Enhancing Plant Growth Through IoT-enabled Daily Light Tracking

Lorenzo Grandi
Università di Bologna
Bologna, Italia
Email: lorenzo.grandi7@studio.unibo.it

Abstract—**Proper light exposure is critical for plant health, yet determining the optimal placement within an indoor environment remains challenging. Existing methods often rely on trial and error, leading to suboptimal conditions and plant mortality. This project addresses these challenges by leveraging IoT technology to enhance plant growth and vitality through precise light monitoring. The project aims to develop an IoT system utilizing luminosity sensors to monitor and analyze daily light exposure for optimizing the placement of plant pots. By collecting and analyzing light data from multiple positions, the system seeks to recommend optimal locations for plants based on their specific light requirements.**

I. Introduction

In recent years, the beneficial impact of plants on human health and wellness has become increasingly well-recognized, backed by a growing body of scientific research[1]. Beyond their aesthetic appeal, plants contribute significantly to physical health, primarily by enhancing indoor air quality. In urban areas especially, people are often exposed to various pollutants and volatile organic compounds (VOCs), which are emitted by everyday items like furniture, cleaning products, paints, and electronics. VOCs can lead to respiratory problems, allergies, and even long-term health complications when present in high concentrations[2][3]. By improving air quality, plants can alleviate respiratory issues and reduce the risk of other health problems associated with prolonged exposure to indoor pollution, creating a more breathable, less toxic environment for individuals.

Houseplants often struggle to thrive, and many fail to last long primarily due to common mistakes rooted in a lack of botanical knowledge. One of the most frequent issues is insufficient light exposure, as indoor plants are often placed in areas that don't receive adequate sunlight. This lack of natural light directly impacts the plant's health, inhibiting its growth and longevity. To address this problem, our project proposes the development and implementation of a scalable, wireless sensor network designed to monitor and analyze indoor spaces, identifying locations with the best light exposure throughout the day.

By collecting real-time data on light levels across different areas, the sensor system can identify ideal spots for houseplants, maximizing their chances of receiving sufficient sunlight. The technology is versatile and adaptable, allowing users to adjust placement as seasonal light changes or as the layout of the space evolves. Ultimately, this solution offers a sustainable method to enhance the environment for houseplants, introducing a model-predictive control able to alert what will be the best spot for the future hours. The key aspects of the application are ease of use, reduced costs and system scalability.

The rest of the article is organized as follows. Section II briefly introduce the used technologies, discussing the node compositon, features and communication mechanisms. The methodology used to setup the network and the computed performance metrics are explained in Section III. Section IV presents and analyzes the results of the system. Section V concludes.

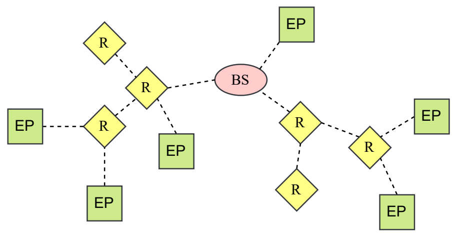


Fig. 1. WSN architecture

II. Technology Background

Wireless Sensor Networks (WSNs) mark a transformative approach in sensing, monitoring, and data collection technologies. These networks are composed of spatially distributed, autonomous sensors that monitor various physical or environmental conditions, such as temperature, sound, pollution levels, humidity, and motion, and cooperatively transmit their data across the network to a central hub, as illustrated in Fig. 1. WSNs are primarily designed to provide robust, reliable, and scalable systems for continuous monitoring of critical parameters over large geographic areas, including those that may be inaccessible to humans. With unique capabilities like real-time data provision, distributed operation, and adaptability to changing conditions, WSNs have become essential in modern technology. Moreover, as the number of sensors increases, so does the accuracy of the generated measurement map, particularly in indoor environments, where overall noise levels are typically higher than in outdoor settings.

A. WSN Architecture

The core component of a WSN is the sensor node, typically a microcontroller unit (MCU) such as an Arduino or ESP32. Equipped with physical sensors, the MCU can collect data and transmit it to a base station. Designed for battery-powered operation, these boards are highly efficient, low-cost, and lightweight, making them preferable to more complex, high-power systems like processors, or high cost ones like FPGAs. In addition to sensing, these nodes can also function as actuators, enabling full automation of tasks and supporting programmable GPIO protocols like SPI, I2C, UART, and PWM. To facilitate Cloud connectivity, these devices include various communication protocols operating in the data link layer, such as Wi-Fi, Bluetooth Low Energy (BLE), and 802.15.4, capabilities provided by the ESP32-C6 System on Chip (SoC), the primary board used in this work. To extend battery life, such systems often employ sleep mode, a technique that significantly reduces power consumption while the system awaits new stimuli. This approach allows the end device's battery to last considerably longer than it would in normal operating mode.

