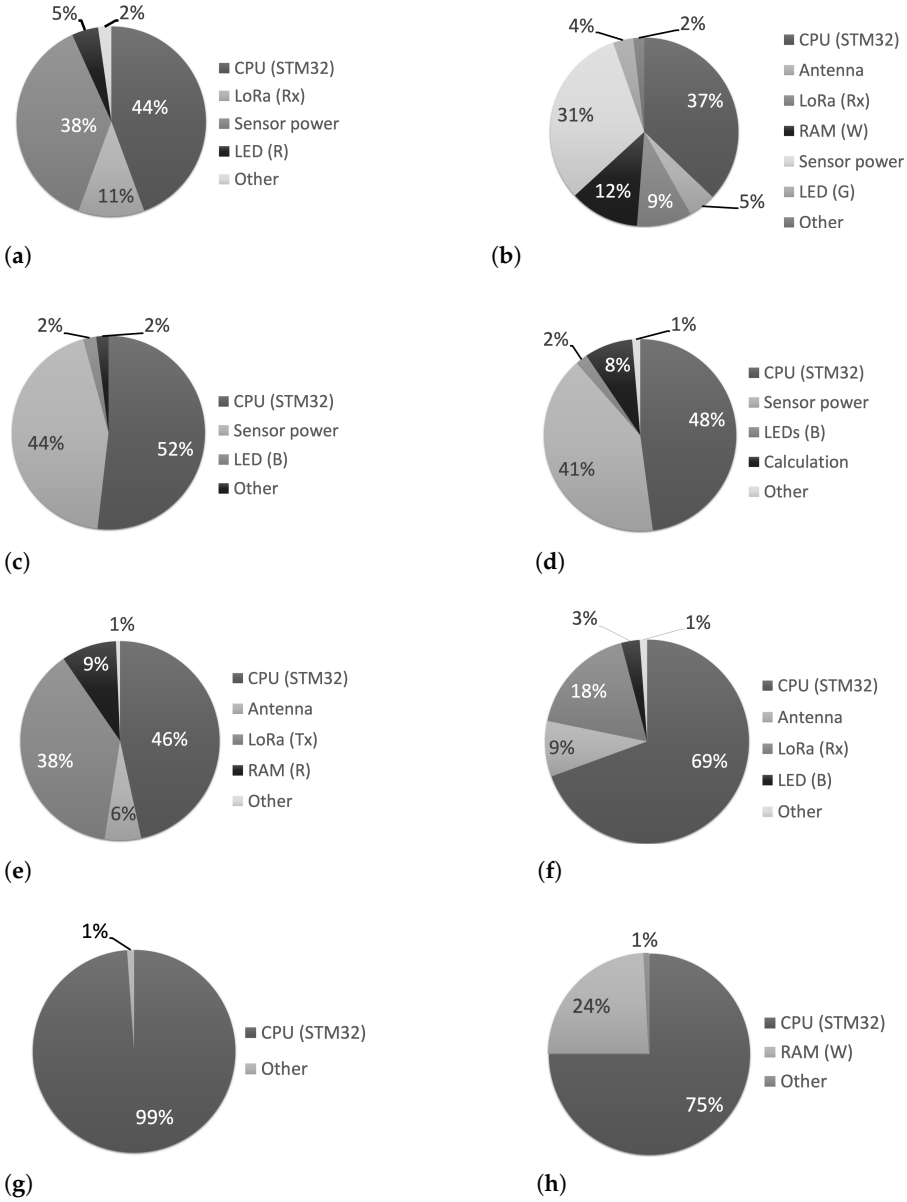


Figure 10. Power consumption breakdown for the different steps occurring in Figures 7 and 8. (**a**) Step 1; (**b**) Step 2; (**c**) Step 3; (**d**) Step 4; (**e**) Step 5; (**f**) Step 6; (**g**) Step 7; (**h**) Step 8.

During the execution of the routine, the sensor power management is activated only during steps 3 and 4. Figures 10c,d show that it is the second most power consuming function after the CPU. It is activated for 10 s, as in Case I. This represents 9% of the routine execution duration in Figure 9a and only 5% in Figure 9b. This allows for stable data collection from analog sensors. To provide the most precise data, it is necessary to have so much time. However, digital sensors can take less time to acquire data than analog sensors in Case II. This explains the shorter duration in Case II. The complexity of the data processing affects the processing time, but not the power consumption.

The CYCom sends data to CYEdge in steps 5 and 6. Here, the duration depends on the communication load between both devices. The CYCom will attempt to send data to the CYEdge five times before storing it (step 8). During steps 5 and 6, Figure 10e,f shows that

the LoRa communication is activated and represents up to 38% of the power consumption during the data transmission, i.e., the third most power consumer function.

Lastly, the process performs the sleep function, i.e., step 7. In Figure 10g, it is the step with the least activated functions: only the CPU is activated, resulting in the lowest power consumption. According to the use case, this function can be more or less time consuming. Figure 9a shows that in the worst-case scenario in Case I, CYCom is in this state 57% of the execution routine. Additionally, depending on the frequency of the data measurement, this step can be repeated and reach more than 95% of the process duration. In some cases, this duration can be shortened depending on the measuring frequency and the total number of CYComs deployed in the network to improve the bandwidth. With 1 s sleep duration in Case II, the routine spends 25% of its time in the worst-case scenario.

