A Low-Cost IoT Sensor for Indoor Monitoring with Prediction-Based Data Collection

Paolo Capellacci, Lorenzo Calisti, Emanuele Lattanzi
Department of Pure and Applied Sciences, University of Urbino, Urbino, Italy

***Abstract*—The proliferation of Internet of Things technologies has revolutionized the landscape of indoor environmental monitoring, offering opportunities to enhance comfort, health, and energy efficiency. This paper presents the development and implementation of a low-cost IoT sensor system designed for indoor monitoring with a Machine Learning-driven prediction-based data collection approach. Leveraging deep learning algorithms, the IoT device predicts significant environmental changes and dynamically adjusts the data collection frequency to optimize energy consumption and data transmission. Experimental results demonstrate the system's ability to accurately predict environmental variations, resulting in a reduction in data transmission and power usage up to 96% without compromising the monitoring quality. The findings highlight the potential of prediction-based data collection as a viable solution for sustainable and effective indoor environment monitoring on low-cost IoT devices.**

***Keywords*—IoT; indoor monitoring; prediction-based data collection; deep-learning**

I. Introduction

The widespread diffusion of Internet of Things (IoT) technologies has transformed various contests, ranging from environmental monitoring to smart buildings, agriculture, and transportation. In the indoor monitoring field, IoT systems enable real-time data collection from numerous sensors, providing opportunities to enhance comfort, health, and energy efficiency. By monitoring parameters such as temperature, humidity, air quality, and occupancy, these systems can optimize environmental conditions for the well-being of occupants while minimizing energy consumption [1]. However, traditional IoT-based environmental monitoring systems often rely on continuous data collection, leading to excessive power consumption and inefficient data transmission, especially in resource-constrained environments [2]. The challenge of balancing data accuracy and energy efficiency is particularly significant for low-cost IoT systems, where prolonged battery life and reduced data transmission costs are critical. Recent advancements in Machine Learning (ML) offer potential solutions to this issue. ML techniques can enable IoT devices to predict environmental changes dynamically, allowing for adaptive data collection strategies that optimize energy use without compromising the quality of monitoring. This paradigm shift from static to Prediction-Based Data Collection (PBDC) has the potential to enhance both the sustainability and effectiveness of IoT systems in indoor environments [3].

In particular, PBDC retains the original sampling frequency of the application while reducing energy consumption by minimizing the amount of data that needs to be transmitted [4]. This is achieved by constructing a model of the sensed data, which is then used both at the sensor and at the sink to estimate the sampled data points. When a new sample is obtained, the sensor checks if it falls within the predefined error tolerance. If it does, no further action is required (i.e. data are not sent); if not, a new model is created and sent to the sink.

In this paper, we present the development and implementation of a low-cost IoT sensor system for indoor environmental monitoring that leverages machine learning to optimize data collection and transmission. By employing ML algorithms to predict significant environmental changes, our system dynamically adjusts the frequency of data collection, resulting in reduced power usage and data transmission overhead. Experimental results demonstrate that the proposed system accurately predicts environmental variations, achieving energy savings of up to 96% while maintaining high data quality.

The contributions of this work are twofold:

- We introduce a low-cost IoT sensor for indoor monitoring capable of easily measuring environmental parameters, and we characterize it on a real deployment.
- We implement a PBDC strategy by means of deep-learning models, and we deeply characterize its efficiency in terms of average error and transmission ratio with respect to a traditional approach.

The rest of the paper is organized as follows: Section II provides related work on IoT low-cost sensors and PBDC-enabled devices. Section III describes the hardware-software IoT architecture and provides a detailed description of the PBDC mechanism we implemented using ML models. In Section IV, we provide a detailed description of the case study based on a real prototype deployment, and we discuss the results and findings, while Section V concludes the paper.

II. Related Works

In the literature, several authors have addressed the research and development of tools and systems for monitoring environmental metrics, particularly focused on measuring and controlling Indoor Air Quality (IAQ). Among these, Saini et al. [5] presents a systematic review of the current state of the art in IoT-based IAQ monitoring systems. They examined numerous studies published between 2015 and 2020, finding that most of the research has been conducted in countries such as Portugal, China, India, and Malaysia. Their findings also reveal that a significant portion of studies focuses on monitoring comfort parameters and CO_2 levels, with many systems being implemented using Arduino or Raspberry Pi.

Several studies have investigated the use of edge and fog computing devices in air quality monitoring. For instance, Bianconi et al. [6] proposed an IoT system to improve citizens' well-being in Savona, Italy. Their findings highlight the network's capability to monitor well-being over the long term and promptly respond to critical situations when necessary. Similarly, Narayana et al. [7] introduced an advanced real-time environmental monitoring system leveraging IoT and wireless sensors. This system employs innovative techniques for monitoring water treatment and air quality, capturing real-time data on air, water, waste, energy, and soil, thereby making a significant contribution to sustainable living.

Focusing on indoor environments, various systems have been developed to monitor air quality using IoT technology, capable of tracking $PM_{2.5}$, CO_2, temperature, and humidity simultaneously [8]. Some of these systems also include predictive capabilities, providing alerts for potentially hazardous conditions [9]. Additionally, Sung et al. [10] applied a combination of air quality indices, including the carbon dioxide index and the air quality index from the American Society of Heating, Refrigerating, and Air-Conditioning Engineers, to simulate the impact of these factors on indoor air quality.

AI-based approaches are also gaining momentum, with models such as Long Short-Term Memory (LSTM) being applied to predict future CO_2 concentrations, thus improving IAQ management systems by enabling preemptive actions. Other studies have integrated multi-headed Convolutional Neural Network (CNN) models to enhance air quality predictions through transfer learning, which boosts system efficiency, even in environments with limited data [11].

In sensor-based IoT applications, the majority of the energy budget is consumed during the transmission of collected data [12]. In the literature, various methods, such as data compression, data quantization, and data aggregation, are available [13]. However, we focus on PBDC due to its simplicity and proven effectiveness in scenarios involving periodic data collection [14].

In PBDC, the original sampling period required by the IoT application is preserved, but the total amount of transmitted data is reduced [15] by creating a forecasting model for the sensed data. This model is shared among both the IoT sensors and the collecting server. At the IoT device level, each new sample is checked to see if it falls within acceptable error margins. If it does, no action is taken (i.e. no data transmission occurs); if it doesn't, a new model is generated and sent to the sink. If the model accurately reflects the data trend, network communication, and energy can be significantly reduced, sometimes by as much as 99% [16], [17], [18], [19]. Since the early days of IoT research, various models have been explored. Probabilistic models [20], [21] approximate data with user-specified confidence but require domain experts to encode special data characteristics. Other techniques include linear regression [22], [14], autoregressive models [23], and Kalman filters [24], but these demand substantial memory and computational resources, making them challenging to implement on resource-constrained devices.