These data link protocols enable all sensed data to be stored in the cloud for offline processing, allowing the system to perform high-power computations outside the sensor node environment. This approach achieves greater accuracy and precision without impacting the battery life of the end devices. Before reaching the cloud, data is first directed to the base station, nodes tasked with gathering data from end devices and forwarding it to the cloud. Given that physical distance can limit connectivity, routers are integrated into the WSN, adding intermediary nodes to bridge the gap between sensor nodes and the base station.

In this architecture, data from end devices is not only reliably collected and transmitted to the cloud, minimizing errors from physical constraints like electromagnetic interference (EMI), but also enables potentially unlimited scalability. This is possible because the workload is supported by a system capable of highly parallelizing transmission paths, with the communication link between base stations and the cloud serving as the system's primary bottleneck.

B. Application Layer Protocols

Above the data link layer, endpoint applications rely on the application layer, utilizing protocols like User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) along with specific protocols such as Constrained Application Protocol (CoAP), MQ Telemetry Transport (MQTT), and Hypertext Transfer Protocol (HTTP). These protocols facilitate message transfer between endpoints and the base station, using network paradigms that streamline communications, allowing various systems to connect reliably with each endpoint.

The primary distinction between these protocols lies in how they manage links, often structuring communication around publish-subscribe messaging models. Each message carries all relevant information, ensuring that receivers have everything

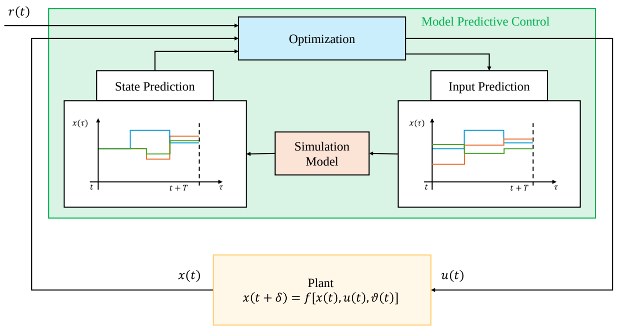


Fig. 2. MPC structure

necessary for processing, minimizing errors and enhancing reliability.

C. Control Models

As mentioned earlier, WSNs function not only as frameworks for data sensing, transmission, and storage but also as actuators. Over the years, numerous control algorithms have been developed, ranging from traditional approaches like Proportional-Integral-Derivative (PID) control to more advanced methods, such as Model Predictive Control (MPC), which incorporate predictive models[4][5]. A key feature of these algorithms is their ability to integrate weighted models, including deep neural networks (DNNs) or, more broadly, machine learning (ML) applications. This integration enables control models to respond to the environment based on predictions rather than solely on immediate sensor data, enhancing adaptability and precision. Recently, new online-training models[6] have been incorporated into System-on-Chip (SoC) applications due to the increasing capabilities of these technologies. Advances include faster processing units that enable high-speed computation, along with AI-integrated hardware designed to accelerate the demanding computational tasks of machine learning algorithms.

The structure of the control model is depicted in Fig. 2. This figure illustrates how model predictive control generates the optimal control input, $u(t)$, based on the current state, $x(t)$, derived from the plant that represents the controlled system. Both the state and input predictors can utilize various types of predictive models, such as Machine Learning (ML) or Deep Learning algorithms.

In typical systems, these predictive models are often integrated directly into the system, such as within the MCU, to enhance control effectiveness. However, this integration poses significant optimization challenges due to the limited computational resources available. As a result, lightweight predictive models are generally favored, emphasizing the design principle of "less is more". Nevertheless, if application constraints allow it, predictive models can instead be offloaded to Cloud processing, enabling more complex computations, such as computer vision or Deep Learning tasks.