4.4. Gained Insights and Discussion

As a representative ultra-low-power device of the CYSmart system, the CYComs were the primary focus of the above use cases. There is also another component, the CYEdge, which embeds a Raspberry Pi 4 and a LoRa communication shield. Based on the power measurements of the CYEdge under normal operating conditions, it can be classified as a class 3 device, as described in Table 1. Below are some insights regarding CYSmart's current implementation and potential improvements.

Gained insights. The CYCom component of the CYSmart system utilizes a commercial-off-the-shelf (COTS) microcontroller manufactured by STMicroelectronics. Choosing this approach reduces the cost of the component as well as the development time. A CYCom's CPU is the primary energy consumer in the aforementioned use cases, followed by the LoRa module and the sensor power supply. Each of these three modules can be improved.

STM32 boards are based on von Neumann microarchitecture, leading to costly data movement between different hardware units. As a result, future improvements could include designing a customized solution that meets the requirements of the domain applications. This is consistent with the notion of domain-specific hardware accelerators as described in [85]. There is a lot of power consumed during one routine iteration in the second use case from the previous section without any data processing being performed. This unnecessary power consumption must be eliminated in order to improve the energy efficiency of the system. This problem may be solved by means of power gating, for example. A customized solution that incorporates such a mechanism is therefore desirable. Suitable design approaches should be considered for design space exploration by selecting high-level methodologies, e.g., [86–88], covering different abstraction levels: high-level analytical modeling [89–92], transaction-level modeling [93–95], cycle-accurate design [96–98], or register transfer level [99]. As surveyed in this paper, it is possible to implement the architecture using FPGA or ASIC designs at the expense of higher costly implementation efforts. As for CYComs, the CYEdge power consumption can be reduced by applying the same design methods.

Sensor power consumption is difficult to reduce since it is heavily dependent on the type of sensor being used. A wide range of digital and analog sensors can be interfaced with the CYCom. External 24 V lithium-ion batteries are currently used to power the integrated sensors. Instead of analog sensors, digital sensors with internal 3.3 V batteries could be considered here to reduce power consumption. Depending on the measurement environment, LoRa modules consume varying amounts of power. In both Cases I and II, the system can communicate across a reinforced concrete wall 90 cm thick with the initial parameters. In the case of a 15 dBm data transmission capacity and a spreading factor of 12, the maximum transmission delay is 2 s. Therefore, its maximum power consumption is 166 mW. These parameters can be adjusted according to the operational environment in order to reduce the LoRa module's power consumption.

Comparison of CYSmart w.r.t. selected industrial solutions. As a mature low-power edge computing solution, the CYSmart system can be compared with a number of industrial technologies. For this purpose, we consider some relevant criteria, described as follows:

- *Device classes*: supported device classes as defined in Table 1. This criterion implicitly suggests a range of power consumption;
- *Sensor diversity*: diversity of sensor types supported by a technology, such as digital versus analog sensors, as well as sensor voltage ranges. The criterion is qualitative in nature and can be rated on three levels: high, average, and low.
- *Transmission speed*: the speed of data transmission between the sensors at the edge frontier and the gateway or centralized system that is responsible for pre-processing the data. Generally, it is measured in terms of the number of samples per second (S/sec) or bits per second (b/sec);
- *Communication distance*: the distance over which a technology communicates wirelessly. It is essential in critical environments, such as basements, bunkers, and nuclear power plants;
- *Number of edge layers*: the number of layers considered in the hierarchical edge computing implementation, as shown in Figure 1;
- *Measurement points*: the number of data measurement points (i.e., sensors) managed by a single gateway or centralized system;
- *Dimension of measurement device*: the form factor of a device that incorporates sensors to collect data during the deployment of a technology;
- *Dimension of gateway/central device*: the form factor of a gateway or centralized system that manages sensor data;
- *Easy deployment*: the effort required for an easy deployment of a technology. This is a qualitative criterion;
- *Application diversity*: it refers to the variety of applications that can be leveraged by a technology, such as smart-home, smart-industry, and smart-city. The criterion is also qualitative in nature.

In light of the aforementioned criteria, Table 6 provides a comparison of CYSmart w.r.t. the industrial edge computing technologies summarized in the sequel. The *TMI-Orion* company proposes a solution for the design and manufacture of high level technologies that target harsh environments. A key component of its edge computing technology is a network of smart sensors such as NanoVACQ Fullradio [100], which communicate via a 2.4 GHz radio protocol with a Radio Transceiver [101]. Using a serial protocol, the latter transmits data to a host computer that manages and displays data from a sensor network. The *Gravio* company develops an IoT platform that is capable of connecting several sensors. Using the ZigBee wireless protocol, these sensors communicate with the Gravio Hub [102]. Data can be viewed and managed by users.

The *moneo appliance* is an edge solution manufactured by IFM company [103]. It consists of a dedicated software toolbox that allows for the management of sensor parameters as well as data display. Templates are provided in the toolbox for defining network configurations. Sensors are connected to the moneo appliance via an IO-Link Master, which serves as an interface between the appliance and the gateway computer.

The *Advantech* company developed an IoT solution that relies on data measurement devices named WISE (e.g., WISE-4060 [104]) and an intelligent edge server (e.g., EIS-D150 [105]). By using the WiFi protocol, the WISE devices send data from the sensors to the edge server. Users are provided with a real-time dashboard for managing WISE devices. The *InHand Networks* company has defined a specific gateway [106], which provides data optimization in the IoT infrastructure and provides real-time response times. The gateway device can be connected to a local network. It is compatible with real-time Ethernet protocols and supports the Docker software system.