With the introduction of artificial intelligence tools, many researchers began exploring a novel approach to time series forecasting based on machine learning. They employed classical machine learning methods and models, such as support vector machines (SVM), random forests (RF), and backpropagation (BP), for time series forecasting tasks achieving better results [25], [26].

In the last decades, the rapid development of IoT technology has led to the generation of massive amounts of time series data. These extensive time series datasets often exhibit high dimensionality and nonlinearity, making it challenging to achieve the desired prediction accuracy using previous machine learning or statistical methods. To address this challenge, researchers have introduced deep learning methods such as LSTMs, Recurrent Neural Networks (RNNs), and Gated Recurrent Units (GRUs) to tackle the prediction and analysis of massive multivariate time series data achieving significant success [27], [28]. In the IoT domain, forecasting by means of deep-learning techniques has been prevalently used to reconstruct missing or corrupted data in order to enhance system reliability [29], [30], [31], [32]. The computational demands of deep learning tools present a significant challenge in meeting the constraints of tiny sensors. Consequently, in-sensor direct deep-learning forecasting approaches seem to be uncommon. Recently, Lalouani et al. proposed a study on the energy efficiency of an ECG wearable sensor through predictive sampling [33]. In particular, the authors used an LSTM network to assess whether a sample can be predicted using the previous ones, subject to a certain inaccuracy bound. Samples that can be accurately predicted are skipped, while the rest are buffered to be sent to the gateway. The authors simulate the energy efficiency of the method by computing the number of skipped samples starting from an ad-hoc created ECG dataset.

The various methods described so far offer significant advantages but present challenges limiting their effectiveness. For example, edge and IoT devices provide rapid responses but face scalability and reliability issues, which may be imprecise in complex environments. Data compression and aggregation techniques reduce transmission costs but introduce the risk of losing critical information. Finally, machine learning techniques ensure accurate predictions but require high computational resources, making them less suitable for IoT devices with limited capabilities.

This paper presents a low-cost IoT sensor platform capable of monitoring indoor air quality with an integrated PBDC algorithm. The hardware-software stack of the platform is described, and the performances are deeply characterized. Our work differs from Lalouani's in two ways: first, we introduce a prototype of a general-purpose IoT platform deployed on a real testbed for indoor monitoring; second, we deeply characterize the PBDC strategy on top of the real testbed by highlighting the tradeoff between measurement error and the ratio of saved packets with respect to a traditional approach.

III. The Proposed Architecture

In this section, we introduce the IoT sensor architecture, examining both its hardware and software components. We provide a detailed analysis of the main processing unit, the ESP32, and discuss the various environmental sensors installed. The main schematic of the device is reported in Fig. 1 while the 3D model, comprising the enclosure box, is shown

in Fig. 2. Additionally, we present the software stack that manages all operational phases of the IoT sensor. Lastly, we describe the PBDC system capable of dynamically choosing when sending data to reduce energy consumption.

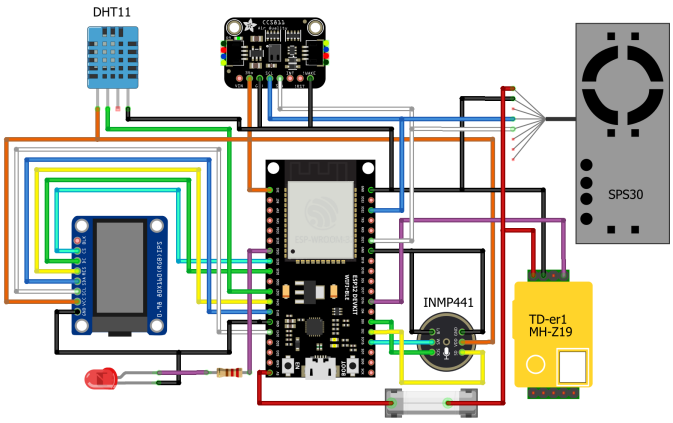


Fig. 1. Schematic of the IoT device together with the main sensors connected to the Espressif ESP32 development kit.

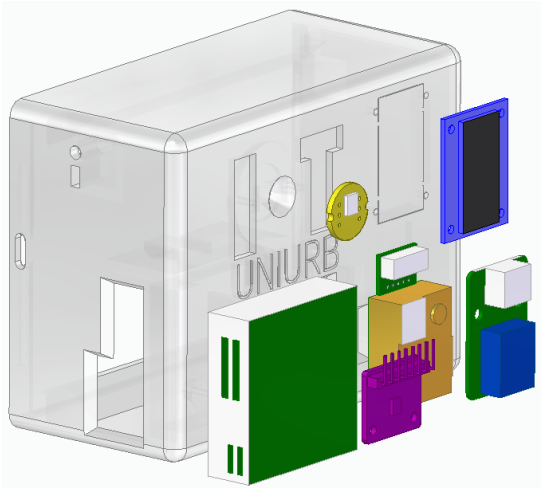


Fig. 2. 3D model view of the custom-made sensors enclosure box along with a 3D representation of the sensors installed in the device.

A. Main Processor

The main processor used within the IoT sensor is the ESP32. The ESP32 is a MicroController Unit (MCU) developed by Espressif Systems and renowned for its wide range of applications and its versatility in the field of IoT and embedded projects requiring wireless connectivity. Our application used the ESP32-wroom-32 DevKit variant, equipped with Wi-Fi and Bluetooth connectivity, an Xtensa dual-core processor, 400 KB of RAM, and 4 MB of flash memory [34]. Applications for the ESP32 are typically written using the Espressif IoT Development Framework (ESP-IDF) using the C or C++ language [35]. This framework offers a series of libraries that allow for the management of every aspect of the device, such as communication with the integrated GPIO pins, connecting to the Wi-Fi network, and enabling deep-sleep mode.

