Sustainable Computing: Informatics and Systems 50 (2026) 101353

Contents lists available at ScienceDirect

Sustainable Computing: Informatics and Systems

journal homepage: www.elsevier.com/locate/suscom

Deep learning-driven sensing for sustainable Internet of Things

Lorenzo Calisti, Chiara Contoli, Emanuele Lattanzi *

Department of Pure and Applied Sciences, University of Urbino, Piazza della Repubblica 13, Urbino, 61029, Italy

ARTICLE INFO

Dataset link: https://github.com/IoTUniurb/deep-learning-based-data-collection/tree/main/datasets

Keywords:
Tiny machine learning
Deep-learning
Prediction-based data collection
Sustainable computing

ABSTRACT

The rapid proliferation of the Internet of Things has transformed various domains of everyday life, from familiar household items like smart light bulbs to healthcare resources, including medical devices, wearable technologies, smart devices, and ultimately, smart cities. However, the increasing use of these devices has raised concerns about their energy consumption, which poses a significant challenge to their sustainability. This study presents a novel, deep-learning-based sensing framework for energy-efficient data collection. Using predictive models to forecast sensor readings, the framework adapts to environmental changes and optimizes the time spent sensing and transmitting data. This leads to significant energy savings while preserving data quality. Extensive simulations, combined with experiments conducted on a case study based on three popular IoT hardware platforms, demonstrate the approach's effectiveness, achieving up to 89% energy savings in the active states, compared to traditional monitoring methods, while maintaining high data accuracy and system performance. This research contributes to the sustainable growth of IoT ecosystems, promoting more energy-efficient and environmentally responsible systems for the future.

1. Introduction

The Internet of Things (IoT) is reshaping modern society by embedding intelligence into everyday objects and environments. From household appliances and wearable technologies to industrial machinery and smart urban infrastructure, billions of interconnected devices are now continuously sensing, processing, and transmitting data in real time [1,2]. Their rapid expansion is revolutionizing healthcare, transportation, agriculture, and energy management sectors, fostering unprecedented levels of automation and data-driven decision-making [3–5].

As of 2024, it is estimated that there are over 16 billion active IoT devices in operation worldwide, with projections reaching over 30 billion by 2030 [6]. Recent studies suggest that, in the near future, IoT devices will consume tens of terawatt-hours (TWh) of energy annually, with projections reaching approximately 46 TWh per year by 2025. This figure is significant, roughly equivalent to the annual electricity consumption of a small country [7]. Forecasts based on more recent models estimate that energy consumption by IoT devices could reach approximately 112–131 TWh/year by 2025, and if trends continue, IoT could account for as much as 25% of global energy consumption by 2030, depending on how efficiently the industry implements low-power design and renewable energy integration. [8]. Despite the quantitative differences between the various predictive models, it is evident that energy consumption related to IoT devices is set to increase in the coming years. It is therefore essential to design hardware and software solutions that reduce energy consumption in order to ensure greater sustainability for the IoT ecosystem.

The IoT network generally has a distributed architecture. Numerous edge devices communicate with each other and with intermediate nodes, called fog nodes, to send data to a few central cloud servers. This configuration enables efficient information management, reducing latency and enhancing system responsiveness. However, the need for continuous data collection creates a high energy demand for edge devices, which can be problematic, especially when resources are limited [9,10]. These devices consume energy for sensing, data processing, and, especially, wireless communication, the latter of which is often the largest contributor to their energy footprint [11,12].

Traditional approaches to mitigating IoT energy consumption focus on several aspects of device operation. Techniques such as duty cycling (periodically turning off radios), adaptive sampling rates, and low-power communication protocols have yielded promising results [13,14]. However, these methods often have limitations. They may rely on static configurations, lack adaptability to dynamic environmental changes, compromise data accuracy or timeliness to save energy, or fail to optimize the combined energy costs of sensing and transmission. There is still an urgent need for intelligent, adaptive frameworks that can significantly reduce energy expenditure without compromising essential system performance or data fidelity.

Recent advances in Deep Learning (DL) present a range of promising solutions to this challenge. DL models, particularly Recurrent Neural

* Corresponding author.
E-mail address: emanuele.lattanzi@uniurb.it (E. Lattanzi).

https://doi.org/10.1016/j.suscom.2026.101353
Received 10 October 2025; Received in revised form 26 February 2026; Accepted 2 April 2026
Available online 4 April 2026

Networks (RNNs) like Long Short-Term Memory networks (LSTMs) and Gated Recurrent Units (GRUs), and Temporal Convolutional Networks (TCNs), have demonstrated remarkable capabilities in learning complex temporal patterns and forecasting sensor data streams with high accuracy [15–17]. This predictive capability presents a transformative opportunity: by forecasting near-future sensor readings, devices could potentially reduce the frequency of physical sensing operations, activating sensors only when necessary to correct significant deviations from predictions or to gather data crucial for model retraining.

Integrating predictive models with IoT devices enables the anticipation of future environmental conditions and the dynamic adjustment of data collection strategies. Among such methods, Prediction-Based Data Collection (PBDC) has gained prominence for reducing data transmissions without compromising the original sampling frequency [18, 19].

In PBDC, an identical forecasting model is deployed on the edge device and the cloud. Upon acquiring a new sample, the system evaluates whether the observed value falls within a predefined tolerance range of the predicted value. If the prediction is accurate within this threshold, the sample is not transmitted; otherwise, it is sent to the server for processing. This approach effectively conserves energy and optimizes bandwidth by limiting data transmission to instances where significant deviations occur.

This paper addresses the critical challenge of energy sustainability in IoT systems by introducing a novel deep learning-based predictive sensing framework for end-to-end energy optimization. Our solution dynamically adapts the data sensing and transmission activities of IoT devices according to DL forecasts of sensor data and real-time environmental dynamics. In particular, we present two complementary strategies, namely Deep Learning-Based Data Collection (DLBDC) and Deep Learning-Driven Sensing (DLDS). DLBDC reduces the number of transmitted packets, while DLDS extends the duration of deep sleep states by lowering the frequency with which nodes wake up.

We provide formal models for both strategies and develop a simulation environment to evaluate their behavior under diverse operating conditions. We also thoroughly characterize their implementation on three real, constrained devices. Experimental results, conducted on a case study of three popular IoT hardware platforms, demonstrate that our framework yields substantial energy savings, up to 36% with DLBDC and up to 89% with DLDS. These results surpass the performance of state-of-the-art baselines while maintaining data accuracy within controlled bounds.

This work makes a significant contribution toward sustainable IoT ecosystems, advancing the development of more energy-efficient and environmentally responsible ubiquitous computing systems.

The remainder of this paper is organized as follows: Section 2 reviews existing literature on energy optimization in IoT through data collection strategies. Section 3 presents the problem formulation and introduces the proposed deep learning-based framework for adaptive sensing and data acquisition. Section 4 describes the experimental setup, detailing the environmental datasets, the simulation environment, the accuracy metrics adopted for evaluation, the state-of-the-art baseline methods used for comparison, and the case study conducted on real devices. Section 5 reports and discusses the experimental findings. Finally, Section 6 outlines the current limitations of the framework and discusses directions for future developments, while Section 7 concludes the paper.

2. Related work

Energy consumption in the IoT ecosystem is a well-known concern that necessitates the development of energy-efficient solutions to ensure sustainability. According to the literature, particularly that which pertains to wireless sensor networks, data transmission is among the most energy-intensive tasks [20]. This has prompted the research community to explore strategies aimed at reducing the volume and frequency of data transmission.

Over the last two decades, time series forecasting has received increasing attention as an approach to reducing communication in monitoring systems. For instance, in 2006, Tulone and Madden used an autoregressive model to predict local sensor readings, thereby improving data collection performance and reducing communication among sensors in the network [21]. Gedik et al. proposed an adaptive sampling approach in which a subset of nodes collects data directly, while another subset of nodes is dynamically selected as non-samplers and predicts the values using probabilistic models that are locally and periodically constructed [22]. In 2023, Giordano et al. proposed an energy-aware adaptive sampling rate algorithm designed for deployment in battery-powered IoT devices. Their algorithm, based on a Finite State Machine (FSM), maximizes sensor sampling rates without risking battery depletion [23]. Similarly, Ben-Aboud et al. implemented two adaptive sampling techniques: a lightweight algorithm that computes the distance between two consecutive sample values, and an optimized uniform sampling method that tries to maximizes the sampling interval while maintaining an acceptable data quality [24]. Law et al. proposed another approach that leverages the concept of adaptive sampling to reduce power drain [25]. In their work, the authors designed an algorithm to skips sampling when data loss is estimated to be low (i.e., when the forecasts might not deviate significantly from the readings without actually acquiring them). Skipping some samples has also been used to minimize the energy consumption of a wearable sensor systems for diagnostic purposes [26]. In particular, the authors collected electrocardiography, electromyography, acoustic cardiography, and acoustic myography data without compromising the accuracy of the physiological measurements. Other authors focused on IoT tasks offloading instead. Rahmani et al. developed an A3C-AHP offloading framework solution to optimize task offloading for multiple IoT users in blockchain-enabled environments [27]. Moghaddasi et al. propose an algorithm that uses an advanced Deep Reinforcement neural network to optimize task offloading in Device-to-Device (D2D), Device-to-Edge (D2E), and Device-to-Cloud (D2C) communications [28]. They use Deep Learning to discern patterns, evaluate potential offloading decisions, and adapt in real time to dynamic environments.

The approaches mentioned above are also known as model-driven data acquisition [29] and have in common the fact that nodes can avoid communicating all sensed data as long as the prediction does not exceed a certain threshold [30]. A slightly different approach was used in [19], where Płaczek proposed a prediction algorithm that determines future sensor reading value intervals based on observed increases and decreases rates in historical data. Płaczek's approach considers the length of the sleep mode, i.e., the amount of time that the sensor node remains dormant before transmitting the next sensor reading. Han et al. also utilized the concept of the interval in their work, as described in [31]. In this study, the authors built a simulator consisting of a server, a database, and a number of sensors to implement their adaptive data collection mechanism. Other approaches combine data compression-based mechanisms, such as Principal Component Analysis (PCA), with autoregressive prediction mechanisms, such as Autoregressive Integrated Moving Average (ARIMA), to reduce the transmission overhead [32,33]. Besides autoregressive mechanisms, one can leverage deep learning models. For instance, Putra et al. employed a deep learning multilayer perceptron to predict future sensor data and optimize the energy efficiency of industrial IoT networks [34]. Table 1 summarizes the referenced works, highlighting their main contributions and shortcomings and noting whether they were evaluated on real embedded hardware.

Our methodology is inspired by the work in [30]. However, unlike the existing literature, we propose two novel approaches: Deep Learning-Based Data Collection (DLBDC) and Deep Learning-Driven Sensing (DLDS). The former is a novel approach that employs a deep learning forecasting model to predict future sensor values, thereby

Table 1
An overview of relevant studies.

Author(s)	Key contributions	Shortcomings	Hardware
Płaczek [19]	Reduces sampling and transmissions by dynamically selecting a target node based on sensor readings prediction.	Sensitive to communication delays.	Yes
Tulone et al. [21]	Uses autoregressive models built at each sensor to predict local readings. Nodes transmit these local models to a sink node. Low computational cost and memory needs.	Requires geographic clustering.	No
Gedik et al. [22]	Proposes adaptive sampling via dynamic samplers and probabilistic models.	Centralized prediction overhead.	No
Giordano et al. [23]	Designs Finite State Machine-based adaptive sampling to improve sustainability.	Reaches 0.85x optimal performance.	Yes
Ben-Aboud et al. [24]	Adjusts sampling intervals via normalized sample distance. Lightweight and accurate.	Depends on historical data patterns.	Yes
Law et al. [25]	Reduces sampling up to 49% using adaptive analysis.	Upper limit capped at 50%.	No
Lalouani et al. [26]	Proposes a replicated LSTM on the sensor and gateway to cut communication cost.	Requires reliable synchronization.	Yes
Rahmani et al. [27]	Reduces energy/latency via task offloading in blockchain MEC-IoT.	High computation overhead.	No
Moghaddasi et al. [28]	Applies Rainbow Deep Q-Network for task offloading in D2D–Edge–Cloud.	Needs high-quality training data.	No
Pötsch et al. [29]	Uses Derivative-Based Prediction (DBP) to suppress up to 99% transmissions of temperature data.	Prediction performance is sensitive to data patterns.	Yes
Raza et al. [30]	Uses Derivative-Based Prediction (DBP) to suppress up to 99.7% transmissions of raw data.	Loss of a model introduces large errors.	Yes
Han et al. [31]	Minimizes power consumption in distributed sensing via statistical models.	Requires gateway-side computation.	No
Hussein et al. [32]	Uses compression-based distributed prediction for IoT.	Assumes linear data relationships.	No
Pandey et al. [33]	Minimizes unnecessary data transmissions in IoT WSN.	Requires complete datasets.	No
Putra et al. [34]	Uses DL-based prediction for industrial IoT efficiency.	Computationally complex.	No

further reducing data transmission. The latter investigates an additional dimension of power optimization by extending the device's deep sleep period, reducing the frequency of node wake-ups, without impairing the system's monitoring capabilities. Most of the works mentioned above focus on prediction accuracy and simulate the environment, either neglecting or evaluating theoretically, the energy consumption. In this study, we propose two formal models of the aforementioned strategies and assess their implementation on three real constrained popular devices: the Espressif ESP32 [35], the NXP MCXN647 [36], and the Raspberry Pi Pico 2 W [37]. In particular, the objective is to measure the real energy and power consumption of each device, in addition to the Mean Absolute Percentage Error (MAPE) across different datasets. Our research aims to show the feasibility of these techniques and improve the efficiency of IoT devices to enable their sustainable proliferation, thus paving the way for more energy-efficient and environmentally responsible ubiquitous computing systems.