The MCM200 series components (e.g., MCM-204 [107]) are edge computing solutions designed by the *Adlink* company. They are standalone data acquisition devices (i.e., no host

computer is required) that can monitor, analyze, and execute real-time actions. WiFi or Ethernet ports are available for communication.

Finally, the *Analog Devices* company offers the *SmartMesh Wireless HART* technology which consists of a small network manager (LTP5903-WHR [108]) that communicates with a number of sensor nodes called "motes" (e.g., LTP5900-WHR [109]). The network manager and motes must be programmed by the user. The network manager is responsible for centralizing data and communicating it to the host computer. Using analog data from sensors, the motes transmit data to the network manager.

Table 6 globally illustrates that CYSmart and Advantech technologies offer several advantages over other solutions. There are many similarities between these two technologies; however, CYSmart is capable of supporting a larger wireless communication distance than Advantech's solution. Because of this, CYSmart is well suited to critical environments, such as nuclear power plants.

Table 6. Comparison of CYSmart with similar edge computing technologies.

	CYSmart	**TMI Orion** [100,101]	**Gravio** [102]	**Moneo Appliance** [103]	**Advantech** [104,105]	**InHand Networks** [106]	**ADLINK** [107]	**Smartmesh WirelessHART** [108,109]
Device classes	0 and 3	0 and 2	2	4	0 and 3	2	2	0 and 2
Sensor diversity	high	low	low	average	high	low	high	average
Transmission speed	0.3 S/sec	10 S/sec	-	2500 S/sec	-	1000 Mb/sec	256 KS/sec	250 Kb/sec
Communication distance (in meters)	800	30	100	wired	110	wired	wired	200
Number of edge layers	2	2	2	2	3	2	2	2
Measurement points	20	4	64	16	-	6	20	500
Dimension of measurement device (in millimeters)	170 × 90 × 65	31 × 129 × 79	36 × 36 × 9	-	80 × 98 × 25	-	-	39 × 24 × 8
Dimension of gateway/central device (in millimeters)	245 × 110 × 85	127 × 8 × 46	97 × 97 × 29	35 × 105 × 150	260 × 140 × 54	180 × 115 × 45	110 × 40 × 126	103 × 56 × 20
Easy deployment	average	average	high	high	average	low	average	low
Application diversity	high	low	average	low	high	average	average	high

5. Summary

Embedded architectures for future edge devices likely will need to support training, control, and optimization capabilities, according to the current trends in edge computing. In this paper, we discuss recent efforts regarding energy-efficient hardware solutions for machine learning at the edge. We reviewed current design approaches and devices targeted at implementing IoT and smart edge computing with limited computing and power capabilities. Candidate low-power devices that could meet IoT and smart edge computing requirements have been surveyed. CYSmart, a flexible low-power edge computing system, was demonstrated as an interesting solution. To evaluate its power efficiency, a few working scenarios have been considered. Finally, a brief comparison of CYSmart with selected industrial edge computing technologies was presented.

Author Contributions: Conceptualization, L.M.W., J.-M.B., G.B. and A.G.; methodology, L.M.W., J.-M.B. and A.G.; software, L.M.W. and J.-M.B.; validation, L.M.W., J.-M.B. and A.G.; writing—original draft preparation, L.M.W., J.-M.B., G.B. and A.G.; writing—review and editing, L.M.W. and A.G.; supervision, J.-M.B., G.B. and A.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: The authors would like to thank Guillaume Devic and Gilles Sassatelli for their feedback in early discussions on part of the current work.

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

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
ANN	Artificial Neural Network
ASIC	Application-Specific Integrated Circuit
CNN	Convolution Neural Network
COTS	Commercial-Of-The-Shelf
CPU	Central Processing Unit
DRAM	Dynamic Random Access Memory
FPGA	Field-Programmable Gate Array
GPU	Graphics Processing Unit
HBM	High-Bandwidth Memory
HMC	Hybrid Memory Cube
I/O	Input/Output
IMU	Inertial Measurement Unit
IoT	Internet of Thing
ISA	Instruction Set Architecture
LoRa	Long Range
ML	Machine Learning
NVM	Non-Volatile Memory
RAM	Random Access Memory
ReRAM	Resistive RAM
ROS	Robot Operating System
SoC	System-on-Chip
STT-RAM	Spin Transfer Torque RAM
SVM	Support Vector Machines
TPU	Tensor Processing Unit
TSV	Through-Silicon-Vias