B. Sensors

We installed a series of sensors inside the device that can measure different physical parameters of the environment. The $\mathbf{CO_2}$ sensor measures the concentration of carbon dioxide using an infrared system. This sensor measures infrared light passing through a gas and absorbed proportionally to the concentration of CO_2 present in the air. In particular, this sensor is able to measure CO_2 values in a range from 0 to 5000 parts per million (ppm) and communicates with the MCU through UART and PWM protocols. The dust sensor measures the concentration of particles suspended in the air using laser optical technology. This sensor can detect particles of variable sizes, from $PM_{1.0}$ to $PM_{10.0}$ with great precision even in very low concentrations. The sensor is equipped with UART and I2C digital interface. The air quality sensor uses an array of chemical sensors that detect different gases to measure Volatile Organic Compounds (VOCs) and equivalent CO_2 (eCO_2) with high accuracy. The temperature and humidity sensor is capable of measuring temperatures in a range from 0 $^{\circ}C$ to 50 $^{\circ}C$ and humidity in a range from 20 %RH to 90 %RH at a frequency of 1 HZ and using a single wire communication protocol that makes integration with microcontrollers easy. Lastly, the noise sensor uses a microphone to sample the sounds around it and monitor the level of noise pollution in the environment. The microphone is capable of sampling sound at a frequency from 20 Hz to 20 kHz, characteristic of human hearing, reducing interference and noise to a minimum. The sensor uses the I2C protocol to transmit audio data to the microcontroller while consuming a few mW of power.

In addition to the sensors presented above, the platform also exports several communication buses of the MCU to allow connecting other external digital sensors. Moreover, the platform also holds a red-colored LED and a 0.96” RGB display. Their purpose is to show visual feedback to users and maintainers; for example, the LED can flash with a precise pattern to signal errors during operation. The display, conversely, can be used to periodically show text information related to the current air quality.

C. Software Stack

The sensor is designed to be a reliable and fully configurable platform while maintaining low power consumption, fast response time, and future expandability. The firmware was developed using the ESP-IDF framework based on the FreeRTOS [36] operating system and written entirely using C++. Fig. 3 shows a diagram of the main execution flows characterizing the software stack. In particular, the software is composed of three tasks: (i) measurement task, (ii) Telnet task, and (iii) OTA task, represented in the figure by three self-loops, each of which can execute in parallel using the two available cores and the internal FreeRTOS scheduler.

The first task we will talk about is the Telnet task; as the name implies, it is responsible for managing the Telnet service and offers remote access and configuration to the device. This service waits for new commands on port 23 and, as soon as a new one arrives, it executes it. The interface provided by the Telnet system is very similar to a common desktop Command Line Interface (CLI). Listing 1 shows the code for a structure representing a basic Telnet command. The `name` is a string

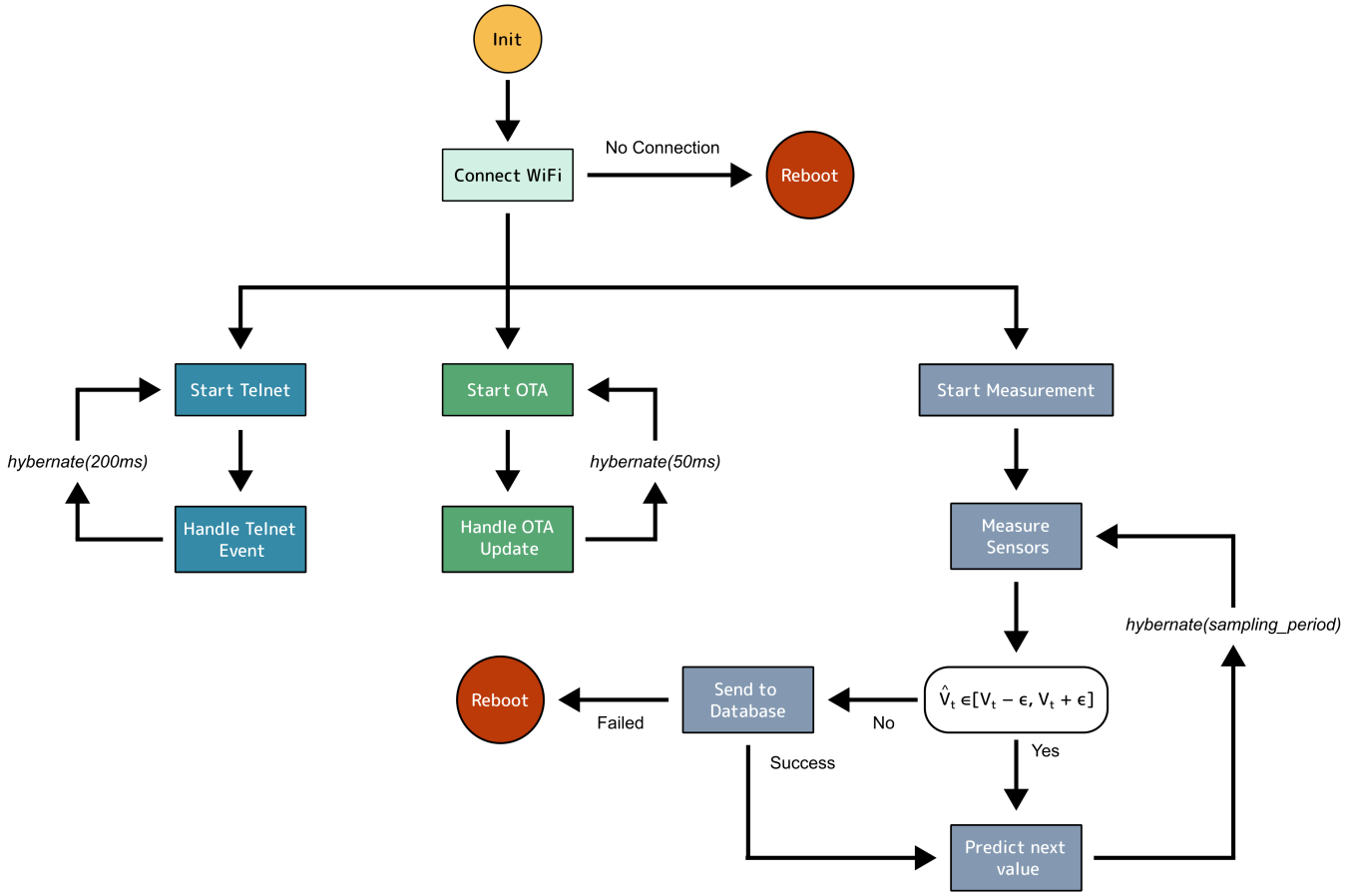


Fig. 3. The main execution flows of the software stack of the IoT sensor.

that uniquely identifies the command, while `run` is a pointer to a function executed when the command is called; finally, `help` is a human-readable string used by the *help* command describing the usage of all available commands.

```
typedef void (*cmd_on_run)(String cmd);

struct TelnetCommand {
  String     name;
  cmd_on_run run;
  String     help;
};
```

Listing 1: Code for a structure representing a generic telnet command.