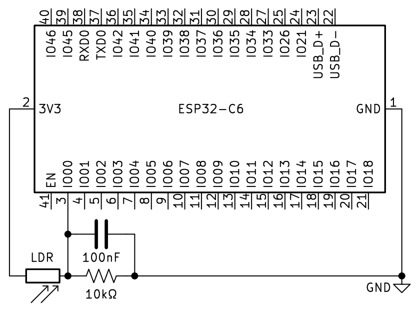


Fig. 3. Endpoint schematic

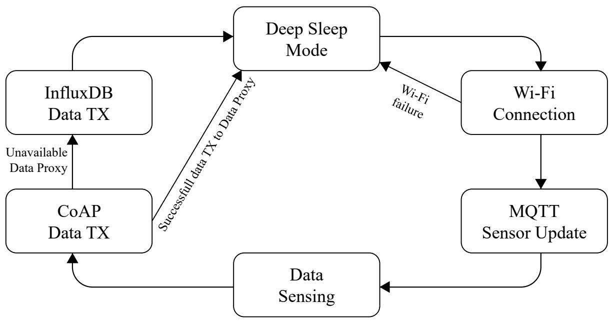


Fig. 4. Time multiplexing scheme of the MCU

III. Methodology

In this work, a scalable WSN using ESP32-C6 SoCs was implemented. Since the sensor count was relatively low and the distances were limited, no routers were needed, allowing sensors to communicate directly with the base station via the local Wi-Fi network. However, the WSN remains fully scalable, so additional nodes can be easily incorporated. Unlike larger WSNs, where a mesh topology is often preferred, a star network was chosen here, which is sufficient for the project's scope. If the number of end devices increases, a mesh topology should be considered, leveraging the BLE protocol capabilities in the ESP32-C6. This approach would enable a large, robust sensor network that benefits from the ESP32-C6's low-power technology. In such a case, however, EMIs from reflections in indoor environments should be considered, as it may become more significant as the number of sensors increases.[7].

A. WSN architecture

The implemented WSN comprises two ESP32-C6 devices serving as sensor nodes, each equipped with a Light Dependent Resistor (LDR) to measure light intensity. The LDRs are configured in a voltage divider circuit, as illustrated in Fig. 3, to appropriately set the voltage level at the MCU's input pin, which includes an integrated Analog-to-Digital Converter (ADC) peripheral. To minimize noise before ADC conversion, a 100 nF ceramic capacitor is placed between the input pin and ground[8]. This setup supports sampling at a resolution of 12 bits and a rate of 100 kSPS (kilo samples per second), far exceeding the requirements of the application. Given that typical variations in light intensity occur over several minutes, such a high sampling rate is unnecessary, although the user has the option to adjust the sampling period as needed.

In the MCU code, the signal captured by the ADC peripheral is normalized to a range between 0 and 1. This normalization allows users to easily interpret the measurements, especially since no direct physical unit is applicable. When incident light on an LDR (Light-Dependent Resistor) exceeds a certain threshold frequency, the photons absorbed by the semiconductor provide bound electrons with enough energy to transition to the conduction band. This process generates free electrons, which conduct electricity and reduce the resistance of the LDR. Since the resistance range and sensitivity of photoresistors can vary between devices, normalizing the sensed values ensures consistency and provides a clearer and more standardized representation for the user, resulting is a percentage scale on the dashboard.

The sensor nodes are designed to run on battery power, making the integration of low-power features, such as deep sleep modes, highly advantageous. In this mode, the endpoint remains in sleep state during periods when sensing is not required. This approach significantly extends battery life, as the sensor consumes only up to 10μA in deep sleep while maintaining memory retention and keeping the Real-Time Clock (RTC) active. For instance, with a battery supply of 38Ah, the sensor could operate for approximately 3,800,000 hours, equivalent to nearly 433 years.

However, this approach has the drawback of reduced responsiveness to user settings. Specifically, when the sensor is in deep sleep mode, configuration adjustments cannot be made until the sensor wakes up. Additionally, frequent wake-ups may cause connection issues, increasing the likelihood of failures when linking to the base stations. Any such failures were detected and resolved through redundancy, ensuring the system's reliability. In the time control flow shown in Fig. 4 is highlighted the redundancy applied to the system to cover possible failures of data transmission.

B. Data Proxy

The primary function of the WSN's devices is to act as data collectors, supplying input to the MPC algorithm. While the application could leverage the integrated ESP-DL library[9], doing so would require the endpoint to be equipped with dedicated memory capable of storing a sufficient amount of data. This is necessary to ensure that any locally implemented model can achieve performance comparable to cloud-based applications. To address these challenges, a cloud-based application was selected, with a data proxy serving as the link between the endpoints and the cloud.