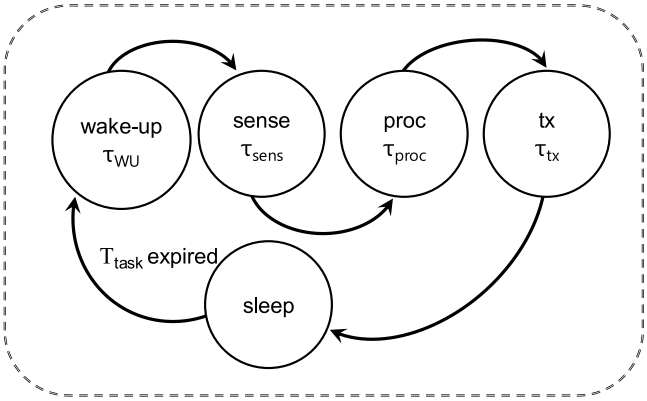


Fig. 1. State diagram of a traditional IoT monitoring task.

3. Proposed approach

This section presents the proposed deep-learning-driven data collection and sensing strategies by providing the problem formulation and a deep analysis of their theoretical energy performance.

3.1. Problem formulation

The IoT network adopts a distributed architectural paradigm, in which many edge devices communicate bidirectionally with one another and intermediary nodes, commonly referred to as fog nodes, before ultimately transmitting data to centralized cloud servers.

In general, an IoT task that runs on an edge device and involves monitoring can be modeled as a finite state machine that considers different operational states, as shown in Fig. 1.

In particular, for a task characterized by a period T_{task} (i.e., the time interval between successive activations of the task), we can figure out a chain of five different states. When the timer expires, the IoT node wakes from a sleep state with low power consumption. It goes

through a transient state (*wake-up*), accounting for the reactivation of the microcontroller unit and of the input/output peripherals. When ready, it starts a new measure using the internal sensors (*sense*) and, possibly, processes the data (*proc*) which are, finally, transmitted to the cloud (*tx*) before returning to *sleep*. Each state is characterized by an energy expenditure and time that strictly depend on the edge hardware platform. In general, we can summarize the overall energy consumption (E_{task}) as composed of the following contributions:

$$\begin{aligned} E_{task} &= \mathcal{E}_{WU} + \mathcal{E}_{sens} + \mathcal{E}_{proc} + \mathcal{E}_{tx} \\ &+ (T_{task} - \tau_{WU} - \tau_{sens} - \tau_{proc} - \tau_{tx}) \cdot \mathcal{P}_{sleep} \end{aligned} \quad (1)$$

Where $\mathcal{E}_{WU}$, $\mathcal{E}_{sens}$, $\mathcal{E}_{proc}$, and $\mathcal{E}_{tx}$ are, respectively, the energy accounted for node wake-up, data sensing, processing, and transmission (hereafter referred as *Active Energy*). In turn, the energy consumed in the sleep state is due to the power consumption in the sleep state $\mathcal{P}_{sleep}$ multiplied by the task period T_{task} subtracted of the τ_{WU}, τ_{sens}, τ_{proc}, and τ_{tx} which are, respectively, the time spent during wake-up, data sensing, processing, and transmission.

For most microcontrollers used in IoT applications, power consumption during active operation is substantially higher than in low-power sleep modes. These modes leverage quiescent states, such as deep sleep or hibernation, to minimize overall energy usage. Among active states, wireless data transmission is generally the most energy-intensive task due to the activation of transceiver components. In some instances, these components can account for most of the device's energy consumption. Therefore, reducing data transmission significantly saves energy and benefits the application's sustainability. Additionally, power optimization can be achieved by significantly increasing the time spent in low-power states at the expense of active ones.

3.2. Prediction-based data collection strategies

Typically, an IoT task involves the continuous monitoring of parameters such as environmental conditions, and it can tolerate a limited degree of inaccuracy in the reported data. Unlike scenarios where the cloud server expects exact measurements in every report, the correctness of these applications is not compromised if the reported values closely approximate the actual values. Deviations from exact data are permissible if the magnitude of the error remains within acceptable bounds [18]. The main challenge is achieving an optimal balance between data accuracy and energy efficiency, which is essential for effectively using IoT technologies in real-time monitoring applications.

Deep Learning-Based Data Collection (DLBDC) and Deep Learning-Driven Sensing (DLDS) are two methodologies aimed at reducing power consumption in IoT devices. We proposed them, inspired by the Prediction-Based Data Collection (PBDC) methodology presented by Raza et al. in 2012 [30].

PBDC is a widely adopted strategy that adjusts the number of transmitted data samples dynamically without altering the original sampling frequency. In this approach, the same predictive model is deployed on both the edge device and the cloud server simultaneously. When a new data sample is acquired, the IoT node evaluates its deviation from the model's predicted value based on a predefined tolerance threshold. If the deviation is within the acceptable range, transmission of the data is suppressed. Otherwise, the sample is forwarded to the server. By transmitting only samples that significantly diverge from the model's forecast, PBDC utilizes bandwidth more efficiently, reduces energy consumption, and minimizes superfluous data transmissions while preserving data fidelity. Furthermore, since transmission is triggered by data diverging from the forecast, the method is robust against unpredictable trends, sudden changes, and anomalies.

The proposed DLBDC is a novel approach that uses a deep learning forecasting model to predict future sensor values, reducing the need for frequent data transmissions even further. In contrast, DLDS explores an additional dimension of power optimization by prolonging deep sleep periods by reducing the frequency of node wake-ups without compromising the system's monitoring effectiveness.

3.3. Deep learning-based data collection

The proposed DLBDC technique leverages a Deep Learning-based forecasting model running simultaneously on the edge devices and the cloud server. This approach enables the system to predict expected sensor values and selectively transmit only the necessary data, optimizing communication efficiency without compromising the overall reliability of the collected information.

Fig. 2 refers to the finite-state machine representing the DLBDC task. To better illustrate the technique, we consider the case of a single edge node running a task characterized by a sampling period T_{task} on top of which a forecasting deep-learning model runs to predict the measured value. Let V_t be the value measured by the device at time t and $\hat{V}_t$ the corresponding value predicted at time t using a buffer of length ws containing data sampled a times $[t_{-ws}, t_{-\ldots}, t_{-2}, t_{-1}]$. For a tolerance ϵ accepted by the cloud server, a prediction is considered "approved" if its value falls within the range:

$$\hat{V}_t \in [V_t - \epsilon, V_t + \epsilon] \quad (2)$$

In practice, if $|\hat{V}_t - V_t| < \epsilon$, the estimated value at time t is considered a good approximation of the actual value. In this case, no transmission is needed, and the cloud server can, in turn, predict the value using the same deep-learning model, substituting $\hat{V}_t$ for V_t. Employing a shared forecasting model and maintaining an identical copy of the values in the history buffer on both the cloud and the edge device makes this transmission saving possible. This enables the cloud server to predict values in the same manner as the edge node. Conversely, sending the measured data is necessary only when the value is outside the tolerance range (i.e., $|\hat{V}_t - V_t| >= \epsilon$).

The assumption that makes this approach convenient is that, in most cases, if the monitored quantity does not change too rapidly or shows a well-defined trend, the data will be predicted with sufficient precision to fall within the acceptable range.

Notice that, for simplicity, we will not consider all possible problems that can arise with an unstable internet connection, such as transmission errors and delays. We will also assume that the transmission medium does not have any of these problems, as they are outside the scope of this work. In general, in the DLBDC technique, a transmission failure would be misinterpreted by the cloud server as a non-transmission of a predictable sample (i.e., a value falling inside the tolerance of the forecasting model) while it actually is not. This would introduce slight inaccuracies that may appear in the measurements until the edge and the cloud buffers are synchronized again.

Concerning the forecasting models, it is important to underline that they are trained once and for all on top of a dedicated machine and then embedded into the device firmware without needing periodic over-the-air transmission.

Starting from Eq. (1), we derive the energy model for the DLBDC approach as follows:

$$\begin{aligned} E_{task(DLBDC)} =& \mathcal{E}_{WU} + \mathcal{E}_{sens} + \mathcal{E}_{proc} + \mathcal{E}_{pred} + \mathcal{E}_{tx} \cdot (1 - SR_{tx}) + \\ & [T_{task} - \tau_{WU} - \tau_{sens} - \tau_{proc} - \tau_{pred} - \tau_{tx} \cdot (1 - SR_{tx})] \cdot \mathcal{P}_{sleep} \end{aligned} \quad (3)$$

where SR_{tx} represents the suppression ratio in the transmission achieved by the forecasting model for a given value of tolerance ϵ, which practically depends on the model's strength in predicting data. From the energetic point of view, the contribution of the transmission $\mathcal{E}_{tx}$ should be reduced by a factor represented by SR_{tx} while the cost of the prediction phases $\mathcal{E}_{pred}$ should be added to the task. Moreover, considering the sleep time, we must subtract the contribution needed to make the prediction τ_{pred}, as the CPU remains active to execute the forecasting model. In contrast, the time spent to transmit data τ_{tx} must be reduced by the transmission suppression ratio SR_{tx}.

Fig. 3 shows the energy saved in transmission, concerning a traditional IoT task, which can be theoretically achieved by the proposed

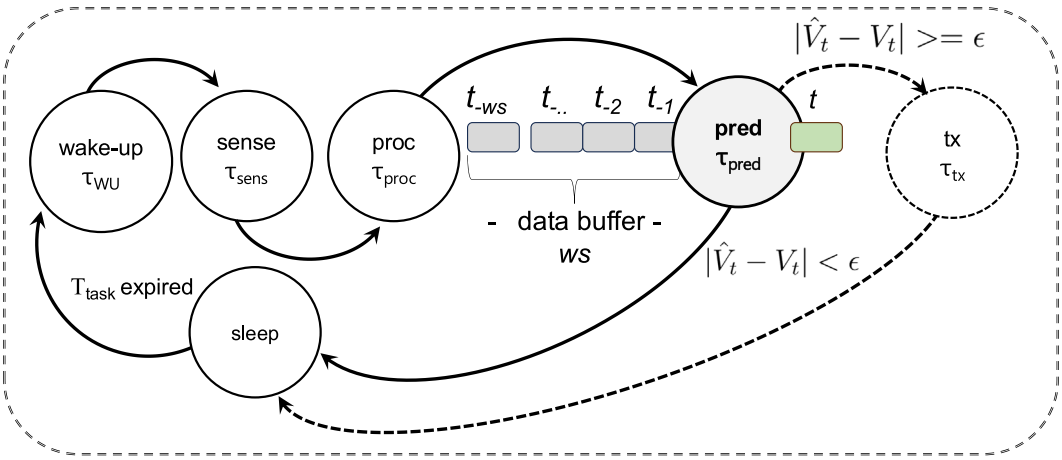


Fig. 2. State diagram of the IoT monitoring task with the DLBDC methodology.

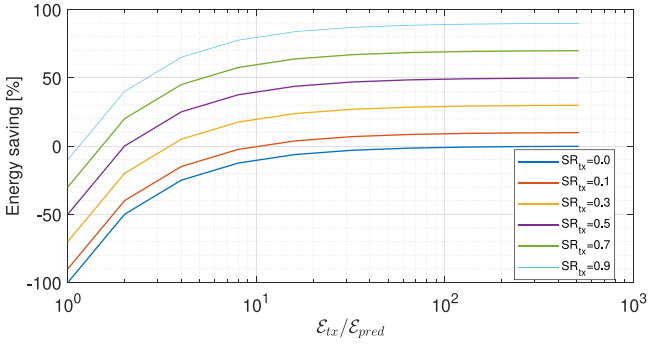


Fig. 3. Theoretical transmission energy saved by the DLDBC approach when varying the $\mathcal{E}_{tx}/\mathcal{E}_{pred}$ ratio, for different values of SR_{tx}.

DLDBC strategy when varying the ratio between the transmission and the prediction energy costs ($\mathcal{E}_{tx}/\mathcal{E}_{pred}$), for different values of SR_{tx}. Considering an energy ratio of about 10x, which is quite common in most microcontrollers, to obtain a significant saving (more than 20%) requires a transmission suppression ratio of at least 30%. Obviously, for further increasing suppression, the energy saving increases proportionally. On the other hand, the $\mathcal{E}_{tx}/\mathcal{E}_{pred}$ ratio plays a fundamental role in the energy balance, as for values lower than 10x, the savings rapidly reduce, making the proposed approach quickly ineffective even for high suppression values. This means that, given the energy cost of transmission, which depends on the available hardware, the model used for prediction must be carefully sized to ensure a convenient $\mathcal{E}_{tx}/\mathcal{E}_{pred}$ ratio. This objective is non-trivial, as more accurate models, capable of achieving a high transmission suppression rate (SR_{tx}), tend to be computationally more complex and thus incur higher energy costs. Consequently, identifying an optimal trade-off between computational energy consumption and predictive performance is critical to the system's overall efficiency.

Concerning the error introduced into the data stored in the cloud, it is important to point out that the ϵ parameter places an upper bound that ensures no data will exceed it.