In the code, a big static list is instantiated containing all the commands available. At runtime, the Telnet service parses the new events searching for one with a `name` that matches and executes its `run` function, otherwise, it returns a *"command not found"* error to the user. Several commands are available through Telnet allowing the configuration of critical device parameters, such as the device name, the calibration range for each sensor, the time interval between measurements, and even which sensors are enabled within the system. To ensure these parameters persist after the device is powered off, they are stored in the ESP32's internal flash memory and are accessed at every start.

The OTA task, on the other hand, implements an Over-The-Air (OTA) update service, enhancing the convenience of managing firmware updates. This service enables new firmware versions to be uploaded remotely via a Wi-Fi connection, eliminating the need to disassemble the device for manual flashing of the MCU. The OTA service continuously listens for updates on port 3232. When a new firmware version is detected, it initiates the update process by downloading the firmware and verifying its integrity and security. To facilitate this process, the ESP32's memory is divided into two partitions. One partition is dedicated to the boot process, while the other stores the new firmware. At the end of the update procedure, the system switches an internal flag to indicate which partition should be used for booting. The device then reboots, starting the new firmware, ensuring a seamless and efficient update process without physical intervention.

Lastly, the measurement task serves as the core function of the system. Its primary role is to periodically gather data from the various sensors installed on the device, process this information, and transmit it to the remote server for further analysis. Other than simply collecting and sending data, this task also implements the PBDC strategy.

Fig. 3 highlights several watchdog points, labeled as *Reboot*. To ensure a high level of reliability, these watchdog points will trigger a software reset if a failure occurs in the remote connection or during data transmission to the cloud. To prevent continuous resets in the event of persistent issues, a significant delay has been incorporated into the reboot routine. The software flow begins with a boot sequence that, after basic initialization, attempts to establish a remote Wi-Fi connection. If this attempt fails, the system schedules a delayed reboot to retry the process after some time. Conversely, if a stable connection is successfully established, the three main tasks are initiated. Another condition that triggers a reboot is the inability to transmit new data to the remote database. Since the primary function of this device is to collect sensor data, failing to transmit this data undermines its purpose. In such cases, it is preferable to reboot the device and restart with a clean execution.

The system makes use of InfluxDB [37] as its cloud server. Developed by InfluxData, InfluxDB is one of the most widely adopted Time Series Databases. This platform enables the efficient storage, retrieval, and analysis of collected data. Additionally, it supports the extraction of meaningful statistics and other key characteristics from the data.

D. Prediction-Based Data Collection

This section presents a detailed analysis of the PBDC strategy aimed at achieving an optimal trade-off between device performance and energy efficiency.

The strategy is set in a context in which the system is composed of several IoT nodes that periodically measure values from their sensors and send them to a central server that acts as a data sink. Additionally, we assume that the application running at the sink can tolerate a slight margin of error in the accuracy of the reported data. Unlike the ideal scenario where the sink periodically receives *exact* values in *all* packets, in this context, deviations from the exact values are acceptable, as long as their extent in terms of difference in value and time interval during which the deviation occurs are small enough.

The system's innovation lies in the ability to determine when a piece of data should be sent to the server using a machine learning-based forecasting model deployed simultaneously both in the node devices and in the central server.

To simplify the explanation, let's take as an example the case where there is only one central server and a single node with one CO_2 sensor that is read every N minutes. Let V_t be the CO_2 value read by the node at time t and let $\hat{V}_t$ be the value estimated by the regression model for the same time t. The tolerance value is defined as

$$\epsilon = max(\epsilon_{abs}, \frac{V_t}{100}\epsilon_{rel}) \quad (1)$$

where ϵ_{abs} is the absolute error and ϵ_{rel} is the relative error. The estimated value $\hat{V}_t$ is an acceptable approximation of V_t if it remains in the range defined by the tolerance value, $\hat{V}_t \in [V_t - \epsilon, V_t + \epsilon]$.

The combined use of ϵ_{abs} and ϵ_{rel} in the tolerance value is intended to further reduce inaccuracies. Using only the absolute error would risk considering all those minimally perceptible variations. Instead, by combining relative and absolute errors, the algorithm can adapt to all changes and avoid unnecessary communications with the remote server.

Algorithm 1 provides a detailed pseudo-code of the PBDC algorithm. The algorithm's internal state consists of the predicted value, $\hat{V}_t$, at time t and a window buffer with fixed-size ws. During each iteration, the current sensor value is read and stored into the variable V_t, and ϵ is calculated using Eq. 1. If $\hat{V}_t$ falls outside the tolerance range, V_t is transmitted to the server. Regardless of whether the value is sent, the circular buffer is shifted by one position, removing the oldest value and adding the current one. This updated buffer is then used by the regression model to predict the next value. The device then enters hibernation, awaiting the next iteration.

Algorithm 1 The pseudo-code of the PBDC algorithm.

State: buffer ▷ Circular buffer of size ws
State: $\hat{V}_t$ ▷ Predicted value at time t
Require: ϵ_{rel} ▷ Relative error
Require: ϵ_{abs} ▷ Absolute error

```
1:  while true do
2:      V_t ← measureSensor()
3:      ε ← max(ε_abs, (V_t/100) ε_rel)
4:      if V̂_t ∈ [V_t − ε, V_t + ε] then
5:          popFirst(buffer)
6:          appendLast(buffer, V̂_t)
7:          // No need to send
8:      else
9:          popFirst(buffer)
10:         appendLast(buffer, V_t)
11:         sendToServer(V_t)
12:     end if
13:     V̂_t ← predict(buffer)
14:     hybernate(sampling_period)
15: end while
```

Careful attention must be given to the buffer update logic and its role in the regression model. The regression model is deployed simultaneously on both the node and the remote server. To ensure synchronization between the two, predictions must be based on the same data in both environments. Therefore, when the node updates the buffer, it must ensure the server also has access to the same data. If the value V_t is outside the tolerance range (i.e. it has been sent to the server), its value is added to the buffer. However, if V_t is within the tolerance range (i.e. it has not been sent), the predicted value is used instead. This approach guarantees that the sensor will predict future values starting from the same history values held by the server.