References

1. Satyanarayanan, M. The Emergence of Edge Computing. *Computer* **2017**, *50*, 30–39. [CrossRef]
2. Qiu, J.; Wu, Q.; Ding, G.; Xu, Y.; Feng, S. A survey of machine learning for big data processing. *EURASIP J. Adv. Signal Process.* **2016**, *2016*, 67. [CrossRef]
3. Kukreja, N.; Shilova, A.; Beaumont, O.; Huckelheim, J.; Ferrier, N.; Hovland, P.; Gorman, G. Training on the Edge: The why and the how. In Proceedings of the IEEE IPDPS Workshops,Rio de Janeiro, Brazil, 20–24 May 2019; pp. 899–903.
4. LeCun, Y.; Bengio, Y.; Hinton, G.E. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
5. Neto, A.R.; Soares, B.; Barbalho, F.; Santos, L.; Batista, T.; Delicato, F.C.; Pires, P.F. Classifying Smart IoT Devices for Running Machine Learning Algorithms. In Proceedings of the XLV Integrated SW and HW Seminar, Natal, Brazil, 14–19 July 2018.
6. Murshed, M.G.S.; Murphy, C.; Hou, D.; Khan, N.; Ananthanarayanan, G.; Hussain, F. Machine Learning at the Network Edge: A Survey. *ACM Comput. Surv.* **2022**, *54*, 1–37. [CrossRef]
7. Reuther, A.; Michaleas, P.; Jones, M.; Gadepally, V.; Samsi, S.; Kepner, J. Survey and Benchmarking of Machine Learning Accelerators. In Proceedings of the 2019 IEEE High Performance Extreme Computing Conference (HPEC), Waltham, MA, USA, 24–26 September 2019.
8. Andrade, L.; Prost-Boucle, A.; Pétrot, F. Overview of the state of the art in embedded machine learning. In Proceedings of the DATE Conference, Dresden, Germany, 19–23 March 2018; pp. 1033–1038.
9. Gamatié, A.; Devic, G.; Sassatelli, G.; Bernabovi, S.; Naudin, P.; Chapman, M. Towards Energy-Efficient Heterogeneous Multicore Architectures for Edge Computing. *IEEE Access* **2019**, *7*, 49474–49491. [CrossRef]
10. Deng, Y. Deep Learning on Mobile Devices—A Review. *Proc. SPIE* **2019**, 109930A. [CrossRef]
11. Chen, T.; Du, Z.; Sun, N.; Wang, J.; Wu, C.; Chen, Y.; Temam, O. DianNao: A Small-footprint High-throughput Accelerator for Ubiquitous Machine-learning. In Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '14), Salt Lake City, UT, USA, 1–5 March 2014; pp. 269–284. [CrossRef]
12. Shafiee, A.; Nag, A.; Muralimanohar, N.; Balasubramonian, R.; Strachan, J.P.; Hu, M.; Williams, R.S.; Srikumar, V. ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars. In Proceedings of the 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), Seoul, Republic of Korea, 18–22 June 2016; pp. 14–26. [CrossRef]
13. Nurvitadhi, E.; Venkatesh, G.; Sim, J.; Marr, D.; Huang, R.; Ong Gee Hock, J.; Liew, Y.T.; Srivatsan, K.; Moss, D.; Subhaschandra, S.; et al. Can FPGAs Beat GPUs in Accelerating Next-Generation Deep Neural Networks? In Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA'17), Monterey, CA, USA, 22–24 February 2017; pp. 5–14. [CrossRef]
14. Lacey, G.; Taylor, G.W.; Areibi, S. Deep Learning on FPGAs: Past, Present, and Future. *arXiv* **2016**, arXiv:1602.04283.
15. Google. Edge TPU. Available online: https://coral.ai/products (accessed on 27 October 2022).
16. Marantos, C.; Karavalakis, N.; Leon, V.; Tsoutsouras, V.; Pekmestzi, K.; Soudris, D. Efficient support vector machines implementation on Intel/Movidius Myriad 2. In Proceedings of the International Conference on Modern Circuits and Systems Technologies (MOCAST), Thessaloniki, Greece, 7–9 May 2018; pp. 1–4.
17. Peng, T. AI Chip Duel: Apple A12 Bionic vs Huawei Kirin 980. Available online: https://syncedreview.com/2018/09/13/ai-chip-duel-apple-a12-bionic-vs-huawei-kirin-980 (accessed on 27 October 2022).
18. HiSilicon. Kirin. 2019. Available online: https://www.hisilicon.com/en/SearchResult?keywords=Kirin (accessed on 27 October 2022).
19. Guo, K.; Sui, L.; Qiu, J.; Yu, J.; Wang, J.; Yao, S.; Han, S.; Wang, Y.; Yang, H. Angel-Eye: A Complete Design Flow for Mapping CNN Onto Embedded FPGA. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2018**, *37*, 35–47. [CrossRef]
20. Podili, A.; Zhang, C.; Prasanna, V. Fast and efficient implementation of Convolutional Neural Networks on FPGA. In Proceedings of the IEEE 28th International Conference on Application-specific Systems, Architectures and Processors (ASAP), Seattle, WA, USA, 10–12 July 2017; pp. 11–18. [CrossRef]
21. NVIDIA. Jetson TX2. 2019. Available online: https://www.nvidia.com/fr-fr/autonomous-machines/embedded-systems/jetson-tx2 (accessed on 27 October 2022).
22. Hruska, J. Nvidia's Jetson Xavier Stuffs Volta Performance Into Tiny Form Factor. 2018. Available online: https://www.extremetech.com/computing/270681-nvidias-jetson-xavier-stuffs-volta-performance-into-tiny-form-factor (accessed on 27 October 2022).
23. Teich, P. Tearing Apart Google's TPU 3.0 AI Coprocessor. 2018. Available online: https://www.nextplatform.com/2018/05/10/tearing-apart-googles-tpu-3-0-ai-coprocessor/ (accessed on 27 October 2022).
24. Rao, N. Beyond the CPU or GPU: Why Enterprise-Scale Artificial Intelligence Requires a More Holistic Approach. 2018. Available online: https://newsroom.intel.com/editorials/artificial-intelligence-requires-holistic-approach/ (accessed on 27 October 2022).
25. Cutress, I. NVIDIA's DGX-2: Sixteen Tesla V100s, 30TB of NVMe, Only $400K. 2018. Available online: https://www.anandtech.com/show/12587/nvidias-dgx2-sixteen-v100-gpus-30-tb-of-nvme-only-400k (accessed on 27 October 2022).
26. Ignatov, A.; Timofte, R.; Chou, W.; Wang, K.; Wu, M.; Hartley, T.; Gool, L.V. AI Benchmark: Running Deep Neural Networks on Android Smartphones. In Proceedings of the European Conference on Computer Vision (ECCV) Workshops, Munich, Germany, 8–14 September 2018.