3.4. Deep learning-driven sensing

The proposed DLDS technique leverages the capability of deep-learning multistep forecasting models to predict, with good precision, several future points, namely ts, using a history buffer of data of length ws. Fig. 4 illustrates the finite-state machine representing an IoT node that implements this technique.

At time t, the model predicts ts future points in a single inference and schedules a wake-up for a time $ts \times T_{task}$ in the future before going to sleep.

When the node wakes up, it shifts the predicted buffer leftward and integrates its contents into the historical data buffer. Then, it acquires a new measurement, V_t. The node compares this measurement with the most recent predicted value in the buffer. This value was originally denoted as $\hat{V}_{t+ts}$ prior to entering the sleep state and now corresponds to $\hat{V}_t$.

If $|\hat{V}_t - V_t| < \epsilon$, it means the last value of the estimated series of length ts is considered a good approximation of the real value. A good approximation does not imply that all previously predicted values also fall within the predefined tolerance threshold. However, considering that in multistep forecasting models the prediction error generally increases with the temporal distance from the point of observation, the probability that earlier estimates remain within tolerance is still high.

In practice, if $\hat{V}_t$ is within the tolerance threshold, then another ts future points are predicted, and the node returns to sleep without sending any data. Conversely, if the last predicted value exceeds the tolerance threshold, the measured value V_t must be sent to the cloud. However, this condition may imply that the entire buffer has deviated beyond the acceptable tolerance. Unfortunately, the previously untransmitted real values are irrecoverable because they were never acquired. The only viable action in such cases is to *realign the buffer* with new data, thereby synchronizing the model with the actual distribution and enhancing the accuracy of future predictions.

The realignment methodology can include filling the data buffer with newly acquired values or inserting the latest data to resume forecasting immediately.

In this work, we employ an intermediate strategy in which we perform a linear interpolation over the existing values in the buffer using the newly acquired value, denoted by $\hat{V}_t$. This corrects the overall trend. This approach eliminates the need to keep the node active for additional ws cycles to fully repopulate the buffer. It also prevents abrupt discontinuities that would result from simply appending the latest value. From an energy standpoint, the realignment contribution can be neglected due to its reduced computational complexity.

In general, the energy consumption of the overall task is strongly affected by the value of ts, which directly determines the duration of the sleep state and, consequently, the activation frequency. For instance, for a task with period equal to T_{task}, the active states are actually executed only T_{task}/ts times. Consequently, the energy model

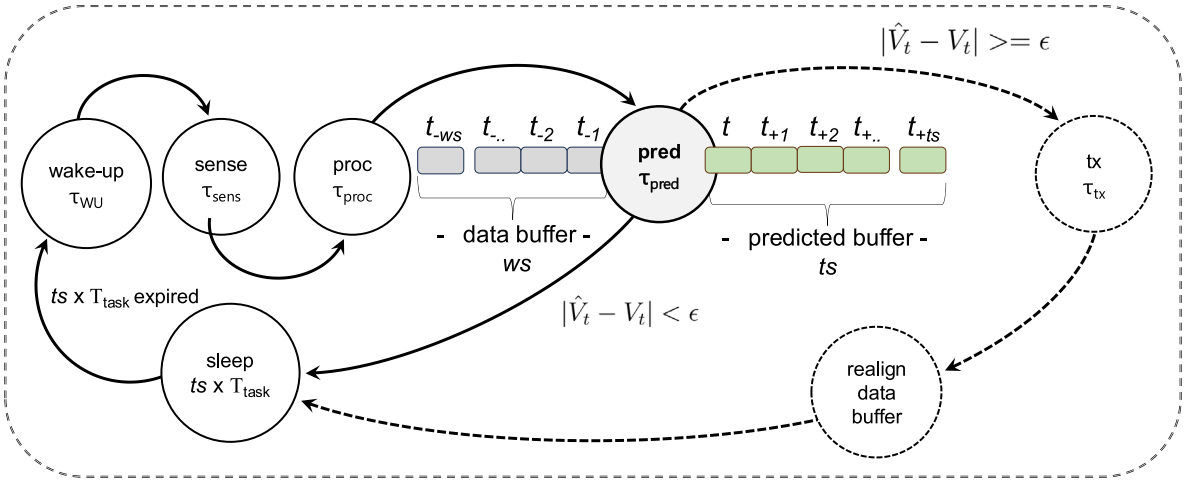


Fig. 4. State diagram of the IoT monitoring task with the DLDS methodology.

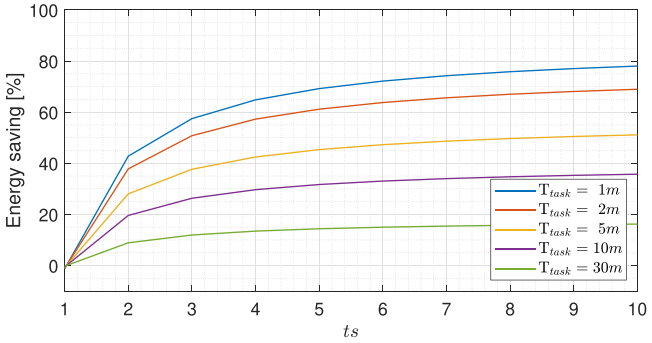


Fig. 5. Theoretical overall energy saved by DLDS approach when varying the number of predicted samples (ts), for different values of T_{task}.

for the DLDS approach becomes:

$$E_{task(DLDS)} = \frac{\mathcal{E}_{WU} + \mathcal{E}_{sens} + \mathcal{E}_{proc} + \mathcal{E}_{pred} + (\mathcal{E}_{tx} + \mathcal{E}_{re}) \cdot (1 - SR_{tx})}{ts} + [T_{task} - \frac{\tau_{WU} + \tau_{sens} + \tau_{proc} + \tau_{pred} + (\tau_{tx} + \tau_{re}) \cdot (1 - SR_{tx})}{ts}] \cdot \mathcal{P}_{sleep} \quad (4)$$

where $\mathcal{E}_{re}$ and τ_{re} are the energy consumption and the length of stay in the *realign* state, respectively. Moreover, neglecting the contribution of the *realign* state allows us to rewrite the previous equation as:

$$E_{task(DLDS)} = \frac{\mathcal{E}_{WU} + \mathcal{E}_{sens} + \mathcal{E}_{proc} + \mathcal{E}_{pred} + \mathcal{E}_{tx} \cdot (1 - SR_{tx})}{ts} + [T_{task} - \frac{\tau_{WU} + \tau_{sens} + \tau_{proc} + \tau_{pred} + \tau_{tx} \cdot (1 - SR_{tx})}{ts}] \cdot \mathcal{P}_{sleep} \quad (5)$$

which, for $ts = 1$, is identical to the model described by Eq. (3) for the DLBDC methodology.

Fig. 5 plots the overall energy saved by the IoT device using the DLDS strategy when varying the number of predicted samples (ts) for different task periods (T_{task}), concerning a traditional IoT task. This plot was obtained using Eq. (5) under the assumption that the power consumption of the active task is 100 times greater than that of the sleep state, i.e., $\mathcal{E}_{tx}/\mathcal{E}_{pred} = 10$, and $SR_{tx} = 0.7$. The task periods are expressed in minutes. Interestingly, most of the energy savings are achieved regardless of the value of T_{task} by predicting just two or three future samples. This is a crucial consideration, as it is reasonable to assume that reconstruction error increases with the number of consecutively predicted points. Consequently, predicting values too far into the future may offer limited benefit and compromise overall data fidelity. The amount of energy saved also depends on the task period. In scenarios with extended sleep durations, the energy contribution of active states becomes negligible, limiting the potential for further energy reduction.

Finally, it is important to point out that, unlike in the DLBDC methodology described in Section 3.3, for the DLDS, the ϵ parameter does not place an upper bound on the actual error in reconstructing the data due to the lack of knowledge of the error on the unacquired values. Therefore, careful tuning of the model parameters, specifically the number of time steps (ts), is crucial to ensuring predictive accuracy and reliable data reconstruction.

3.5. Algorithms implementation

Algorithm 1 presents the pseudo-code for the DLBDC technique, illustrating its implementation on an edge node. The algorithm uses the tolerance threshold ϵ as input, which defines the acceptable margin of error for the monitoring task. Internally, it maintains a circular buffer of size ws, containing the most recent measurements, and a predicted value $\hat{V}_t$ generated during the previous iteration.

Algorithm 1 Pseudo-code of the DLBDC algorithm running on the edge node.

Ensure: buffer ▷ Buffer of size ws
Ensure: $\hat{V}_t$ ▷ Predicted value at time t
Require: ϵ ▷ Tolerance threshold

```
while true do
    V_t ← measureSensor()
    if |V̂_t − V_t| < ε then
        // No need to send
        updateBuffer(buffer, V̂_t)
    else
        sendToServer(V_t)
        updateBuffer(buffer, V_t)
    end if
    V̂_t ← predict(buffer)
    deepSleep(T_task)
end while
```

At each iteration, the edge node samples its sensor to obtain the value of V_t. If this value deviates from the predicted value $\hat{V}_t$ by more than the tolerance ϵ, the actual measurement is transmitted to the server. Otherwise, no transmission occurs.

Regardless of whether data is sent, the circular buffer is updated through the $updateBuffer$ function, which removes the oldest entry and appends either V_t or $\hat{V}_t$, following a First-In-First-Out (FIFO) strategy. Then, a new prediction $\hat{V}_t$ is computed based on the updated buffer. After completing these operations, the device enters a deep-sleep state until the next scheduled iteration.

Conversely, Algorithm 2 delineates the pseudo-code of the DLDS technique executed on the edge node. As in DLBDC, the algorithm takes as input the tolerance threshold ϵ and maintains a circular buffer of size ws to store recent measurements. A key difference is that the internal state also includes a list $\hat{\overline{V}}$ of size ts, which holds the sequence of values predicted in the previous iteration. In this case, the value used for comparison is the last element of the list, $\hat{V}$.

Algorithm 2 Pseudo-code of the DLDS algorithm running on the edge node.

Ensure: buffer ▷ Circular buffer of size ws
Ensure: $\hat{\overline{V}}$ ▷ List of predicted values with size ts
Require: ϵ ▷ Tolerance threshold

```
while true do
    V_t ← measureSensor()
    if |V̂ − V_t| < ε then
        // No need to send
        updateBuffer(buffer, V̂⃗)
    else
        sendToServer(V_t)
        updateBuffer(buffer, V̂⃗)
        realignBuffer(buffer, V_t)
    end if
    V̂⃗ ← predict(buffer)
    deepSleep(ts × T_task)
end while
```

At each iteration, the node samples the sensor to obtain V_t. Then, the node checks whether its value deviates from $\hat{V}$ more than the tolerance ϵ. If the actual measurement lies outside this range (i.e., $|\hat{V} - V_t| < \epsilon$), the actual measurement is transmitted to the server; otherwise, no transmission occurs. When no transmission has occurred, all the predicted values $\hat{\overline{V}}$ are inserted into the circular buffer. Otherwise, the real value V_t is sent to the server, and the appropriate realign technique is applied by calling the function $realignBuffer$. Lastly, the buffer is used to predict the next $\hat{\overline{V}}$ values, and then the node enters a deep sleep state for the next $ts \times T_{task}$ iterations.

As the buffer realign technique, we adopted a simple linear interpolation. The idea is to approximate the intermediate values in the buffer by assuming they lie on a straight line between the first (i.e., the oldest) and the last (i.e., the most recent) element. First, we compute the slope (Δ) of that line by subtracting the value of the last point from the value of the first point, then dividing by the number of elements between them. Next, for each position in the buffer, we calculate its interpolated value as $y_i = y_{first} + i \cdot \Delta$ where y_{first} is the value of the first element, and i is the offset of the current element from the first one.

It is important to note that the DLBDC and DLDS algorithms are intentionally simple in terms of implementation and computational overhead. Their most demanding operations are buffer updates and realignments, which involve linear interpolation and basic list manipulation over lists with a small number of elements. This simplicity makes them well-suited for execution on resource-constrained edge devices, enabling low latency and energy consumption.

We characterized and validated the proposed DLBDC and DLDS algorithms using a dedicated simulation framework, taking as input different datasets containing real-time series of various parameters collected in indoor environments. This framework allows for direct comparison with traditional PBDC techniques that rely on classical modeling approaches. This highlights the advantages and limitations of data-driven solutions in equivalent scenarios.

Finally, a case study was conducted using three popular IoT platforms, in which the algorithms were implemented and installed to empirically quantify energy consumption. The complete source code of the simulation framework, together with the implementations of both algorithms and all scripts used for data preprocessing, model training, and experimental evaluation, is publicly available at: https://github.com/IoTUniurb/deep-learning-based-data-collection.

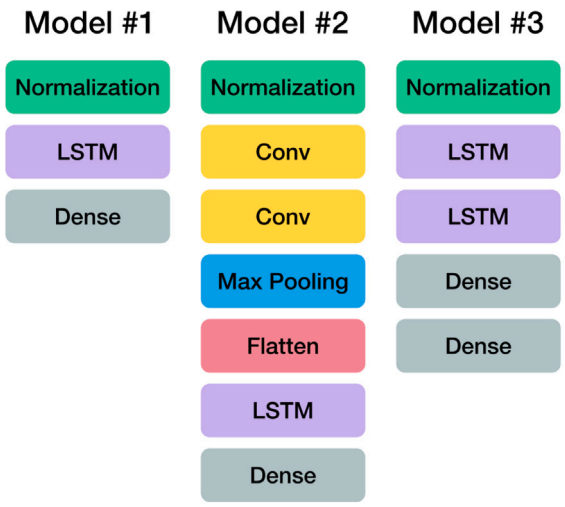


Fig. 6. Diagram highlighting the architecture of the three forecasting models.