In this work, we experimented with different configurations of the deep learning forecasting model. Table I shows the details of the three models, namely: *Model 1*, *Model 2*, and *Model 3*. Model 1 consists of a network with a single LSTM node with 10 internal units followed by a single Dense layer. Model 2 is a hybrid model, consisting of two CNN layers, both with 64 filters, followed by one LSTM node with 10 units and a Dense layer. Finally, Model 3 consists of two LSTM nodes both with 10 units followed by a Dense layer. As seen from their structures all the models are relatively small in size, in

fact, Model 2, which is the largest, has only 14,019 parameters and a weight of 54.76 KB.

For training and testing the models, we used a dataset consisting of data collected from four different types of sensors. In particular CO_2, noise, $PM_{2.5}$, and radioactivity. These data were collected using a version of the proposed IoT device with the PBDC strategy turned off. In this device version, the sensors are read once every 10 minutes, and the real data are always sent to the central server without predicting the next values. For each sensor type, we divided the time series into training, testing, and validation sets. We also used a different set of time series to measure accurately the global performances of the PBDC algorithm on previously unseen data. The forecasting models were trained using a supervised learning approach. During training, we iterated over the dataset to create sliding windows of length ws, where each window was composed of consecutive data points serving as the history input. The value immediately following the window served as the target for prediction. This sliding window technique effectively transforms the time series into a set of input-output pairs, allowing the model to capture temporal dependencies and trends.

TABLE I. DEEP-LEARNING MODELS ADOPTED TO IMPLEMENT THE PBDC STRATEGY

Net ID	Net Type	Net Structure	Parameters	Net Size
Model 1	1LSTM_1D	10U_1D	491	1.92 KB
Model 2	2CNN_1LSTM_1D	64F_64F_10U_1D	14,019	54.76 KB
Model 3	2LSTM_2D	10U_10U_30D_1D	1,681	6.57 KB

IV. CASE STUDY AND RESULTS

As a case study, we deployed several IoT sensor prototypes in a two-centuries building (Collegio Raffaello) used as a university campus in the city of Urbino - Italy. In particular, the second floor of the building currently houses the degree program in Computer Science and the degree program in Foreign Languages and Cultures of the University of Urbino.

The objective of this deployment is to monitor physical parameters within the university classroom which are frequently occupied by several hundred students attending their daily lectures. Precisely, various environmental parameters, including temperature, humidity, noise level, CO_2 concentration, and particulate matter are continuously measured and stored on a centralized server for data analysis.

In this section, we provide results from two sets of experiments. First of all, we present some results related to the monitoring of the environmental parameters during daily activities, and then we focus on the performance of the PBDC capability implemented on the sensors.

A. Monitoring of the Daily Activities

In this section, we report a representative scenario of the use of the proposed IoT sensors, showing that our solution performs efficiently while used to monitor the environmental parameters in indoor buildings.

The testbed, was deployed in the Collegio Raffaello at the end of 2021 and, in the following years, it collected more than 15,000 measures a day. Each day at 7:30 a.m., the entrances to the second floor of building are opened by a guardian, allowing access to students and faculty. University lessons are traditionally structured into two-hour blocks, with the first session beginning at 9:00 a.m. and concluding at 11:00 a.m. A second session follows, ending with a lunch break at 1:00 p.m. Additional sessions may commence in the afternoon, starting at 2:00 p.m. Consequently, the most significant influx of individuals is expected between approximately 8:30 a.m. and 9:00 a.m., as well as between 1:30 p.m. and 2:00 p.m.

As a representative example, we report in Fig. 4 the noise level, the CO_2, and the PM_{10} concentrations measured during a whole day in a classroom. In all three cases, the value of the signals rapidly grows after 9.00 a.m. at the start of lessons. In particular, the noise level instantly reflects the room occupancy. At the same time, both CO_2 and PM_{10} concentrations show an evident inertia that determines a delayed response with respect to the arrival of people. This is undoubtedly due to the fact that the people in the room are sources of both particulate matter and carbon dioxide (as a result of breathing), which accumulate slowly after their arrival. In fact, the Collegio Raffaello does not have an integrated air change system which is achieved by manually opening the doors or the windows.

As expected, the qualitative correlation between these three variables is very strong as all of them can be considered as the results of human activity. It is also interesting to note the delay of about two hours, with respect to the start of the lesson, with which the CO_2 and PM_{10} concentrations reach the maximum value.

B. PBDC Performances

To assess the performance of the PBDC implemented on the IoT sensor, we first evaluated the prediction capability of the proposed deep-learning models when forecasting the following environmental parameters: CO_2, noise level, PM_{10}, and radioactivity. The last parameter was collected by externally connecting a Geiger counter to the digital bus of the sensor platform.

Fig. 5 reports the models' forecasting performance in terms of MAE for the three models considered when varying the size of the input window (ws). Concerning CO_2 data (Fig. 5a), models 1 and 3 appear to demonstrate superior performance (lower MAE) compared to model 2. Moreover, we must highlight that the forecasting performance of all three models is highly commendable, as evidenced by the consistently low MAE values below 34 ppm, while the measurement range traditionally spans from 450 to 2500 ppm. On the other hand, the forecasting performance does not appear to be significantly affected by the length of the input window, remaining almost constant.

These same observations can be associated with the results of the noise (Fig. 5b) and particulate matter (Fig. 5c) while the case of radioactivity appears to be more complex (Fig. 5d). Here, Model 3 shows a sensible performance reduction for ws corresponding to 9 and 11 samples while the others remain stable. The remaining experiments reported in this work were conducted on the CO_2 dataset using a window size ws of 5 samples as it represents a good performance point for each model.