27. Qualcomm. Neural Processing SDK for AI. 2019. Available online: https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk (accessed on 27 October 2022).
28. MediaTek. Helio P60. 2019. Available online: https://www.mediatek.com/products/smartphones/mediatek-helio-p60 (accessed on 27 October 2022).
29. Ananthanarayanan, R.; Brandt, P.; Joshi, M.; Sathiamoorthy, M. Opportunities and Challenges Of Machine Learning Accelerators In Production. In Proceedings of the USENIX Conference on Operational Machine Learning, Santa Clara, CA, USA, 20 May 2019; pp. 1–3.
30. Lavin, A.; Gray, S. Fast Algorithms for Convolutional Neural Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016. [CrossRef]
31. Chen, Y.H.; Yang, T.J.; Emer, J.; Sze, V. Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2018**, *9*, 292–308. [CrossRef]
32. Xilinx. Tearing Apart Google's TPU 3.0 AI Coprocessor. 2019. Available online: https://www.xilinx.com/products/boards-and-kits/ek-u1-zcu102-g.html (accessed on 27 October 2022)
33. Peccerillo, B.; Mannino, M.; Mondelli, A.; Bartolini, S. A survey on hardware accelerators: Taxonomy, trends, challenges, and perspectives. *J. Syst. Archit.* **2022**, *129*, 102561. [CrossRef]
34. Han, S.; Mao, H.; Dally, W.J. Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding. In Proceedings of the 4th International Conference on Learning Representations, ICLR, San Juan, Puerto Rico, 2–4 May 2016.
35. Marculescu, D.; Stamoulis, D.; Cai, E. Hardware-aware Machine Learning: Modeling and Optimization. In Proceedings of the International Conference on Computer-Aided Design (ICCAD '18), San Diego, CA, USA, 5–8 November 2018; pp. 137:1–137:8.
36. Gupta, S.; Agrawal, A.; Gopalakrishnan, K.; Narayanan, P. Deep Learning with Limited Numerical Precision. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015. [CrossRef]
37. C4ML organizers. Compilers for ML. 2019. Available online: https://www.c4ml.org/ (accessed on 27 October 2022).
38. Balasubramonian, R.; Chang, J.; Manning, T.; Moreno, J.H.; Murphy, R.; Nair, R.; Swanson, S. Near-Data Processing: Insights from a MICRO-46 Workshop. *IEEE Micro* **2014**, *34*, 36–42. [CrossRef]
39. Liu, J.; Zhao, H.; Ogleari, M.A.; Li, D.; Zhao, J. Processing-in-Memory for Energy-Efficient Neural Network Training: A Heterogeneous Approach. In Proceedings of the IEEE/ACM MICRO Symposium, Fukuoka, Japan, 20–24 October 2018; pp. 655–668.
40. Choe, H.; Lee, S.; Park, S.; Kim, S.J.; Chung, E.; Yoon, S. Near-Data Processing for Machine Learning. 2017. Available online: https://openreview.net/pdf?id=H1_EDpogx (accessed on 27 October 2022).
41. Endoh, T.; Koike, H.; Ikeda, S.; Hanyu, T.; Ohno, H. An Overview of Nonvolatile Emerging Memories—Spintronics for Working Memories. *IEEE JETCAS* **2016**, *6*, 109–119. [CrossRef]
42. Senni, S.; Torres, L.; Sassatelli, G.; Gamatié, A.; Mussard, B. Exploring MRAM Technologies for Energy Efficient Systems-On-Chip. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2016**, *6*, 279–292. [CrossRef]
43. Pawlowski, J.T. Hybrid memory cube (HMC). In Proceedings of the IEEE Hot Chips Symposium (HCS), Stanford, CA, USA, 17–19 August 2011; pp. 1–24.
44. Kusriyanto, M.; Putra, B.D. Smart home using local area network (LAN) based arduino mega 2560. In Proceedings of the 2nd International Conference on Wireless and Telematics (ICWT), Yogyakarta, Indonesia, 1–2 August 2016; pp. 127–131.
45. Drgoňa, J.; Picard, D.; Kvasnica, M.; Helsen, L. Approximate model predictive building control via machine learning. *Appl. Energy* **2018**, *218*, 199–216. [CrossRef]
46. Sousa, R.d.S. Remote Monitoring and Control of a Reservation-Based Public Parking. Ph.D. Thesis, Universidade de Coimbra, Coimbra, Portugal, 2021.
47. Brun, D.; Jordan, P.; Hakkila, J. Demonstrating a Memory Orb—Cylindrical Device Inspired by Science Fiction. In Proceedings of the 20th International Conference on Mobile and Ubiquitous Multimedia, Leuven, Belgium, 5–8 December 2021; pp. 239–241.
48. Stolovas, I.; Suárez, S.; Pereyra, D.; De Izaguirre, F.; Cabrera, V. Human activity recognition using machine learning techniques in a low-resource embedded system. In Proceedings of the 2021 IEEE URUCON, Montevideo, Uruguay, 24–26 November 2021; pp. 263–267.
49. Edge Impulse. Detect objects with centroids (Sony's Spresense). Available online: https://docs.edgeimpulse.com/docs/tutorials/detect-objects-using-fomo (accessed on 27 October 2022).
50. SparkFun Electronics. Edge Hookup Guide. 2019. Available online: https://learn.sparkfun.com/tutorials/sparkfun-edge-hookup-guide/all (accessed on 27 October 2022).
51. Jin, G.; Bai, K.; Zhang, Y.; He, H. A Smart Water Metering System Based on Image Recognition and Narrowband Internet of Things. *Rev. D'Intelligence Artif.* **2019**, *33*, 293–298. [CrossRef]
52. Alasdair Allan. Deep Learning at the Edge on an Arm Cortex-Powered Camera Board. 2018. Available online: https://aallan.medium.com/deep-learning-at-the-edge-on-an-arm-cortex-powered-camera-board-3ca16eb60ef7 (accessed on 27 October 2022).
53. Nyamukuru, M.T.; Odame, K.M. Tiny eats: Eating detection on a microcontroller. In Proceedings of the 2020 IEEE Second Workshop on Machine Learning on Edge in Sensor Systems (SenSys-ML), Sydney, Australia, 21 April 2020; pp. 19–23.