Table 2
Parameters of the Deep Learning-based forecasting models.

ID	Net type	Net structure	Params [k]	Net size [kB]
#1	1LSTM_1D	$10U\_tsD$	≈ 0.5	≈ 2
#2	2CNN_1LSTM_1D	$64F\_64F\_10U\_tsD$	≈ 14	≈ 55
#3	2LSTM_2D	$10U\_10U\_30D\_tsD$	≈ 1.7	≈ 7

3.6. Forecasting models

The core component of our sensing approach is a Deep Learning-based forecasting model. To ensure a comprehensive test across different technologies, we designed three distinct forecasting model configurations: Model #1, Model #2, and Model #3. Fig. 6 shows a diagram of the models' architectures, while Table 2 summarizes their corresponding hyperparameter configurations.

Model #1 consists of a network with a single Long Short-Term Memory (LSTM) layer containing 10 units. Model #2 adopts a hybrid architecture, composed of two Convolutional Neural Network (CNN) layers with 64 filters each, followed by a single LSTM layer with 10 units. Finally, Model #3 consists of two LSTM layers, each with 10 units, followed by a dense layer with 30 neurons. It is noteworthy that all three model configurations end with a final dense layer, whose size depends on the target output demanded by each technique. Specifically, a single neuron for DLBDC, and ts neurons for DLDS.

Overall, all three models are lightweight and compact. Model #2 is the largest and most complex, with about 15,000 parameters and a size of about 55 kB. In contrast, Model #1 is the smallest, with roughly 500 parameters and a size of about 2 kB. Finally, Model #3 offers a good trade-off, with approximately 1700 parameters and a size of about 7 kB. These values are approximate, as the exact number of parameters varies slightly depending on the size of the final dense layer, which is determined by the value of ts. However, this variation is minimal.

Model training has been conducted on different environmental datasets. A supervised learning approach was adopted, where the input data were divided into consecutive windows of size ws. For each window, the subsequent ts values were set aside and used as the ground

truth for the predicted values. For the models used in the DLBDC technique, ts was set to 1.

This windowing technique transforms a time series into a set of input/output pairs, simplifying the training process and improving model accuracy. During training, each input window was grouped into batches of size 32, which is a standard practice that reduces the training time required for each epoch.

4. Experimental setup

4.1. Indoor air quality datasets

We chose to evaluate the performance of the proposed strategies in one of the wide domains of the application of the IoT, namely, the Indoor Air Quality (IAQ) monitoring. IAQ is a critical research topic because of its direct influence on human health, cognitive performance, and overall well-being, particularly in enclosed spaces. People typically spend 80%–90% of their time indoors, often at home, but also in schools, offices, and other buildings. Thus, making the quality of indoor air a decisive factor in everyday life [38]. Poor IAQ is increasingly recognized as a significant public health concern, with air pollution contributing to millions of premature deaths worldwide [39].

The dataset used in our experiments was constructed ad hoc, and measured four environmental variables from the classrooms of the Computer Science and Technology degree program at the University of Urbino. These classrooms where selected because they provide a reliable benchmark for real indoor environments, with periods of high occupancy during lectures and periods of inactivity that closely resemble typical indoor dynamics. Sensors were installed in the rooms and configured to collect data every 5 min for a period of over six months, resulting in a comprehensive, high-resolution time series.

Among the different measured physical quantities, we selected the following to include in this study: (i) Carbon Dioxide (CO_2), (ii) Particulate Matter (specifically $PM_{2.5}$), (iii) $Noise$, and (iv) γ $dose\ rate$. CO_2 and $PM_{2.5}$ were chosen because they are well-established air quality indicators, particularly in indoor settings. Carbon dioxide is primarily produced through human respiration and accumulates in poorly ventilated enclosed spaces. On the other hand, particulate matter can originate from various indoor sources, such as cooking, smoking, and using heating appliances. These particles have a size of less than 2.5 μm and can be inhaled deeply into the lungs, posing serious health risks. $Noise$ levels were also included, as they are closely associated with human activities. Excessive noise exposure in urban environments is known to impair concentration, increase stress levels, and contribute to both mental and physical health issues. The γ $dose\ rate$ represents the level of gamma radiation present in the surrounding environment. When it is derived from discrete gamma photon counts over short time intervals, such as those recorded by Geiger–Müller counters, as in this case, the underlying distribution of the detected events typically follows a Poisson distribution, where the variance equals the mean. Although this variable is not commonly included in studies of indoor environments, its high variability and unpredictability make testing the robustness of forecasting models particularly challenging.

To effectively train and evaluate the forecasting models, each dataset was split into two main segments: a training segment and a simulation segment. The training segment was then divided into training and testing subsets using a 75/25 ratio. Following standard practices, the training set was further divided into training and validation subsets using a 90/10 split. Conversely, the simulation segment was explicitly reserved to evaluate the performance of the DLBDC and DLDS techniques on previously unseen data.

Both training and simulation segments were obtained by partitioning the original datasets into non-overlapping blocks of 500 samples each. Then, ten randomly selected blocks were concatenated to form the simulation set. The remaining blocks constituted the training set. This procedure ensures that the models are tested on data distributions unaffected by seasonal periodicity. In summary, the training segment consists of 47,000 samples collected at 5-minute intervals, covering a total duration of over 3900 h, or approximately 160 days. In contrast, the simulation segment contains 5000 samples, spanning more than 400 h, or roughly 17 days. All the datasets, along with the scripts used for processing them, are made available at: https://github.com/IoTUniurb/deep-learning-based-data-collection/tree/main/datasets

4.2. State-of-the-art comparison techniques

The performance of the proposed methodologies was compared to that obtained by two of the state-of-the-art PBDC techniques grounded in traditional approaches: (i) Derivative-Based Prediction, and (ii) Kalman Filter.

Derivative-Based Prediction (DBP) was originally proposed by Raza et al. [40,41], as a lightweight technique for Wireless Sensor Networks (WSN). It employs a simple linear forecasting model that can capture data trends while remaining efficient and resistant to the noise in the data. In many real-world scenarios, the physical quantities evolve smoothly over time, allowing a linear model to accurately approximate their local behavior. Unlike approaches that focus on minimizing the fitting error over historical data points, DBP prioritizes consistency with the observed trend, making it particularly resilient to noise and outliers.

Model generation begins with an initialization phase where the sensor node collects a *learning window* of m data points. From this window, the first and last l samples are identified as *edge points*. The model computes the slope of the line connecting the average of the edge points at the beginning and end of the window, called Δ. This operation is analogous to a derivative, hence the name Derivative-Based Prediction.

After initialization, the node enters a runtime phase where it continuously buffers the last m samples in a sliding window. Each new measurement is compared with the predicted value produced by the current model to determine its validity. The model is considered valid as long as the prediction error stays within a fixed tolerance value ϵ_V, or if larger errors do not occur for more than ϵ_T consecutive samples. If these conditions are violated, a new model is computed and its updated slope Δ is transmitted to the central server.

Kalman Filters (KF) are recursive data filtering algorithms that are widely used to estimate the hidden internal state of a dynamic system over time. This type of filter provides an efficient way to infer the current state of a system at time t, using a combination of the state estimate from the previous time step $(t-1)$ and noisy observations. Kalman Filters are particularly effective in scenarios where the system is affected by uncertainty in both its evolution and the measurements.

Their internal operation is divided into two main phases: (i) prediction phase, and (ii) update phase. In the prediction phase, the system estimates the new internal state and its uncertainty based on the previous estimate using the following equations:

$$x_t = F x_{t-1} + \omega_t \tag{6}$$

$$P_t = F P_{t-1} F^T + Q_t \tag{7}$$

here, x_t represents the new internal state of the system, and P_t is the corresponding predicted covariance matrix. The term x_{t-1} refers to the old (*a priori*) estimate of the state at the previous time step. The matrix F is the state-transition matrix, which is responsible for the evolution of the system's state. Finally, ω_t denotes the process noise, and Q_t is its associated covariance matrix.

After the prediction phase, the predicted value can be estimated using the following equation:

$$z_t = H x_{t-1} + v_t \tag{8}$$

where the matrix H is the observation model that maps the internal state space to the observed space, and v_t is the measurement noise.

Subsequently, the update phase refines the predicted state by incorporating the actual measurement to reduce the estimation error. This is done through the following set of equations:

$$K = P_t H^T (H P_t H^T + R)^{-1} \tag{9}$$

$$x_t = x_t + K(z_t - H x_t) \tag{10}$$

$$P_t = (I - KH) P_t \tag{11}$$

Here, K is the Kalman gain, which determines the weight to be given to the new measurement. Lastly, R is the covariance matrix of the observation noise, and I is the identity matrix.

In the context of PBDC, Jain et al. propose a Dual Kalman Filter (DKF) approach based on Kalman Filters [42]. For each data stream i, two synchronized Kalman filters are instantiated: one at the central server (KF_s^i) and one at the remote sensor (KF_r^i). Due to the low computational cost of these filters, the central server can maintain a separate filter instance for each sensor without significant overhead. During execution, the remote sensor only transmits a new measurement if the local prediction error exceeds a predefined threshold δ_i, thereby reducing communications. Furthermore, the system can accept an optional parameter F_i to adjust the filter's responsiveness, allowing for further adaptation to specific application requirements.

4.3. Accuracy metrics

To validate the forecasting models during both the training and testing phases, and to quantify the performance of the DLBDC and DLDS techniques, we employ two metrics capable of measuring the reconstruction error introduced by the models: (i) Mean Absolute Error (MAE), and (ii) Mean Absolute Percentage Error (MAPE).

The MAE measures the average absolute deviation between predicted and observed values, expressed in the same units as the data. It is defined as

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{12}$$

where y_i represents real values measured by the sensors and $\hat{y}_i$ represents the values predicted by the forecasting model. Similarly, the MAPE measures the average forecasting error relative to the true values and is expressed as a percentage. This allows for scale-independent comparisons across datasets. It is defined as:

$$MAPE = \frac{1}{N} \sum_{i=1}^{N} |\frac{y_i - \hat{y}_i}{y_i}| \tag{13}$$

While MAE treats all deviations equally, MAPE emphasizes relative error magnitude and is sensitive to small or zero target values, which can lead to instability. Together, these metrics provide a more complete picture of model accuracy by combining interpretability and comparability.

4.4. Simulation and training

To properly train and evaluate the forecasting models, the environmental dataset was partitioned into training and simulation segments, as described in Section 4.1. The training segment was then used for model training. The forecasting models were trained using a supervised learning approach, meaning that each element of the training set must be associated with a target output that the model is expected to learn.

The time series was transformed into a sequence of consecutive windows of length ws. Each window was paired with the immediately subsequent ts values, which served as the ground-truth. Since each window is independent from the other, the windows were generated with a stride of one time step to maximize the number of available samples.

We trained the forecasting models using the open-source deep-learning library Keras [43] running on top of TensorFlow [44]. Specifically, *Keras 2.15* and *TensorFlow 2.15* were used on a high-performance workstation equipped with two Intel® Xeon® Silver 4314 Central Processing Units (CPUs) and three NVIDIA® A100 Graphical Processing Units (GPUs). Once trained, the models were optimized for deployment on resource-constrained devices. First, we converted the full models into a compressed, hardware-friendly, representation using LiteRT [45] (formerly known as TensorFlow Lite). Next, the converted models were further processed using the LiteRT for Microcontrollers framework [46], which translates them into statically allocated, array-like C structures. These structures can be embedded in microcontroller firmware and executed without the need for an operating system or dynamic memory allocation.

The simulation framework was implemented using the Python programming language at version 3.10 and executed on the same workstation used for model training. In this phase, we used the simulation dataset, which was intentionally excluded from training, to ensure that the models were evaluated on previously unseen samples. This provides a realistic assessment of their generalization capabilities.

Unlike the training phase, which relies on supervised learning, the simulation emulates the real operating conditions of the algorithms processing one value at a time and managing a circular buffer of size ws. To properly simulate the two approaches, we implemented Algorithms 1 and 2 in Python, emulating the $measureSensor$ function by simply reading the next value in the dataset at each time step. At the same time, the transmissions to the central server and the deep-sleep operations were replaced with no-op actions to isolate the behavior from hardware-specific aspects. Throughout execution, we recorded all operations performed by the algorithms to track relevant performance metrics. All simulations were carried out using fixed random seeds to ensure reproducibility.

4.5. Hardware platforms of the case study

To evaluate the energy savings provided by the proposed methodology, we implemented an ad-hoc monitoring task to characterize the power consumption of three IoT devices: the ESP32, the MCXN947, and the Raspberry Pi Pico 2 W.

The **ESP32** is a microcontroller developed by Espressif® that is highly popular in the field of IoT and wearable applications due to its ease of use, low price, and low power consumption [35]. Applications for the ESP32 are written entirely in C or C++ using the Espressif IoT Development Framework (ESP-IDF) [47]. In our experiments, we utilized the ESP32-DevKitC. This device features an Xtensa dual-core processor running at 240 MHz, 400 kB of RAM, and 4 MB of flash storage. Also, it supports Wi-Fi and Bluetooth connectivity and can directly interface with a wide range of sensors via its integrated GPIO pins and a series of communication protocols, including I2C, PWM, and UART.