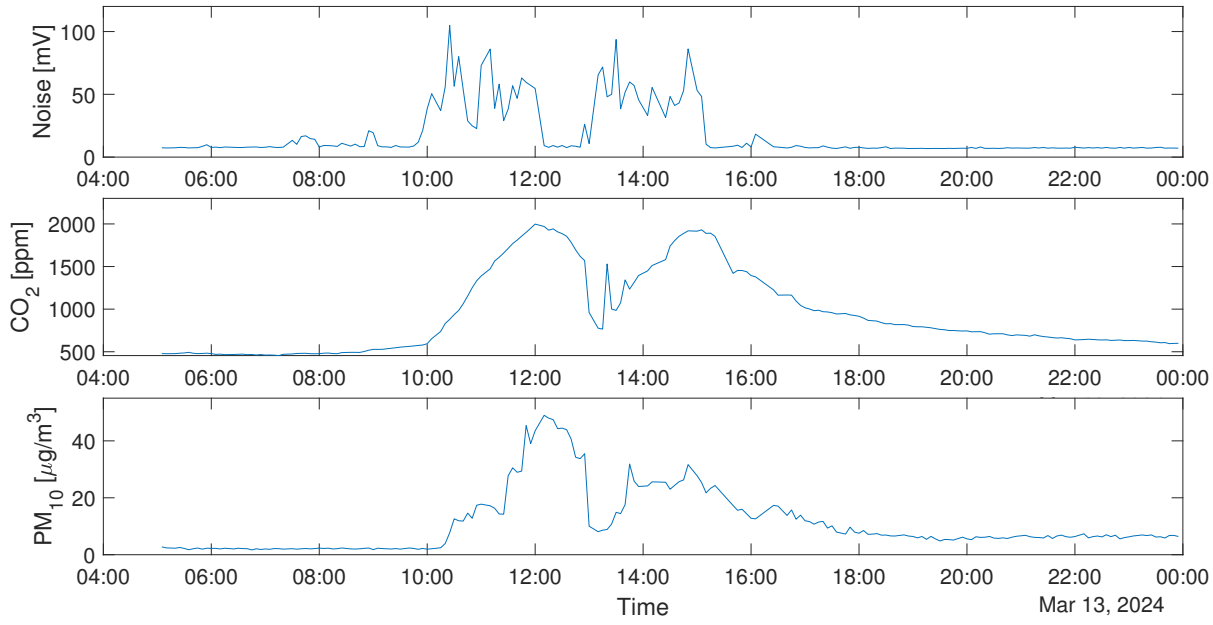
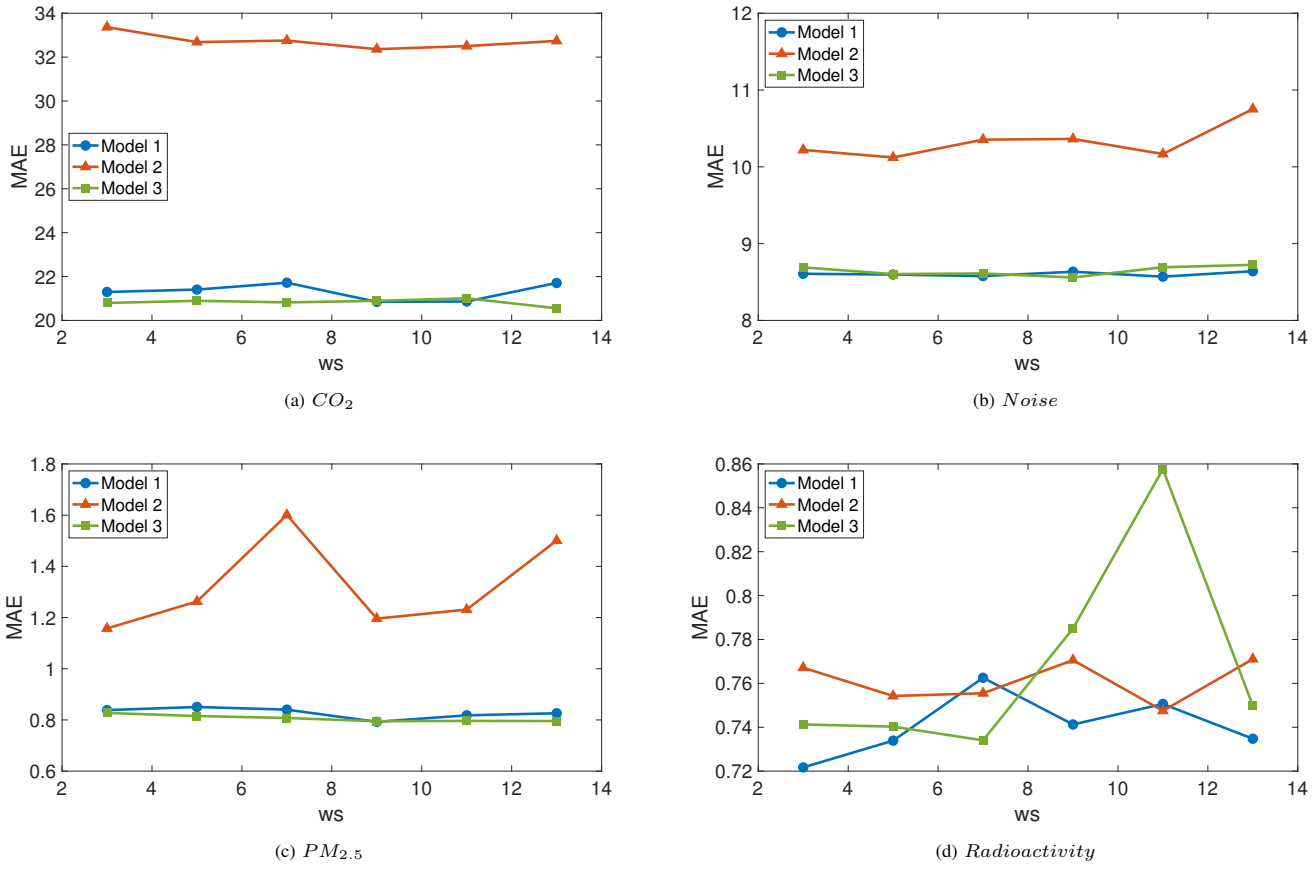


Fig. 4. Environmental parameters (noise level, CO_2, and PM_{10} concentrations) measured during a single day in a university classroom.