54. Sharad, S.; Sivakumar, P.B.; Narayanan, V.A. The smart bus for a smart city—A real-time implementation. In Proceedings of the IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), Bangalore, India, 6–9 November 2016; pp. 1–6.
55. Nayyar, A.; Puri, V. A Review of Beaglebone Smart Board's-A Linux/Android Powered Low Cost Development Platform Based on ARM Technology. In Proceedings of the 9th International Conference on Future Generation Communication and Networking (FGCN), Jeju Island, South Korea, 25–28 November 2015; pp. 55–63.
56. Zhang, Y.; Suda, N.; Lai, L.; Chandra, V. Hello Edge: Keyword Spotting on Microcontrollers. *arXiv* **2017**, arXiv:1711.07128.
57. Wang, G.; Bhat, Z.P.; Jiang, Z.; Chen, Y.W.; Zha, D.; Reyes, A.C.; Niktash, A.; Ulkar, G.; Okman, E.; Hu, X. BED: A Real-Time Object Detection System for Edge Devices. *arXiv* **2022**, arXiv:2202.07503.
58. Wang, C.; Yu, Q.; Gong, L.; Li, X.; Xie, Y.; Zhou, X. DLAU: A Scalable Deep Learning Accelerator Unit on FPGA. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2016**, *36*, 513–517. [CrossRef]
59. RISC-V Foundation. RISC-V: The Free and Open RISC ISA. 2019. Available online: https://riscv.org/ (accessed on 27 October 2022).
60. Gonzalez-Huitron, V.; León-Borges, J.A.; Rodriguez-Mata, A.; Amabilis-Sosa, L.E.; Ramírez-Pereda, B.; Rodriguez, H. Disease detection in tomato leaves via CNN with lightweight architectures implemented in Raspberry Pi 4. *Comput. Electron. Agric.* **2021**, *181*, 105951. [CrossRef]
61. Ledig, C.; Theis, L.; Huszar, F.; Caballero, J.; Cunningham, A.; Acosta, A.; Aitken, A.; Tejani, A.; Totz, J.; Wang, Z.; et al. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. In Proceedings of the The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
62. Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; Kalenichenko, D. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In Proceedings of the The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018.
63. Rodríguez-Gómez, J.P.; Tapia, R.; Paneque, J.L.; Grau, P.; Eguíluz, A.G.; Martínez-de Dios, J.R.; Ollero, A. The GRIFFIN perception dataset: Bridging the gap between flapping-wing flight and robotic perception. *IEEE Robot. Autom. Lett.* **2021**, *6*, 1066–1073. [CrossRef]
64. Valladares, S.; Toscano, M.; Tufiño, R.; Morillo, P.; Vallejo-Huanga, D. Performance Evaluation of the Nvidia Jetson Nano Through a Real-Time Machine Learning Application. In *Proceedings of the Intelligent Human Systems Integration 2021*; Russo, D., Ahram, T., Karwowski, W., Di Bucchianico, G., Taiar, R., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 343–349.
65. Chemel, T.; Duncan, J.; Fisher, S.; Jain, R.; Morgan, R.; Nikiforova, K.; Reich, M.; Schaub, S.; Scherlis, T. Tartan Autonomous Underwater Vehicle Design and Implementation of TAUV-22: Kingfisher. 2020. Available online: https://robonation.org/app/uploads/sites/5/2022/06/RS2022_Carnegie_Mellon_University_TartanAUV_TDR.pdf (accessed on 27 October 2022).
66. Long, C. BeagleBone AI Makes a Sneak Preview. 2019. Available online: https://beagleboard.org/blog/2019-05-16-beaglebone-ai-preview (accessed on 27 October 2022).
67. Hochstetler, J.; Padidela, R.; Chen, Q.; Yang, Q.; Fu, S. Embedded Deep Learning for Vehicular Edge Computing. In Proceedings of the IEEE/ACM Symposium on Edge Computing (SEC), Bellevue, WA, USA, 25–27 October 2018; pp. 341–343.
68. Xu, R.; Nikouei, S.Y.; Chen, Y.; Polunchenko, A.; Song, S.; Deng, C.; Faughnan, T. Real-Time Human Objects Tracking for Smart Surveillance at the Edge. In Proceedings of the International Conference on Communications (ICC), Kansas City, MO, USA, 20–24 May 2018; pp. 1–6.
69. Triwiyanto, T.; Caesarendra, W.; Purnomo, M.H.; Sułowicz, M.; Wisana, I.D.G.H.; Titisari, D.; Lamidi, L.; Rismayani, R. Embedded Machine Learning Using a Multi-Thread Algorithm on a Raspberry Pi Platform to Improve Prosthetic Hand Performance. *Micromachines* **2022**, *13*, 191. [CrossRef] [PubMed]
70. Willems, L. Detect People on a Device that Fits in the Palm of Your Hands. Bachelor's Thesis, University of Twente, Enschede, The Netherlands, 2020.
71. Flamand, E.; Rossi, D.; Conti, F.; Loi, I.; Pullini, A.; Rotenberg, F.; Benini, L. GAP-8: A RISC-V SoC for AI at the Edge of the IoT. In Proceedings of the International Conference on Application-Specific Systems, Architectures and Processors (ASAP), Milan, Italy, 10–12 July 2018; pp. 1–4.
72. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861.
73. Kang, D.; Kang, D.; Kang, J.; Yoo, S.; Ha, S. Joint optimization of speed, accuracy, and energy for embedded image recognition systems. In Proceedings of the 2018 Design, Automation Test in Europe Conference Exhibition (DATE), Dresden, Germany, 19–23 March 2018; pp. 715–720. [CrossRef]
74. Cass, S. Taking AI to the edge: Google's TPU now comes in a maker-friendly package. *IEEE Spectr.* **2019**, *56*, 16–17. [CrossRef]
75. Campmany, V.; Silva, S.; Espinosa, A.; Moure, J.; Vázquez, D.; López, A. GPU-based Pedestrian Detection for Autonomous Driving. *Procedia Comput. Sci.* **2016**, *80*, 2377–2381. [CrossRef]
76. Liu, Q.; Huang, S.; Han, T. Fast and Accurate Object Analysis at the Edge for Mobile Augmented Reality: Demo. In Proceedings of the 2nd ACM/IEEE Symposium on Edge Computing, SEC'17, San Jose/Fremont, CA, USA, 12–14 October 2017; pp. 33:1–33:2.
77. Ezra Tsur, E.; Madar, E.; Danan, N. Code Generation of Graph-Based Vision Processing for Multiple CUDA Cores SoC Jetson TX. In Proceedings of the International Symposium on Embedded Multicore/Many-core SoC (MCSoC), Hanoi, Vietnam, 12–14 September 2018; pp. 1–7.