The **Raspberry Pi Pico 2 W** is a low-cost and flexible microcontroller board developed by Raspberry Pi, designed as a compact development platform for the RP2350 MCU [37]. The board features a unique dual-core, dual-architecture design integrating both an Arm® Cortex-M33 and a RISC-V Hazard3 running at up to 150 MHz. The board includes 520 kB of SRAM, 4 MB of external flash, and on-board 2.4 GHz wireless connectivity with support for Wi-Fi and Bluetooth LE. This platform exposes 40 GPIO pins supporting a comprehensive set of peripherals, including SPI, I2C, UART, PWM channels, and a 12-bit Analog to Digital Converter (ADC). Applications for the Pico can be developed in C and C++ using the official SDK [48] or in Python using MicroPython [49], making it a versatile platform for embedded and IoT prototyping.

The **MCXN947** is a compact and scalable development board produced by NXP and designed for rapid prototyping of industrial IoT

Table 3
Characterization of the power and energy consumption of the different operation states of the three representative IoT devices (ESP32, Pico 2 W, and MCXN947).

State	ESP32			Pico 2 W			MCXN947		
	$\mathcal{P}$ [mW]	τ [ms]	$\mathcal{E}$ [mJ]	$\mathcal{P}$ [mW]	τ [ms]	$\mathcal{E}$ [mJ]	$\mathcal{P}$ [mW]	τ [ms]	$\mathcal{E}$ [mJ]
$WakeUp$	155	2775	430	68	5800	394	224	3262	731
$Sense$	375	54	20	68	82	6	414	39	16
$Proc.$	387	4	2	77	6	1	431	3	1
$Pred_{DL}$	387	61	24	77	226	17	431	42	18
$Pred_{DBP}$	387	30	12	77	131	10	431	19	8
$Pred_{KF}$	387	120	46	77	426	33	431	88	38
Tx	469	690	324	576	603	347	418	125	52
$Sleep$	29	–	–	11	–	–	31	–	–

applications [36]. It integrates a dual-core Arm® Cortex-M33 microcontroller unit (MCU) running at 150 MHz and equipped with up to 2 MB of flash and 512 KB of full-ECC RAM. Additional features include Ethernet connectivity, flash memory expansion, and standard SPI, I2C, and UART interfaces exposed through a set of 124 GPIO pins. A notable characteristic of this platform is its support for multiple hardware accelerators, including a Digital Signal Processing (DSP) accelerator and an eIQ® Neutron N1-16 Neural Processing Unit (NPU), which enables efficient execution of advanced machine learning workloads. Applications for this device are developed in C++ using the MCUXpresso Integrated Development Environment (IDE) [50].

During the execution, we powered the devices using an NGMO2 Rohde & Schwarz dual-channel power supply [51] connected in series with an 2.5 Ω shunt resistor. To measure the energy consumption, we sampled the voltage drop across the shunt resistor by means of a National Instruments NI-DAQmx PCI-6251 16-channel data acquisition board connected to a BNC-2120 shielded connector block [52,53].

Table 3 reports the average power, time, and energy consumption measured for each operational state across the three evaluated boards. Since the load of the prediction state ($Pred$) depends on the strategy in use, we characterized it for the proposed deep learning (DL) models as well as the reference systems (i.e., DBP and Kalman Filter).

The $WakeUp$ state is the most energy-intensive across all devices. Its high impact primarily stems from its long duration, which can be up to 5.8 s for the Pico 2 W. In fact, when waking up from deep sleep, the devices execute a full boot sequence, including sensor initialization and network setup. Despite its low power draw, the Pico 2 W takes longer to wake up due to its lower operating frequency. Conversely, the MCXN947 is the fastest but consumes the most energy, reaching up to 731 mJ. The ESP32 offers intermediate behavior, combining the shortest wake-up time with moderate energy consumption.

The sampling ($Sense$) and processing ($Proc$) of both sensors reduce energy consumption across all boards, thanks to their short duration. The ESP32 has the highest values of 20 mJ and 2 mJ, respectively.

The prediction phases ($Pred$) associated with the three evaluated strategies introduce a modest overhead. On the ESP32, the proposed DL models consume an average of about 24 mJ, while the reference systems DBP and Kalman Filter consume, respectively, 12 mJ and 46 mJ. Notably, the inference time remains remarkably low for model #3, which includes four layers with approximately 1.7k parameters, thanks to the TensorFlow Lite runtime and ESP-NN optimizations. A similar trend is observed on the Pico 2 W and the MCXN947. Despite their disparate power profiles (77 mW and 431 mW, respectively), their resulting energy costs are similar. This is due to the Pico 2 W's lower operating frequency, which consequently dictates the longest prediction time. The MCXN947 achieves the fastest inference by exploiting its accelerator. The ESP32, by contrast, provides an optimal balance between execution time and efficiency.

In terms of energy consumption, transmission (Tx) is the second operation. The ESP32 and the Pico 2 W have comparable transmission costs of 324 and 347 mJ, respectively, due to their similar Wi-Fi transceivers. In contrast, the MCXN947 exhibits substantially lower transmission energy, approximately one order of magnitude smaller, due to its use of Ethernet communication, a considerably more power-efficient method than Wi-Fi. Given the difference in energy required for data transmission and the minimal cost of data forecasting, PBDC techniques can yield significant energy savings when properly configured.

Furthermore, exploiting the exceptionally low power consumption of the deep sleep mode (approximately 11 mW for the Pico 2 W), as in the DLDS technique, can lead to additional and potentially substantial energy savings.

Table 4
Forecasting performance of the proposed deep-learning models with respect to the reference approaches.

Dataset	Model	MAE	Unit	MAPE
CO_2	Model #1	14.250	[ppm]	0.022
	Model #2	7.766		0.012
	Model #3	**3.597**		**0.006**
	DBP	9.518		0.014
	KF	11.251		0.017
$PM_{2.5}$	Model #1	0.233	[µg/m3]	0.063
	Model #2	0.202		0.057
	Model #3	**0.121**		**0.035**
	DBP	0.203		0.054
	KF	0.199		0.052
$Noise$	Model #1	0.076	[mV]	0.039
	Model #2	0.064		0.033
	Model #3	**0.037**		**0.019**
	DBP	0.064		0.033
	KF	0.066		0.034
γ $dose$ $rate$	Model #1	33.881	[nSv/h]	0.122
	Model #2	34.155		0.106
	Model #3	**18.247**		**0.064**
	DBP	32.695		0.096
	KF	30.288		0.087

5. Results

In this section, we provide results from three sets of experiments. The first one reports the forecasting performance of the proposed deep learning models, while the following two are aimed at characterizing the DLBDC and DLDS strategies, both concerning energy saving and measurement errors.

5.1. Performance of the forecasting models

Table 4 shows the performance of the deep-learning models to be used in the DLBDC strategy (i.e., the model forecasting a single future value $ts = 1$) expressed both in MAE and MAPE concerning the four IAQ datasets compared to the reference systems.

In particular, Model #3 produces the most accurate results across all datasets, achieving a remarkable percentage error of only 0.58%, corresponding to a MAE of only about 3.6 ppm in CO_2 forecasting. However, performance decreases when estimating quantities with less

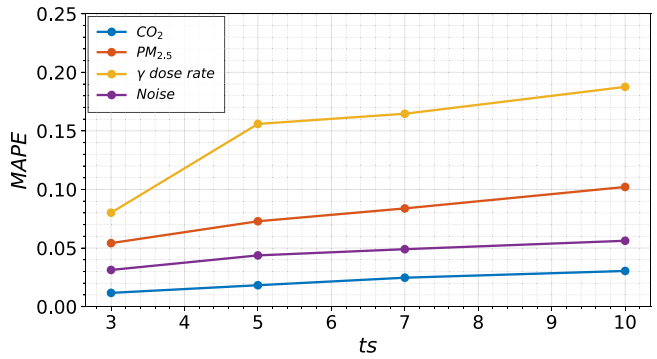


Fig. 7. Multistep forecasting error when increasing the number of samples to be predicted at a time (ts).

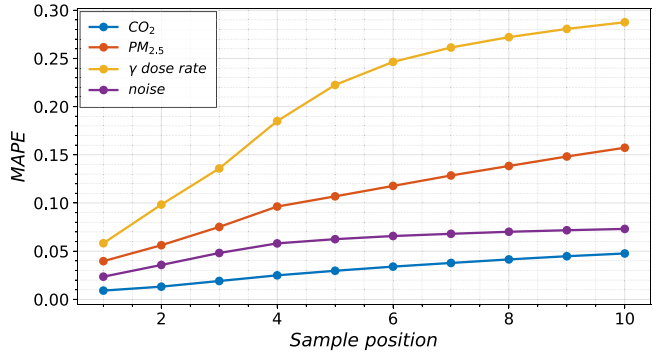


Fig. 8. Multistep forecasting error related to each sample position in a model configured with $ts = 10$.

defined and predictable trends, such as particulate matter ($PM_{2.5}$) or radioactivity (γ *dose rate*), where the average percentage error reaches approximately 3.5% and 6.4%, respectively. In general, a model that is more effective at predicting the future should have a higher suppression ratio. Therefore, the subsequent experiments for the characterization of the DLBDC strategy were all conducted with Model #3.

Compared to the two reference techniques (DBP and Kalman Filter), the deep-learning model outperforms in all four datasets, with a greater gap in the quantities more difficult to predict.

Fig. 7 reports the models' characterization in terms of MAPE, for the multistep forecasting methodology (i.e., when predicting more than a single future step at a time, as in the DLDS strategy) when increasing the value of ts.

As expected, the longer the future period under consideration, the larger the average error of the model. Furthermore, it appears evident that, for quantities that are more difficult to predict, such as radioactivity, the increase is more rapid than for others, soon reaching percentage error values above 15%. Fortunately, as highlighted in the theoretical analysis of Section 3.4, the DLDS strategy achieves good energy efficiency already with small values of ts (between 3 and 4), which allows the prediction error to remain sufficiently low.

To better characterize the prediction error in the multistep forecasting, we evaluate the MAPE related to each sample position in a model configured with $ts = 10$. The main idea is to investigate how the prediction error is distributed along the ts samples. Fig. 8 shows the MAPE versus the sample positions when predicting the four environmental quantities. Interestingly, the trends are significantly different. For example, the error for CO_2 appears to increase linearly with a modest slope, indicating that the model effectively predicts over extended periods. In contrast, the other quantities show a sharp increase in error in the first four positions, which then tends to reduce as the position progresses. Furthermore, the slope of the curves continues to reflect the intrinsic difficulty of prediction, with higher values for radioactivity.

When implementing the DLDS strategy, it is particularly interesting to study the error distribution along the sample positions, as the last point of the multistep prediction, denoted as $\hat{V}_{t+ts}$, is used to evaluate the accuracy of the estimation of all previous samples. This process helps to determine whether the model should be realigned, so properly sizing the ts parameters appears to be of fundamental importance, especially if the error trend in relation to the sample position is non-linear.

5.2. DLBDC performance

Fig. 9 shows the Suppression Rate (SR) calculated using the selected datasets for the DLBDC compared to the reference systems as the tolerance threshold ϵ varies, expressed as relative error (δ). It is important to note that the SR is calculated as the ratio between the actual number of suppressed messages and the total number of messages that should have been sent without a PBDC mechanism.

As the relative error accepted by the server (δ) increases, the number of packets transmitted by each PBDC technique decreases, since a larger portion of predicted values fall within the acceptable threshold. This, in turn, leads to an increase in SR. Furthermore, the DLBDC technique exhibited superior efficacy in reducing the number of messages transmitted to the server across the three datasets (CO_2, $PM_{2.5}$, and $Noise$) in comparison to the two reference systems. For instance, in the CO_2 dataset, the proposed method achieves a suppression ratio of approximately 82% for a tolerance value of only 3%. At the same time, DBP obtains a suppression of about 78%, and KF remains at about 27%. This indicates that, when an error of 3% is accepted in CO_2 monitoring, the proposed method ensures that 82% of packets will not be transmitted.

It is well understood that the more effectively a model captures the underlying dynamics of a signal, the higher its SR will be. Therefore, for signals that exhibit slow temporal variation and are more predictable, we observe high SR values. A typical example is indoor CO_2 concentration, which increases gradually due to human respiration and is thus easier to model accurately. In contrast, signals such as ambient $Noise$ or γ *dose rate* are inherently more volatile and less predictable, leading to consistently lower SR values. Moreover, in the case of γ *dose rate*, for instance, the KF demonstrates performance comparable to the proposed method, whereas the DBP strategy shows significant performance improvement when the relative error exceeds 3%. In summary, for all datasets, when considering a $\delta = 0.03$, the suppression rate accounted for the proposed DLBDC strategy ranges between 82% and 38%, thereby demonstrating its potential to reduce energy transmission in the IoT domain.