The data proxy is implemented as a Python (3.12.4) script that functions as the base station for the WSN. It collects data from the endpoints over a Wi-Fi network using the CoAP protocol. The collected data are stored in an InfluxDB database, enabling efficient access and processing. This approach allows for the accumulation of large datasets without the need to design a dedicated storage system, instead leveraging an external platform that ensures both data security and privacy.

A dedicated Command-Line Interface (CLI) was developed to enable users to configure key parameters. The data proxy gathers these user-defined configurations and transmits them to the sensor nodes via the MQTT protocol. The proxy is designed to handle all these tasks concurrently, ensuring seamless operation and responsiveness.

For security and privacy, both CoAP and MQTT protocols are restricted to the local network, exploiting the home router and `mosquitto` security services, while data in InfluxDB is accessible from the remote cloud through the HTTP protocol, utilizing the service's security protocols for secure access.

To enhance scalability, the system is organized using classes, encapsulating all sensor node functionalities, such as the CoAP server, MQTT publisher, and InfluxDB publisher. Each sensor is represented as an object of the `LdrSensorManager` class, allowing efficient management of multiple sensors through a simple list of instances. Similarly, positions and plants are modeled as classes with dedicated features, including the creation of new positions or plants via the CLI.

All sensor device configurations are stored in JSON files to streamline system operations. When a command is issued through the CLI, it writes directly to the JSON files. A dedicated detector monitors these files, extracts the updated information, and synchronizes the sensors in real time. The new settings are sent to the MQTT broker, to which the endpoints are subscribed. This ensures that endpoints are automatically updated with the latest user-defined settings as soon as changes are detected.

C. Model Predictive Control

In this work, an MPC (Model Predictive Control) framework was designed as the control model for the system. Although the selected boards support ESP-DL, a library optimized for Machine Learning (ML) and Artificial Intelligence (AI) applications, the control model was implemented using Python scripts instead. This approach enables the integration of user-friendly forecasting models such as ARIMA (AutoRegressive Integrated Moving Average) and Facebook Prophet, both of which are widely used in time-series analysis. These models are often complemented by more advanced architectures, such as LSTMs (Long Short-Term Memory networks), RNNs (Recurrent Neural Networks), TCNs (Temporal Convolutional Networks), or Transformers, that enhance accuracy but require significant computational resources.

The study was conducted in a populated environment, where light exposure at the endpoints was significantly influenced by human movement. This caused notable variability in the signal, even with low-frequency sampling. For example, some rooms would transition from darkness to light when bulbs were switched on, while moving people introduced fluctuations as their shadows blocked the light reaching the LDR. To mitigate this, a rolling window average was applied to filter out outliers within a four-hour window, identifying samples with a standard deviation exceeding 0.8 as outliers. Additionally,

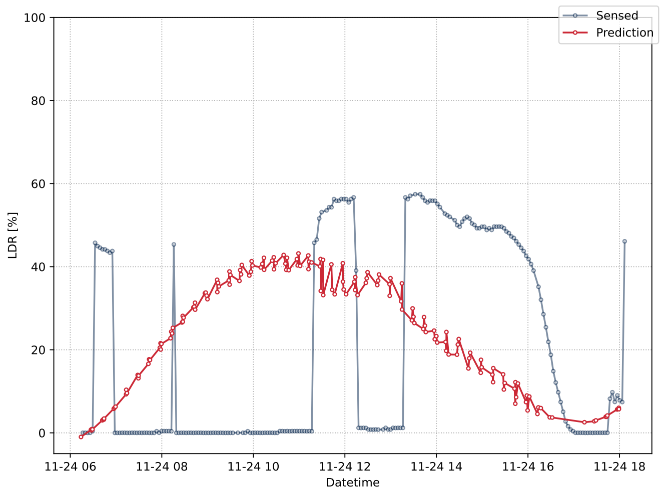


Fig. 5. Prediction oscillations

standard scaling was performed on the data before inputting it into the Prophet model to enhance accuracy.