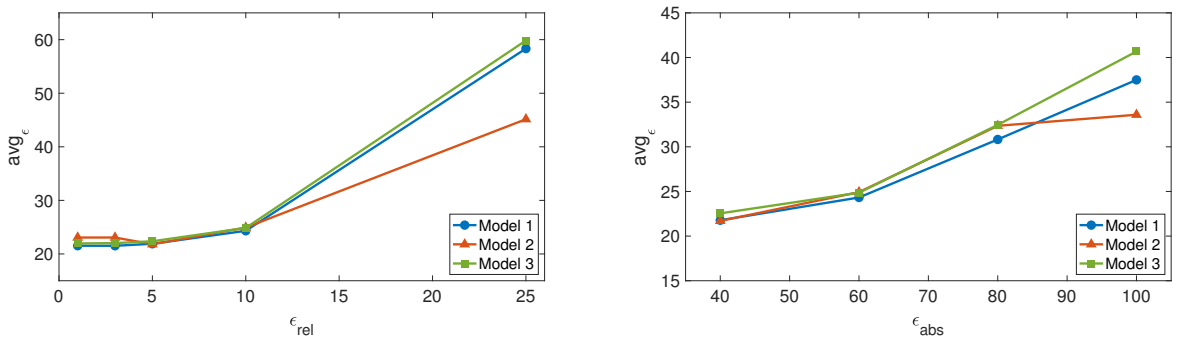
(a) CO_2

(b) $Noise$

(c) $PM_{2.5}$

(d) $Radioactivity$

Fig. 5. Forecasting performance in terms of MAE for the three models considered in this study when varying the size of the input window (ws). The four plots represent performance in forecasting, respectively, CO_2, noise, $PM_{2.5}$, and radioactivity.



(a) avg_ϵ Vs ϵ_{rel} (b) avg_ϵ Vs ϵ_{abs}

Fig. 6. Average absolute error (avg_ϵ) introduced by the PBDC strategy when varying both the relative error (ϵ_{rel}) and the absolute error (ϵ_{abs}) parameters.

Fig. 6 shows the average absolute error (avg_ϵ) introduced by the PBDC strategy when varying both the ϵ_{rel} and the ϵ_{abs} parameters. Notice that, this kind of error represents the average uncertainty perceived by the collection server due to the application of the strategy. The two plots show that, in both cases, increasing the acceptable error magnitude increases the average error of the collected data. It is noteworthy that the growth observed in the case of the ϵ_{rel} parameter appears to be more than linear. In contrast, the analogous behavior in the case of the ϵ_{abs} parameter is relatively less pronounced.

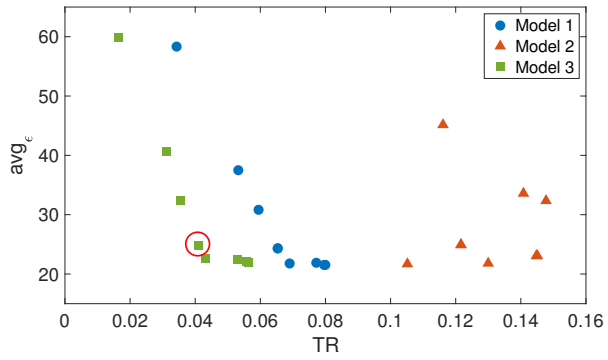


Fig. 7. Pareto chart reporting the average absolute error (avg_ϵ) vs. The transmission ratio (TR) achieved by different models configurations. The red circle identifies the point that minimizes both metrics.

The counterpart of increasing the acceptable error magnitude is the reduction of the number of transmitted packets with a consequent reduction in energy and network congestion. Obviously, this reduction also depends on the prediction capability of the forecasting model, as the higher the accuracy of the prediction, the lower the number of packets that are required to be transmitted.

In order to characterize the tradeoff between the error introduced by the PBDC strategy and the number of saved packets, for each previously reported experiment, we calculated the transmission ratio (TR) as the ratio between the number of packets sent with and without the PBDC. Fig. 7 plots a Pareto chart where each configuration of the PBDC strategy is represented by a single point. Different colors identify the three forecasting models under consideration, while the coordinates of the points are represented by the corresponding avg_ϵ and TR. Each configuration differs in terms of ws, ϵ_{rel}, and ϵ_{abs}. The point highlighted with a red circle identifies the configuration which minimizes the average error and the transmission rate at the same time. In particular, the selected point determines a TR of about 4%, which corresponds to a reduction in transmitted packets of approximately 96% while introducing an average error of about 25 ppm in the CO_2 measurement. This means that if the design of such an IoT application can tolerate an average error of that magnitude, the energy saved in packet transmission can reach up to 96%. For completeness, the corresponding parameter values were: $ws = 5$, $\epsilon_{rel} = 10$, and $\epsilon_{abs} = 60$.

V. Conclusion

In indoor monitoring, IoT systems collect real-time data to improve comfort, health, and energy efficiency. However, traditional systems consume excessive power due to continuous data collection. Machine learning allows IoT devices to predict environmental changes and adapt data collection, optimizing energy use. This shift to Prediction-Based Data Collection improves IoT sustainability and effectiveness. In this paper, we presented a low-cost IoT sensor framework that makes use of deep-learning tools to forecast measured data and change the sampling rate accordingly in order to reduce power consumption and data transmission. Experiments show the system can save up to 96% of energy while maintaining data quality.

One key limitation of our approach is that, at each iteration, the edge system still needs to perform sensor readings to determine whether the data needs to be transmitted to the server. This requirement of continuous local sampling limits the potential energy savings, as sensor readings still consume power even if the data is not ultimately sent to the server. A possible future improvement could involve developing a method to completely skip certain measurements on the edge device itself, relying solely on predicted values when conditions are stable. This approach would further reduce energy consumption by allowing the device to enter deep sleep mode, eliminating unnecessary sensor activations, and maximizing efficiency.

REFERENCES

[1] E. Lattanzi, M. Dromedari, and V. Freschi, "A scalable multitasking wireless sensor network testbed for monitoring indoor human comfort," *IEEE Access*, vol. 6, pp. 17 952–17 967, 2018.

[2] H. K. Apat, R. Nayak, and B. Sahoo, "A comprehensive review on internet of things application placement in fog computing environment," *Internet of Things*, p. 100866, 2023.

[3] S. C. Mukhopadhyay, S. K. S. Tyagi, N. K. Suryadevara, V. Piuri, F. Scotti, and S. Zeadally, "Artificial intelligence-based sensors for next generation iot applications: A review," *IEEE Sensors Journal*, vol. 21, no. 22, pp. 24 920–24 932, 2021.

[4] A. Bogliolo, V. Freschi, E. Lattanzi, A. L. Murphy, and 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)*. IEEE, 2014, pp. 21–28.

[5] J. Saini, M. Dutta, and G. Marques, "Indoor air quality monitoring systems based on internet of things: A systematic review," *International journal of environmental research and public health*, vol. 17, no. 14, p. 4942, 2020.

[6] L. Bianconi, Y. Lechiara, L. Bixio, R. Palermo, S. Pensieri, F. Viti, and R. Bozzano, "Edge and fog computing for iot: A case study for citizen well-being," in *International Summit Smart City 360°*. Springer, 2021, pp. 121–139.