Fig. 10 shows plots that illustrate how MAPE changes as the values of δ increase. The MAPE is calculated using the estimated samples received by the remote server. It is important to note that the imposed value of $delta$ represents an upper limit to the MAPE, as it determines the threshold above which the actual sample needs to be transmitted instead of using the estimated one. Therefore, the closer the MAPE is to the value of δ, the better the technique performs. Compared to the others, the proposed methodology appears to be the best, with differences of almost 3 percentage points for the CO_2 dataset and almost 2 points for the $Noise$ dataset. Since the value of δ represents a tolerance value accepted by design, any reduction in MAPE from it represents an additional gain in data accuracy. For example, using the proposed DLBDC strategy with a 5% tolerance results in a MAPE of just over 2% for CO_2 monitoring at the cloud server. Similarly, for $PM_{2.5}$, $Noise$, and γ *dose rate*, the reconstructed data never show a percentage error above 5%, even with a tolerance of up to 10%.

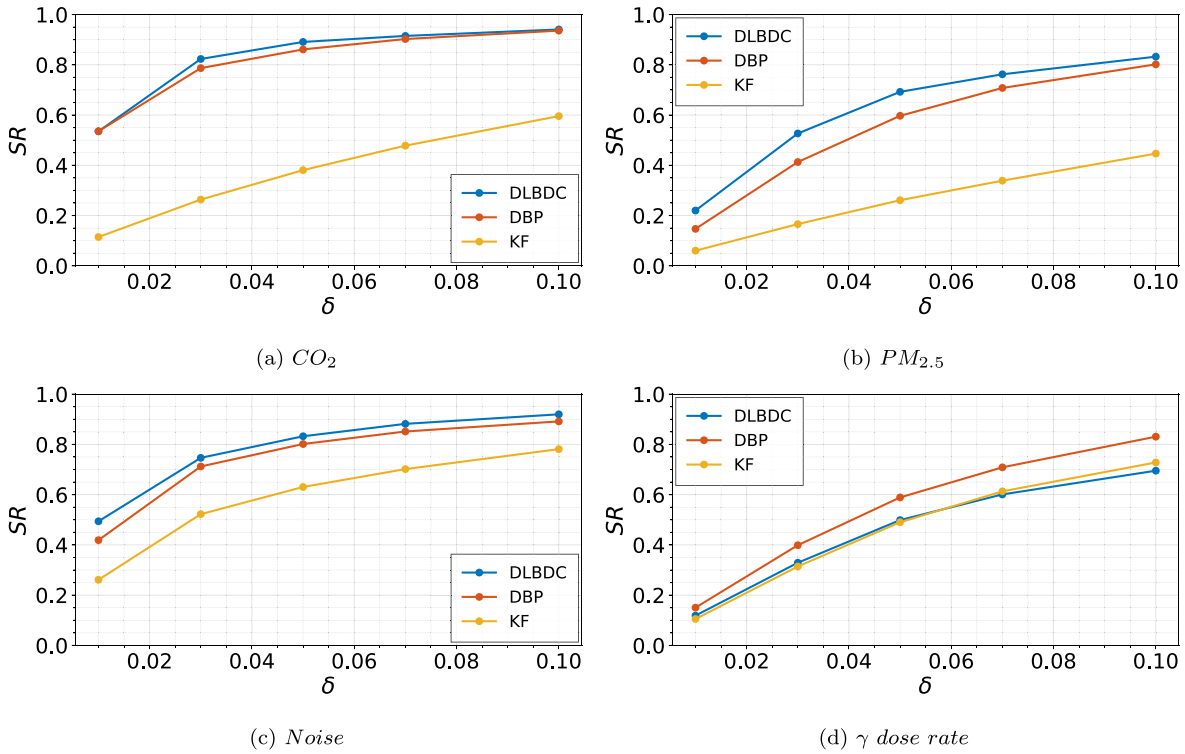


Fig. 9. Suppression rate of the DLBDC strategy with respect to the reference systems when varying the tolerance threshold ϵ expressed as relative error δ.

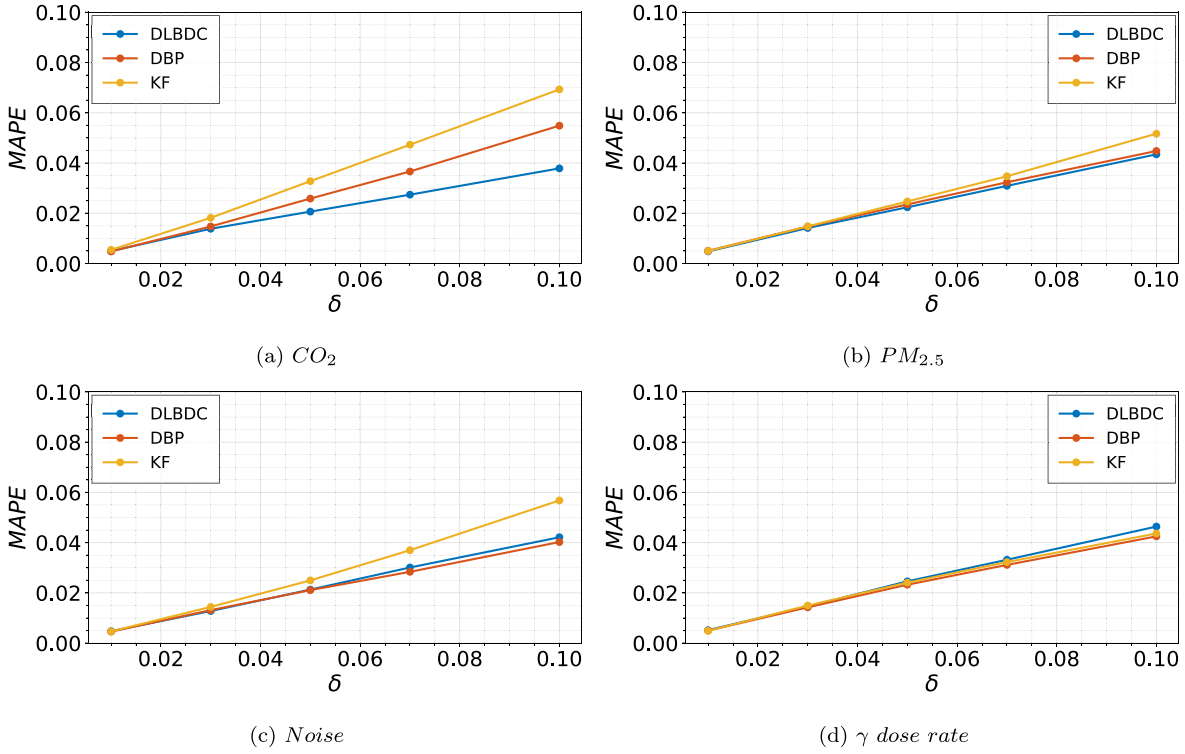


Fig. 10. Variation in the server-side reconstruction error calculated for increasing values of δ.

In conclusion, it is demonstrated that the DLBDC technique has the capacity to reduce data transmission by over 80% through the introduction of a reconstruction error of approximately 1.4% on data accuracy for quantities that vary slowly and predictably as CO_2. Conversely, in the worst-case scenario, it is possible to achieve a reduction of up to 38% by introducing an error margin of just over 1.5%.

To evaluate the energy saving associated with the reduction in data transmission, we calculate the active energy expenditure both with and

Table 5
Energy saving, achieved on top of the three hardware platforms, for the DLBDC task compared to DBP and KF.

Dataset	Technique	MAPE	ESP32	Pico 2W	MCXN947
			Saving [%]	Saving [%]	Saving [%]
CO_2	DLBDC	**0.014**	**31.68**	**36.21**	**3.09**
	DBP	0.015	31.25	35.03	4.11
	KF	0.018	5.42	8.33	*−3.07*
$PM_{2.5}$	DLBDC	**0.014**	**19.55**	**22.06**	**1.43**
	DBP	0.015	16.10	18.36	1.28
	KF	0.015	1.49	3.12	*−4.43*
$Noise$	DLBDC	**0.013**	**28.95**	**32.24**	**3.02**
	DBP	0.013	28.06	32.13	4.17
	KF	0.014	16.88	20.15	*−1.08*
γ *dose rate*	DLBDC	0.015	11.02	13.23	0.34
	DBP	**0.014**	**15.65**	**17.81**	**1.02**
	KF	0.015	7.16	10.03	*−2.02*

without the PBDC strategies using, respectively, Eqs. (3) and (1) applied to the state characterization values of the devices reported in Table 3.

Table 5 reports the total energy consumed in the active states (i.e., without accounting for sleeping energy), alongside the energy saved, and the measured MAPE of the DLBDC technique compared to the reference systems. The computation was conducted for a monitoring task that spanned over 400 h, with a 5-minute period, running on top of each dataset with $\delta = 0.03$. The most cost-effective options for energy savings and MAPE are emphasized in bold text.

In three out of the four datasets, the proposed technique provided higher energy savings and lower MAPE across both the ESP32 and the Pico 2 W, with energy reductions ranging from 19% to 36% and MAPE values between 1.4% and 1.3%. For the γ *dose rate* dataset, the DBP outperformed the DLBDC by about 3%–4% in energy savings while achieving a nearly identical MAPE. On the MCXN947, however, the DLBDC did not yield consistent benefits, reaching a maximum saving of about 3%. All strategies demonstrated marginal or even negative savings, since the reduction in power consumption due to transmission suppression, did not offset the energy cost of prediction. This is primarily due to the lower $\mathcal{E}_{tx}/\mathcal{E}_{pred}$ ratio, a result of the high efficiency of data transmission over Ethernet of the MCXN947 chip. The reported results, in summary, demonstrate that the proposed DLBDC technique, in IoT wireless devices, can save a non-negligible part of the active energy in environmental monitoring at a cost of an accuracy reduction not exceeding 2% of the measurement.

5.3. DLDS performance

The DLDS strategy is characterized by several configuration parameters that need to be tuned to evaluate its exploitability. In particular, the size of the buffer (ws), the length of the predicted values (ts), and the relative error δ. A set of preliminary experiments identified $ws = 5$ as the optimal value, which has therefore been kept constant. In the following, the dependency of the performance on both ts and δ has been evaluated.

Fig. 11 shows how the SR varies with increasing values of ts and different configurations of δ. Notably, it is evident that the percentage of saved packets never falls below 70% for any of the benchmark datasets.

In particular, as the value of ts increases, SR always grows, reaching its maximum value at $ts = 10$. Additionally, increased δ affects performance more at lower ts. This effect can be attributed to the fact that the SR mostly depends on the number of samples estimated at a time, while the value of δ determines whether only the last of the ts value should be transmitted and whether the buffer should be realigned.

Fig. 12 reports how the MAPE of the reconstructed data on the server-side varies for increasing value of ts. For all the datasets, the system shows an almost monotonous growth in the value of MAPE. Again, the δ parameter does not produce very pronounced effects, especially in those hard-to-predict quantities such as γ *dose rate*. In the latter case, the value predicted by the model at position ts almost never satisfies the acceptance threshold δ for any given ts, always generating a data transmission and buffer realignment.

To investigate the tradeoff between energy consumption and reconstructed MAPE, we calculated the active energy expenditure of the system with the DLDS strategy using Eq. (5) as in the previous section.

Fig. 13 shows the Pareto charts of the four datasets, plotting active energy versus the server-side reconstructed error. These charts only consider the energy consumed by the ESP32, which is the intermediate reference point in terms of performance among the three evaluated platforms.

Pareto charts are valuable tools for visualizing relationships between cost metrics and identifying optimal trade-off points. In particular, points closer to the origin of the axes minimize both metrics. Note that each value of δ leads to a different family of points, represented by a different marker. Each point corresponds to a different ts value. In all datasets except for CO_2, the best compromise is obtained with a value of $ts = 5$. For CO_2, however, the best value is $ts = 7$. Concerning the δ parameter, the best family point for CO_2 and $Noise$ appears to be $\delta = 0.01$, which seems to contribute to lowering the MAPE thanks to the low tolerance. For $PM_{2.5}$ and γ *dose rate*, however, the points are much closer together, with barely appreciable performance superiority for $\delta = 0.07$ or 0.10. The reduction in overall performance when low error tolerances are applied is likely due to the frequent buffer realignment, which may cause the prediction model to become unstable.