78. Beckman, P.; Sankaran, R.; Catlett, C.; Ferrier, N.; Jacob, R.; Papka, M. Waggle: An open sensor platform for edge computing. In Proceedings of the 2016 IEEE SENSORS, Orlando, FL, USA, 30 October–3 November 2016; pp. 1–3.
79. Morishita, F.; Kato, N.; Okubo, S.; Toi, T.; Hiraki, M.; Otani, S.; Abe, H.; Shinohara, Y.; Kondo, H. A CMOS Image Sensor and an AI Accelerator for Realizing Edge-Computing-Based Surveillance Camera Systems. In Proceedings of the 2021 Symposium on VLSI Circuits, Kyoto, Japan, 13–19 June 2021; pp. 1–2. [CrossRef]
80. Hardkernel. Odroid-M1. 2022. Available online: https://www.hardkernel.com/2022/03/ (accessed on 27 October 2022).¶
81. Liu, S.; Zheng, C.; Lu, K.; Gao, S.; Wang, N.; Wang, B.; Zhang, D.; Zhang, X.; Xu, T. Evsrnet: Efficient video super-resolution with neural architecture search. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 19–25 June 2021; pp. 2480–2485.
82. Chinchali, S.; Sharma, A.; Harrison, J.; Elhafsi, A.; Kang, D.; Pergament, E.; Cidon, E.; Katti, S.; Pavone, M. Network Offloading Policies for Cloud Robotics: A Learning-based Approach. *Auton. Robot.* **2021**, *45*, 997–1012. [CrossRef]
83. Pouget, A.; Ramesh, S.; Giang, M.; Chandrapalan, R.; Tanner, T.; Prussing, M.; Timofte, R.; Ignatov, A. Fast and accurate camera scene detection on smartphones. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Virtual, 19–25 June 2021; pp. 2569–2580.
84. Dextre, M.; Rosas, O.; Lazo, J.; Gutiérrez, J.C. Gun Detection in Real-Time, using YOLOv5 on Jetson AGX Xavier. In Proceedings of the 2021 XLVII Latin American Computing Conference (CLEI), Cartago, Costa Rica, 25–29 October 2021; pp. 1–7. [CrossRef]
85. Dally, W.J.; Turakhia, Y.; Han, S. Domain-Specific Hardware Accelerators. *Commun. ACM* **2020**, *63*, 48–57. [CrossRef]
86. Apvrille, L.; Bécoulet, A. Prototyping an Embedded Automotive System from its UML/SysML Models. In Proceedings of the Embedded Real Time Software and Systems (ERTS2012), Toulouse, France, 29 January–1 February 2012.
87. Dekeyser, J.L.; Gamatié, A.; Etien, A.; Ben Atitallah, R.; Boulet, P. Using the UML Profile for MARTE to MPSoC Co-Design. Available online: https://www.researchgate.net/profile/Pierre-Boulet/publication/47363143_Using_the_UML_Profile_for_MARTE_to_MPSoC_Co-Design/links/09e415083fb08c939b000000/Using-the-UML-Profile-for-MARTE-to-MPSoC-Co-Design.pdf (accessed on 27 October 2022).
88. Yu, H.; Gamatié, A.; Rutten, É.; Dekeyser, J. Safe design of high-performance embedded systems in an MDE framework. *Innov. Syst. Softw. Eng.* **2008**, *4*, 215–222. [CrossRef]
89. Parashar, A.; Raina, P.; Shao, Y.S.; Chen, Y.H.; Ying, V.A.; Mukkara, A.; Venkatesan, R.; Khailany, B.; Keckler, S.W.; Emer, J. Timeloop: A Systematic Approach to DNN Accelerator Evaluation. In Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Madison, WI, USA, 24–26 March 2019; pp. 304–315. [CrossRef]
90. An, X.; Boumedien, S.; Gamatié, A.; Rutten, E. CLASSY: A Clock Analysis System for Rapid Prototyping of Embedded Applications on MPSoCs. In *Proceedings of the 15th International Workshop on Software and Compilers for Embedded Systems, SCOPES'12*; Association for Computing Machinery: New York, NY, USA, 2012; pp. 3–12. [CrossRef]
91. Caliri, G.V. Introduction to analytical modeling. In Proceedings of the 26th International Computer Measurement Group Conference, Orlando, FL, USA, 10–15 December 2000; pp. 31–36.
92. Corvino, R.; Gamatié, A.; Geilen, M.; Józwiak, L. Design space exploration in application-specific hardware synthesis for multiple communicating nested loops. In Proceedings of the 2012 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation, SAMOS XII, Samos, Greece, 16–19 July 2012; pp. 128–135. [CrossRef]
93. Ghenassia, F. *Transaction-Level Modeling with SystemC: TLM Concepts and Applications for Embedded Systems*; Springer: New York, NY, USA, 2006.
94. Mello, A.; Maia, I.; Greiner, A.; Pecheux, F. Parallel simulation of systemC TLM 2.0 compliant MPSoC on SMP workstations. In Proceedings of the Design, Automation Test in Europe Conference Exhibition (DATE'10), Dresden, Germany, 8–12 March 2010; pp. 606–609. [CrossRef]
95. Schirner, G.; Dömer, R. Quantitative Analysis of the Speed/Accuracy Trade-off in Transaction Level Modeling. *ACM Trans. Embed. Comput. Syst.* **2009**, *8*, 1–29. [CrossRef]
96. Binkert, N.; Beckmann, B.; Black, G.; Reinhardt, S.K.; Saidi, A.; Basu, A.; Hestness, J.; Hower, D.R.; Krishna, T.; Sardashti, S.; et al. The Gem5 Simulator. *SIGARCH Comput. Archit. News* **2011**, *39*, 1–7. [CrossRef]
97. Butko, A.; Gamatié, A.; Sassatelli, G.; Torres, L.; Robert, M. Design Exploration for next Generation High-Performance Manycore On-chip Systems: Application to big.LITTLE Architectures. In Proceedings of the ISVLSI: International Symposium on Very Large Scale Integration; Montpellier, France, 8–10 July 2015; pp. 551–556. [CrossRef]
98. Butko, A.; Garibotti, R.; Ost, L.; Lapotre, V.; Gamatié, A.; Sassatelli, G.; Adeniyi-Jones, C. A trace-driven approach for fast and accurate simulation of manycore architectures. In Proceedings of the 20th Asia and South Pacific Design Automation Conference, Chiba, Japan, 19–22 January 2015; pp. 707–712. [CrossRef]
99. Breuer, M.; Friedman, A.; Iosupovicz, A. A Survey of the State of the Art of Design Automation. *Computer* **1981**, *14*, 58–75. [CrossRef]
100. TMI Orion nano Vacq FUll Radio. Available online: https://www.tmi-orion.com/medias/pdf/en/NanoVACQ-PT-FullRadio-EN.pdf (accessed on 27 October 2022).
101. TMI Orion Transceiver. Available online: https://www.tmi-orion.com/medias/pdf/en/Radio-transceiver-en.pdf (accessed on 27 October 2022).
102. Gravio Hub. Available online: https://doc.gravio.com/manuals/gravio4/1/en/topic/gravio-hub (accessed on 27 October 2022).