[7] T. L. Narayana, C. Venkatesh, A. Kiran, A. Kumar, S. B. Khan, A. Almusharraf, M. T. Quasim *et al.*, "Advances in real time smart monitoring of environmental parameters using iot and sensors," *Heliyon*, vol. 10, no. 7, 2024.

[8] Z. Liu, G. Wang, L. Zhao, and G. Yang, "Multi-points indoor air quality monitoring based on internet of things," *IEEE access*, vol. 9, pp. 70 479–70 492, 2021.

[9] S. Sonawani and K. Patil, "Air quality measurement, prediction and warning using transfer learning based iot system for ambient assisted living," *International Journal of Pervasive Computing and Communications*, vol. 20, no. 1, pp. 38–55, 2024.

[10] W.-T. Sung and S.-J. Hsiao, "Building an indoor air quality monitoring system based on the architecture of the internet of things," *EURASIP Journal on Wireless Communications and Networking*, vol. 2021, pp. 1–41, 2021.

[11] Y. Zhu, S. A. Al-Ahmed, M. Z. Shakir, and J. I. Olszewska, "Lstm-based iot-enabled co2 steady-state forecasting for indoor air quality monitoring," *Electronics*, vol. 12, no. 1, p. 107, 2022.

[12] A. Bogliolo, E. Lattanzi, and V. Freschi, "Idleness as a resource in energy-neutral wsns," in *Proceedings of the 1st International Workshop on Energy Neutral Sensing Systems*, 2013, pp. 1–6.

[13] A. S. Shah, H. Nasir, M. Fayaz, A. Lajis, and A. Shah, "A review on energy consumption optimization techniques in iot based smart building environments," *Information*, vol. 10, no. 3, p. 108, 2019.

[14] U. Raza, A. Bogliolo, V. Freschi, E. Lattanzi, and A. L. Murphy, "A two-prong approach to energy-efficient wsns: Wake-up receivers plus dedicated, model-based sensing," *Ad Hoc Networks*, vol. 45, pp. 1–12, 2016.

[15] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella, "Energy conservation in wireless sensor networks: A survey," *Ad hoc networks*, vol. 7, no. 3, pp. 537–568, 2009.

[16] U. Raza, A. Camerra, A. L. Murphy, T. Palpanas, and 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*. IEEE, 2012, pp. 85–94.

[17] E. I. Gaura, J. Brusey, M. Allen, R. Wilkins, D. Goldsmith, and R. Rednic, "Edge mining the internet of things," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3816–3825, 2013.

[18] F. A. Aderohunmu, G. Paci, D. Brunelli, J. D. Deng, L. Benini, and M. Purvis, "An application-specific forecasting algorithm for extending wsn lifetime," in *2013 IEEE international conference on distributed computing in sensor systems*. IEEE, 2013, pp. 374–381.

[19] H. Harb, C. A. Jaoude, and A. Makhoul, "An energy-efficient data prediction and processing approach for the internet of things and sensing based applications," *Peer-to-Peer Networking and Applications*, vol. 13, no. 3, pp. 780–795, 2020.

[20] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong, "Model-driven data acquisition in sensor networks," in *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, 2004, pp. 588–599.

[21] D. Chu, A. Deshpande, J. M. Hellerstein, and W. Hong, "Approximate data collection in sensor networks using probabilistic models," in *22nd International Conference on Data Engineering (ICDE'06)*. IEEE, 2006, pp. 48–48.

[22] D. Tulone and S. Madden, "Paq: Time series forecasting for approximate query answering in sensor networks," in *European Workshop on Wireless Sensor Networks*. Springer, 2006, pp. 21–37.

[23] ——, "An energy-efficient querying framework in sensor networks for detecting node similarities," in *Proceedings of the 9th ACM international symposium on Modeling analysis and simulation of wireless and mobile systems*, 2006, pp. 191–300.

[24] A. Jain, E. Y. Chang, and 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.

[25] J.-F. Yang, Y.-J. Zhai, D.-P. Xu, and P. Han, "Smo algorithm applied in time series model building and forecast," in *2007 International Conference on Machine Learning and Cybernetics*, vol. 4. IEEE, 2007, pp. 2395–2400.

[26] A. Lahouar and J. B. H. Slama, "Day-ahead load forecast using random forest and expert input selection," *Energy Conversion and Management*, vol. 103, pp. 1040–1051, 2015.

[27] J. Han, G. H. Lee, S. Park, J. Lee, and J. K. Choi, "A multivariate-time-series-prediction-based adaptive data transmission period control algorithm for iot networks," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 419–436, 2021.

[28] C. Cao, J. Huang, M. Wu, Z. Lin, and Y. Sun, "A multivariate time series prediction method based on convolution-residual gated recurrent neural network and double-layer attention," *Electronics*, vol. 13, no. 14, p. 2834, 2024.

[29] İ. Kök and S. Özdemir, "Deepmdp: A novel deep-learning-based missing data prediction protocol for iot," *IEEE Internet of Things Journal*, vol. 8, no. 1, pp. 232–243, 2020.

[30] J. He, Y. Li, X. Zhang, and J. Li, "Missing and corrupted data recovery in wireless sensor networks based on weighted robust principal component analysis," *Sensors*, vol. 22, no. 5, p. 1992, 2022.

[31] N. A. Khan and S. Ali, "Robust sparse reconstruction of signals with gapped missing samples from multi-sensor recordings," *Digital Signal Processing*, vol. 123, p. 103392, 2022.

[32] N. Fatima, S. Riaz, S. Ali, R. Khan, M. Ullah, and D. Kwak, "Sensors faults classification and faulty signals reconstruction using deep learning," *IEEE Access*, 2024.

[33] W. Lalouani, M. Younis, I. White-Gittens, R. N. Emokpae Jr, and L. E. Emokpae, "Energy-efficient collection of wearable sensor data through predictive sampling," *Smart Health*, vol. 21, p. 100208, 2021.

[34] Espressif, "Esp32-c3-wroom-02 datasheet," 2024. [Online]. Available: https://www.espressif.com/en/support/documents/technical-documents

[35] ESP, "Esp iot development framework — espressif systems," 2024. [Online]. Available: https://www.espressif.com/en/products/sdks/esp-idf

[36] FreeRTOS, "Freertos — real-time operating system for microcontrollers," 2024. [Online]. Available: https://www.freertos.org/

[37] InfluxData, "Influxdb — time series data for every workload." 2024. [Online]. Available: https://www.influxdata.com