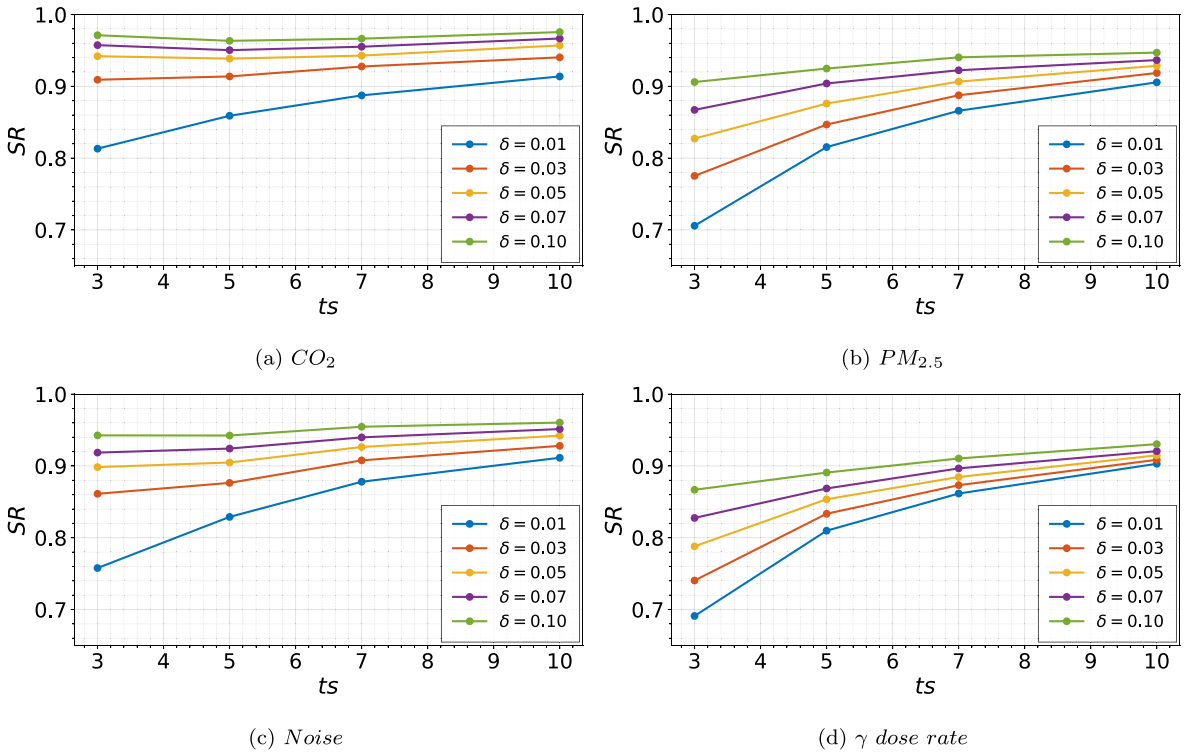
Table 6 shows the optimal balance between energy savings and reconstruction error for each dataset, assuming the cloud server can tolerate a measurement error of up to 3%. In this case, none of the platforms exhibits negative savings, since skipping sensor acquisitions and anticipating entry into the deep-sleep state always reduces energy consumption, regardless of the energy cost of transmission. However, the MCXN947 consistently achieves the lowest savings among the three boards.

For example, in the case of CO_2, the energy savings are about 89% thanks to the high value of ts, which can be utilized. Conversely, for $PM_{2.5}$, the time step must be reduced to 2 to meet the error constraints, resulting in an energy savings of around 58%. The $Noise$ dataset meets the constraint with $ts = 3$, resulting in an energy saving of about 75%. However, when it comes to radioactivity, even when employing the minimum time step (i.e., $ts = 2$), it is not possible to adhere to the error constraint. In this case, an error of up to 5% was measured, resulting in energy savings of approximately 54%.

In summary, the findings of the present experiments demonstrate that the DLDS strategy has the capacity to reduce the energy consumption of IoT nodes in active states by a substantial amount, for example, by up to 89% in terms of CO_2 emissions. However, it is also evident that this strategy cannot be applied indiscriminately to any quantity. Essentially, it depends on the time correlation of the data, which determines its predictability. The case of the γ *dose rate* is representative of a quantity that follows a Poisson distribution, reflecting the random nature of radioactive decay and detection, which is difficult to predict in time.

6. Limitations and future work

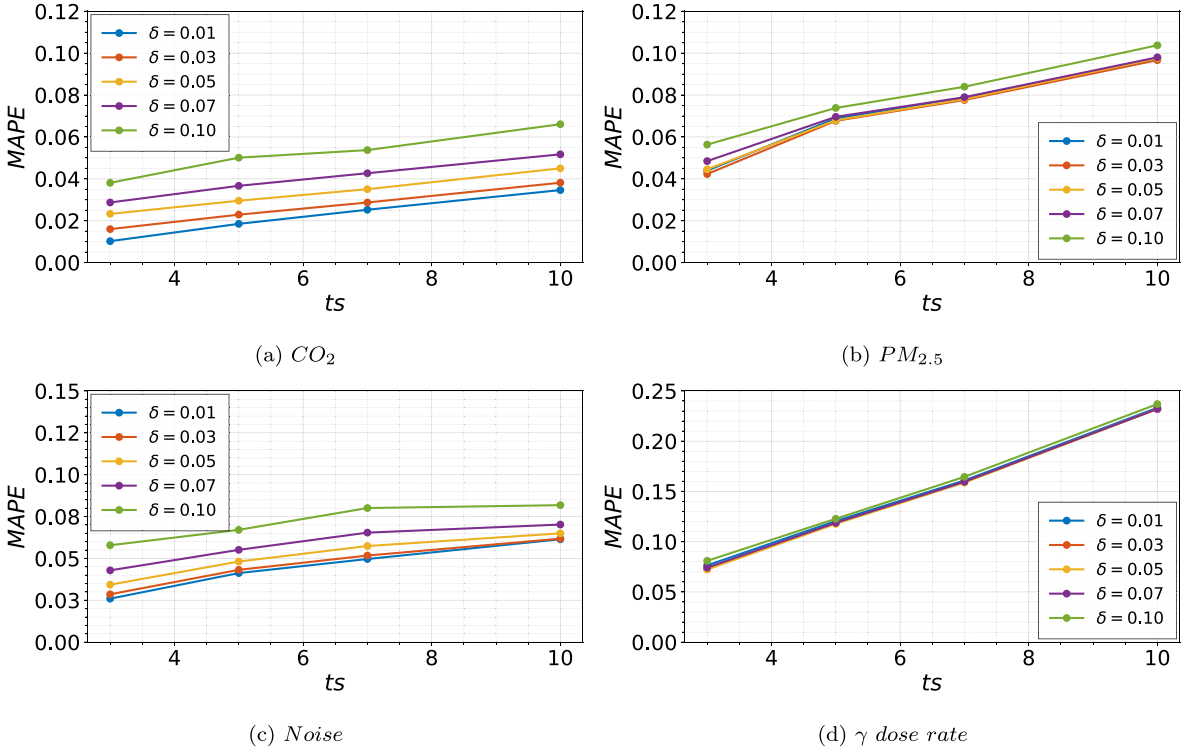
The proposed work presents some limitations that must be carefully considered when assessing its applicability and scalability in real-world deployments. First, a forecasting model must be created and trained for each physical quantity that is to be monitored. Although this approach incurs increased initial setup costs, it ensures that the predictive logic is matched to the specific dynamics of each signal, enhancing the overall



(a) CO_2 (b) $PM_{2.5}$

(c) $Noise$ (d) γ $dose\ rate$

Fig. 11. Suppression rate for increasing value of ts combined with different configurations of δ.

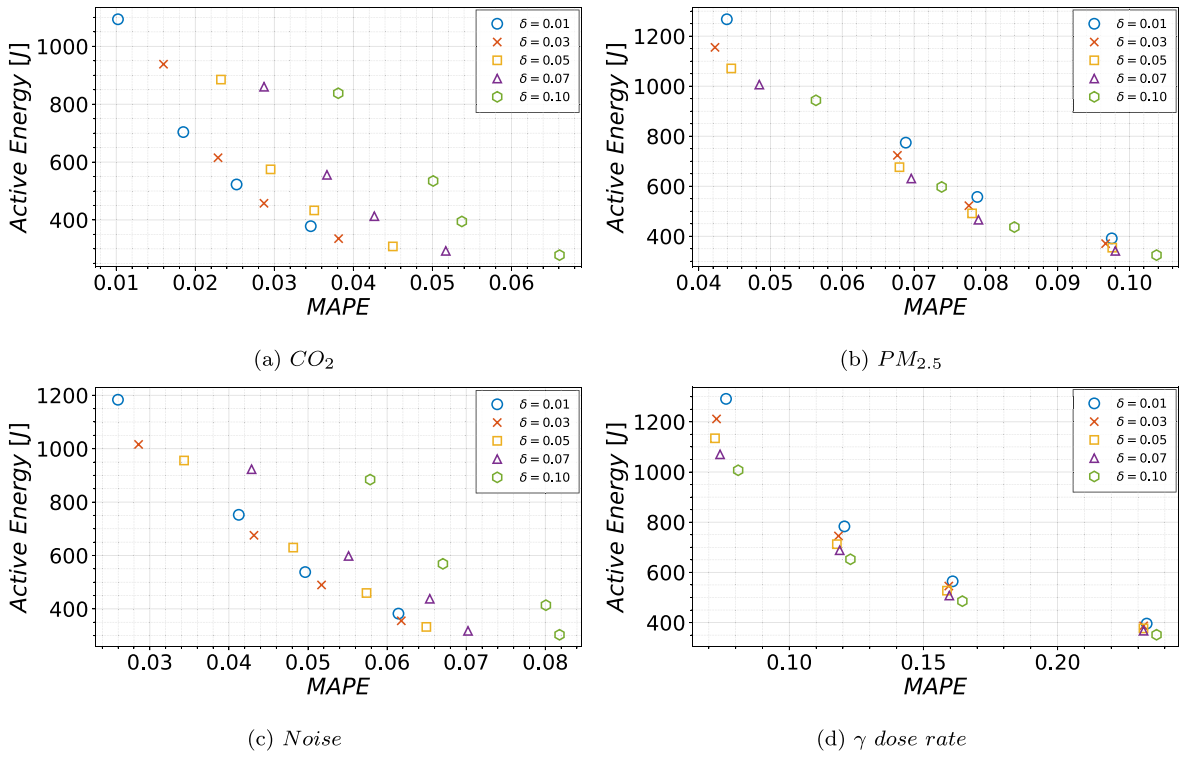


(a) CO_2 (b) $PM_{2.5}$

(c) $Noise$ (d) γ $dose\ rate$

Fig. 12. Error of the reconstructed data on the server-side for increasing value of ts.

effectiveness and reliability of the system. Second, the effectiveness of the strategy depends on the temporal correlation of the sensed data. When such a correlation exists, the method can substantially reduce energy consumption. However, when the signal is dominated by stochastic fluctuations, such as the γ $dose\ rate$, the predictive accuracy, and thus the energy savings, decrease. Third, only the DLDS strategy can achieve a significant advantage in terms of energy saving for IoT devices that provide a cable internet connection, such as those used in industrial IoT. Lastly, unlike traditional IoT monitoring, the cloud server must maintain a copy of the history buffer for every sensor node in order to predict future values. This leads to a memory footprint that grows linearly with the number of nodes. Although this overhead



(a) CO_2 (b) $PM_{2.5}$

(c) $Noise$ (d) γ $dose\ rate$

Fig. 13. Pareto charts plotting active energy versus the server-side reconstructed error.

Table 6
Reconstructed error together with the energy saving for $\delta = 0.03$.

Dataset	MAPE	Steps [#]	ESP32		Pico 2W		MCXN947	
			En. [J]	Saving [%]	En. [J]	Saving [%]	En. [J]	Saving [%]
CO_2	0.029	7	457	88.23	425	**89.11**	568	78.34
$PM_{2.5}$	0.030	2	1155	57.28	1570	**58.66**	1993	22.19
$Noise$	0.028	3	1015	74.05	937	**75.42**	799	69.14
γ $dose\ rate$	*0.053*	*2*	*1,793*	*54.33*	*1,695*	*55.10*	*2,012*	*21.07*

remains moderate in typical deployments, it becomes relevant when scaling to very large installations.

Future research will focus on several aspects of the proposed framework. First, testing the approaches on additional wireless IoT communication protocols, such as LoRa, Bluetooth LE, or Zigbee, would help assess the general applicability across wireless communication channels with different power consumption. Second, it would be interesting to explore alternative network architectures for the forecasting models, including Gated Recurrent Units (GRU), or compact transformer models, and to compare their performance with other deep learning-based techniques. Finally, investigating ways to further optimize the strategies for less predictable signals.

7. Conclusions

The increased use of IoT devices represents a significant opportunity to reduce global energy consumption and enable more energy-efficient pervasive computing.

This study addresses the rising energy consumption of Internet of Things devices by introducing deep-learning-based predictive sensing to enhance sustainability. Two lightweight methods were developed: (i) Deep Learning-Based Data Collection (DLBDC): a forecasting model, installed on both device and cloud, allows for transmitting sensor data only when actual values deviate beyond a tolerance threshold, minimizing communication energy. (ii) Deep Learning-Driven Sensing (DLDS): predicts several future readings in advance, allowing devices to remain in deep-sleep states longer, cutting sensing and wake-up energy.

Implemented on three microcontrollers (i.e., Espressif ESP32, Raspberry Pi Pico 2 W, NPX MCXN947) and tested with indoor air-quality datasets (CO_2, $PM_{2.5}$, $Noise$, γ $dose\ rate$), the models, particularly a two-layer LSTM, outperformed traditional Derivative-Based Prediction and Kalman Filter approaches. In particular, the experimental results show energy savings up to 36% for DLBDC and up to 89% for DLDS, while maintaining data errors below 3%.

The work demonstrates that TinyML-enabled predictive sensing can drastically reduce IoT energy demand, promoting environmentally responsible and energy-efficient smart systems.

CRediT authorship contribution statement

Lorenzo Calisti: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Investigation, Conceptualization. **Chiara Contoli:** Writing – review & editing, Writing – original draft, Validation, Methodology. **Emanuele Lattanzi:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Methodology, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Lorenzo Calisti reports financial support was provided by Papalini S.p.A. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was partially funded by Papalini S.p.A., Via Paolo Borsellino 9, 61032 Fano (PU) - Italy

Data availability

All the datasets used in this work are available at: https://github.com/IoTUniurb/deep-learning-based-data-collection/tree/main/datasets.

References

[1] A. Rejeb, K. Rejeb, S. Simske, H. Treiblmaier, S. Zailani, The big picture on the internet of things and the smart city: a review of what we know and what we need to know, Internet Things 19 (2022) 100565, http://dx.doi.org/10.1016/j.iot.2022.100565.