103. Moneo Appliance. Available online: https://www.ifm.com/us/en/us/moneo-us/moneo-appliance (accessed on 27 October 2022).
104. Advantech WISE-4060. Available online: https://advdownload.advantech.com/productfile/PIS/WISE-4060/file/WISE-4060-B_DS(122121)20221020155553.pdf (accessed on 27 October 2022).
105. Advantech EIS-D150. Available online: https://advdownload.advantech.com/productfile/PIS/EIS-D150/file/EIS-D150_DS(050922)20220509111551.pdf (accessed on 27 October 2022).
106. inHand Networks Edge Gateway. Available online: https://inhandnetworks.com/upload/attachment/202210/19/InHand%20Networks_InGateway902%20Edge%20Gateway_Prdt%20Spec_V4.1.pdf (accessed on 27 October 2022).
107. Adlink MCM Edge DAQ. Available online: https://www.adlinktech.com/Products/Download.ashx?type=MDownload&isDatasheet=yes&file=1938%5cMCM-210_Series_datasheet_20210412.pdf (accessed on 27 October 2022).
108. SmartMesh WirelessHART Network Manager. Available online: https://www.analog.com/media/en/technical-documentation/data-sheets/5903whrf.pdf (accessed on 27 October 2022).
109. SmartMesh WirelessHART 5900. Available online: https://www.analog.com/media/en/technical-documentation/data-sheets/5900whmfa.pdf (accessed on 27 October 2022).