[2] E.H. Houssein, M.A. Othman, W.M. Mohamed, M. Younan, Internet of things in smart cities: Comprehensive review, open issues, and challenges, IEEE Internet Things J. 11 (21) (2024) 34941–34952, http://dx.doi.org/10.1109/JIOT.2024.3449753.

[3] A. Khanna, S. Kaur, Internet of things (IoT), applications and challenges: a comprehensive review, Wirel. Pers. Commun. 114 (2020) 1687–1762, http://dx.doi.org/10.1007/s11277-020-07446-4.

[4] K.M. Hosny, W.M. El-Hady, F.M. Samy, Technologies, protocols, and applications of internet of things in greenhouse farming: A survey of recent advances, Inf. Process. Agric. 12 (1) (2025) 91–111, http://dx.doi.org/10.1016/j.inpa.2024.04.002.

[5] A.U. Rehman, S. Lu, M.B. Bin Heyat, M.S. Iqbal, S. Parveen, M.A. Bin Hayat, F. Akhtar, M.A. Ashraf, O. Khan, D. Pomary, et al., Internet of things in healthcare research: Trends, innovations, security considerations, challenges and future strategy, Int. J. Intell. Syst. 2025 (1) (2025) 8546245, http://dx.doi.org/10.1155/int/8546245.

[6] Statista, Internet of things (IoT) connected devices installed base worldwide from 2015 to 2030, 2025, (Accessed 15 July 2025), https://www.statista.com.

[7] N. Jain, M. Mandot, Energy efficient strategies in internet of things: an overview, in: Proceedings of 4th International Conference on Information and Communication Technology for Competitive Strategies, ICTCS 2019, 2020, pp. 585–592, http://dx.doi.org/10.1201/9781003052098.

[8] M. Medojević, N. Marković, A. Rikalović, Modeling AI-driven IoT energy consumption: From device-level forecasts to system-level dynamics, in: Proceedings of the 10th International Conference on Smart and Sustainable Technologies, SpliTech 2025, 2025, pp. 1–7, http://dx.doi.org/10.23919/SpliTech65624.2025.11091761.

[9] E. A. R. de Oliveira, F. Delicato, A. R. da Rocha, M. Mattoso, A real-time and energy-aware framework for data stream processing in the internet of things, in: Proceedings of the 6th International Conference on Internet of Things, Big Data and Security, IoTBDS 2021, 2021, pp. 17–28, http://dx.doi.org/10.5220/0010370100170028.

[10] Y.L. Cheng, M.H. Lim, K.H. Hui, Impact of internet of things paradigm towards energy consumption prediction: A systematic literature review, Sustain. Cities Soc. 78 (2022) 103624, http://dx.doi.org/10.1016/j.scs.2021.103624.

[11] E. Hittinger, P. Jaramillo, Internet of things: energy boon or bane? Science 364 (6438) (2019) 326–328, http://dx.doi.org/10.1126/science.aau8825.

[12] M.A. Albreem, A.M. Sheikh, M.H. Alsharif, M. Jusoh, M.N. Mohd Yasin, Green internet of things (gIoT): Applications, practices, awareness, and challenges, IEEE Access 9 (2021) 38833–38858, http://dx.doi.org/10.1109/ACCESS.2021.3061697.

[13] L. Farhan, R.S. Hameed, A.S. Ahmed, A.H. Fadel, W. Gheth, L. Alzubaidi, M.A. Fadhel, M. Al-Amidie, Energy efficiency for green internet of things (IoT) networks: A survey, Network 1 (3) (2021) 279–314, http://dx.doi.org/10.3390/network1030017.

[14] R. Gambheer, M.S. Bhat, Optimized compressed sensing for IoT: Advanced algorithms for efficient sparse signal reconstruction in edge devices, IEEE Access 12 (2024) 63610–63617, http://dx.doi.org/10.1109/ACCESS.2024.3396494.

[15] H. Djelouat, A. Amira, F. Bensaali, Compressive sensing-based IoT applications: A review, J. Sens. Actuator Netw. 7 (4) (2018) 45, http://dx.doi.org/10.3390/jsan7040045.

[16] H.A. Selmy, H.K. Mohamed, W. Medhat, A predictive analytics framework for sensor data using time series and deep learning techniques, Neural Comput. Appl. 36 (11) (2024) 6119–6132, http://dx.doi.org/10.1007/s00521-023-09398-9.

[17] M.A. Ala'anzy, R. Zhanuzak, R. Akhmedov, N. Mohamed, J. Al-Jaroodi, Dynamic load balancing for enhanced network performance in IoT-enabled smart healthcare with fog computing, IEEE Access 12 (2024) 188957–188975, http://dx.doi.org/10.1109/ACCESS.2024.3516362.

[18] U. Raza, A. Bogliolo, V. Freschi, E. Lattanzi, A.L. Murphy, A two-prong approach to energy-efficient WSNs: Wake-up receivers plus dedicated, model-based sensing, Ad Hoc Networks 45 (2016) 1–12, http://dx.doi.org/10.1016/j.adhoc.2016.03.005.

[19] B. Płaczek, Prediction-based data reduction with dynamic target node selection in IoT sensor networks, Future Gener. Comput. Syst. 152 (2024) 225–238, http://dx.doi.org/10.1016/j.future.2023.11.007.

[20] G.M. Dias, B. Bellalta, S. Oechsner, A survey about prediction-based data reduction in wireless sensor networks, ACM Comput. Surv. 49 (3) (2016) 1–35, http://dx.doi.org/10.1145/2996356.

[21] D. Tulone, S. Madden, PAQ: Time series forecasting for approximate query answering in sensor networks, in: Proceedings of the European Workshop on Wireless Sensor Networks, vol. 3868, 2006, pp. 21–37, http://dx.doi.org/10.1007/11669463_5.

[22] B. Gedik, L. Liu, P.S. Yu, ASAP: An adaptive sampling approach to data collection in sensor networks, IEEE Trans. Parallel Distrib. Syst. 18 (12) (2007) 1766–1783, http://dx.doi.org/10.1109/TPDS.2007.1110.

[23] M. Giordano, S. Cortesi, P.-V. Mekikis, M. Crabolu, G. Bellusci, M. Magno, Energy-aware adaptive sampling for self-sustainability in resource-constrained iot devices, in: Proceedings of the 11th International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems, ENSSys 2023, 2023, pp. 65–71, http://dx.doi.org/10.1145/3628353.3628545.

[24] Y. Ben-Aboud, D.B. Licea, M. Ghogho, A. Kobbane, On adaptive sampling algorithms for IoT devices, in: IEEE International Conference on Communications, ICC 2021, 2021, pp. 1–7, http://dx.doi.org/10.1109/ICC42927.2021.9500326.

[25] Y.W. Law, S. Chatterjea, J. Jin, T. Hanselmann, M. Palaniswami, Energy-efficient data acquisition by adaptive sampling for wireless sensor networks, in: Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly, 2009, pp. 1146–1151, http://dx.doi.org/10.1145/1582379.1582631.

[26] W. Lalouani, M. Younis, I. White-Gittens, R.N. Emokpae Jr., L.E. Emokpae, Energy-efficient collection of wearable sensor data through predictive sampling, Smart Health 21 (2021) 100208, http://dx.doi.org/10.1016/j.smhl.2021.100208.

[27] A.M. Rahmani, J. Tanveer, F.S. Gharehchopogh, S. Rajabi, M. Hosseinzadeh, A novel offloading strategy for multi-user optimization in blockchain-enabled mobile edge computing networks for improved internet of things performance, Comput. Electr. Eng. 119 (2024) 109514, http://dx.doi.org/10.1016/j.compeleceng.2024.109514.

[28] K. Moghaddasi, S. Rajabi, F.S. Gharehchopogh, A. Ghaffari, An advanced deep reinforcement learning algorithm for three-layer D2D-edge-cloud computing architecture for efficient task offloading in the internet of things, Sustain. Comput.: Inform. Syst. 43 (2024) 100992, http://dx.doi.org/10.1016/j.suscom.2024.100992.

[29] T. Pötsch, L. Pei, K. Kuladinithi, C. Goerg, Model-driven data acquisition for temperature sensor readings in wireless sensor networks, in: Proceedings of the 2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP 2014), 2014, pp. 1–6, http://dx.doi.org/10.1109/ISSNIP.2014.6827658.

[30] U. Raza, A. Camerra, A.L. Murphy, T. Palpanas, G.P. Picco, What does model-driven data acquisition really achieve in wireless sensor networks? in: 2012 IEEE International Conference on Pervasive Computing and Communications, 2012, pp. 85–94, http://dx.doi.org/10.1109/PerCom.2012.6199853.

[31] Q. Han, S. Mehrotra, N. Venkatasubramanian, Energy efficient data collection in distributed sensor environments, in: Proceedings of the 24th International Conference on Distributed Computing Systems, ICDCS 2004, 2004, pp. 590–597, http://dx.doi.org/10.1109/ICDCS.2004.1281626.

[32] A.M. Hussein, A.K. Idrees, R. Couturier, A distributed prediction-compression-based mechanism for energy saving in IoT networks, J. Supercomput. 79 (15) (2023) 16963–16999, http://dx.doi.org/10.1007/s11227-023-05317-w.

[33] S. Pandey, K. Dubey, R. Dubey, S. Kumar, EEDCS: Energy efficient data collection schemes for IoT enabled wireless sensor network, Wirel. Pers. Commun. 129 (2) (2023) 1297–1313, http://dx.doi.org/10.1007/s11277-023-10190-0.

[34] M.A.P. Putra, A.P. Hermawan, D.-S. Kim, J.-M. Lee, Data prediction-based energy-efficient architecture for industrial iot, IEEE Sensors J. 23 (14) (2023) 15856–15866, http://dx.doi.org/10.1109/JSEN.2023.3280485.

[35] Espressif, ESP32, 2025, (Accessed 05 July 2025), https://www.espressif.com/en/products/socs/esp32.

[36] NXP, MCXN947 datasheet, 2025, (Accessed 07 December 2025), https://www.nxp.com/docs/en/data-sheet/MCXNx4xDS.pdf.

[37] Raspberry Pi, Raspberry Pi Pico 2 W, 2025, (Accessed 07 December 2025), https://www.raspberrypi.com/documentation/microcontrollers/pico-series.html.
[38] S. Dimitroulopoulou, M.R. Dudzińska, L. Gunnarsen, L. Hägerhed, H. Maula, R. Singh, O. Toyinbo, U. Haverinen-Shaughnessy, Indoor air quality guidelines from across the world: An appraisal considering energy saving, health, productivity, and comfort, Environ. Int. 178 (2023) 108127, http://dx.doi.org/10.1016/j.envint.2023.108127.
[39] R. Corlan, R. Balogh, I. Ionel, S. Kilyeny, The importance of indoor air quality (IAC) monitoring, in: International Conference on Applied Sciences (ICAS 2020), vol. 1781, 2021, 012062, http://dx.doi.org/10.1088/1742-6596/1781/1/012062.
[40] U. Raza, A. Camerra, A.L. Murphy, T. Palpanas, G.P. Picco, Practical data prediction for real-world wireless sensor networks, IEEE Trans. Knowl. Data Eng. 27 (8) (2015) 2231–2244, http://dx.doi.org/10.1109/TKDE.2015.2411594.
[41] A. Bogliolo, V. Freschi, E. Lattanzi, A.L. Murphy, U. Raza, Towards a true energetically sustainable WSN: A case study with prediction-based data collection and a wake-up receiver, in: Proceedings of the 9th IEEE International Symposium on Industrial Embedded Systems, SIES 2014, 2014, pp. 21–28, http://dx.doi.org/10.1109/SIES.2014.6871181.
[42] A. Jain, E.Y. Chang, Y.-F. Wang, Adaptive stream resource management using kalman filters, in: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, 2004, pp. 11–22, http://dx.doi.org/10.1145/1007568.1007573.
[43] F. Chollet, et al., Keras, 2015, (Accessed 05 July 2025), https://keras.io.
[44] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, 2015, (Accessed 15 July 2025), https://www.tensorflow.org.
[45] Google A.I., LiteRT, 2025, (Accessed 05 July 2025), https://ai.google.dev/edge/litert.
[46] Google A.I., LiteRT for Microcontrollers, 2025, (Accessed 05 July 2025), https://ai.google.dev/edge/litert/microcontrollers/overview.
[47] Espressif, Espressif IoT Development Framework, 2025, (Accessed 05 July 2025), https://idf.espressif.com.
[48] Raspberry Pi, The C/C++ SDK, 2025, (Accessed 07 December 2025), https://www.raspberrypi.com/documentation/microcontrollers/c_sdk.html.
[49] Raspberry Pi, MicroPython, 2025, (Accessed 07 December 2025), https://www.raspberrypi.com/documentation/microcontrollers/micropython.html.
[50] NXP, MCUXpresso Integrated Development Environment (IDE), 2025, (Accessed 07 December 2025), https://www.nxp.com.
[51] Rohde & Schwarz, NGMO2 datasheet, 2025, (Accessed 05 July 2025), https://www.rohde-schwarz.com/us/brochure-datasheet/ngmo.
[52] National Instruments, PC-6251 Datasheet, 2020, Accessed 05 July 2025, http://www.ni.com/pdf/manuals/375213c.pdf.
[53] National Instruments, Installation Guide BNC-2120, 2020, (Accessed 05 July 2025), http://www.ni.com/pdf/manuals/372123d.pdf.