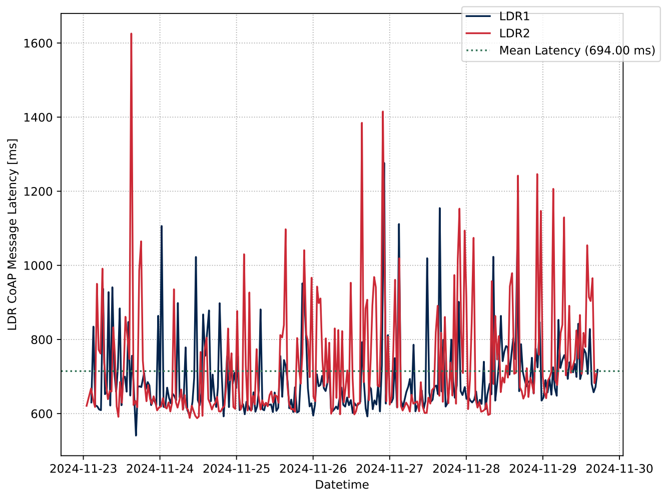
The model incorporated both daily and weekly seasonality to account for the short data acquisition period. A holiday factor was also included to enhance accuracy, although its effects are expected to be more significant with larger datasets. Since weather seasonality strongly affects sunlight hours, incorporating yearly seasonality could be beneficial. However, with the limited dataset used in this study, including this parameter increased the error and was therefore omitted.

To improve consistency across prediction timestamps, a custom function was developed to generate future points. This approach made it possible to reduce the time intervals between predictions and extend the prediction horizon without the irregular oscillations shown in Fig. 5. In this way, predictions were made every 15 minutes with an horizon of 6 hours.

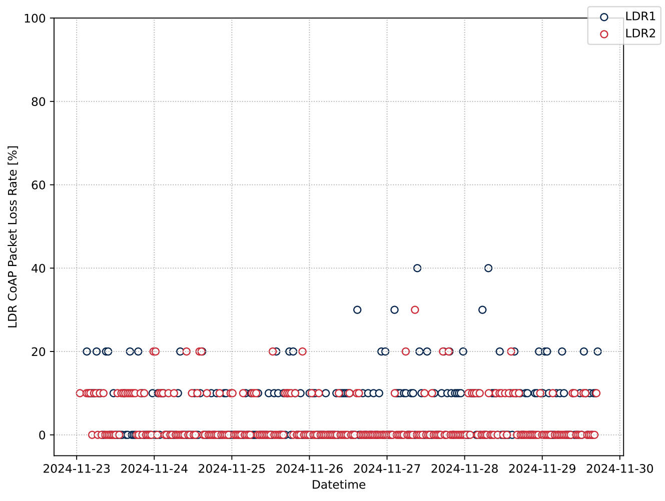
To streamline the process, a basic comparator was chosen as the optimizer. The control unit's primary role is to notify the user via a Telegram bot on their smartphone, prompting them to reposition the plant pots as needed. The comparator evaluates optimal plant placement by analyzing the average conditions over the past 6 hours and predicting the average conditions for the next 1 hour. It then updates the user on the current and expected conditions. To improve the clarity of these alerts, a time series plot is included, enabling the user to make informed decisions about whether to take action.

D. Frontend CLI and Dashboard

To give users full control over the endpoints, a CLI was developed, as previously mentioned. This tool allows users to define various details about the sensor node, including its spatial positioning, the type of houseplant, and the sampling period. Additionally, the interface serves as a frontend application, displaying this information. When a command is issued, the system promptly recognizes the action and updates the sensors via the MQTT protocol. This setup ensures complete customization and scalability of the WSN.



(a) Mean latency over 30 minutes windows



(b) PLR over 30 minutes windows

Fig. 6. LDR sensors performance

Similar functionalities are available in the Telegram bot, which acts as a notifier for the best spots for plants in the upcoming hours.

A Grafana dashboard was developed to present all relevant information, catering to the needs of the most inquisitive users. The panels display the most recent light exposure measurements alongside a time series of all collected samples, overlaid with future predictions. To convey the inherent limitations of the predictions, due to the limited data available, the dashboard also includes the upper and lower uncertainty bounds provided by the Prophet model. This transparency allows users to understand the variability of the forecasts and make informed decisions. Over time, as more data is collected, the system's predictive accuracy is expected to improve.

E. Metrics

Several metrics were considered to comprehensively evaluate the system's performance, from sensor nodes to the predictive model's accuracy.

1) Sensor node: The sensor nodes were assessed by measuring the mean latency of data transmission via the CoAP protocol. Latency was defined as the time elapsed between the transmission of a packet by the endpoint and the reception of the server's response by the sensor node. Measuring the time directly on the board ensured a more precise evaluation. The mean latency within 30-minute windows was calculated and stored in an InfluxDB instance for visualization in a dedicated Grafana panel.

Additionally, failures within the same time window were counted, as some sensors relied on home mesh Wi-Fi extenders to connect to the Internet. The Packet Loss Rate (PLR) was calculated as a measure of system health using the following equation

$$\mathrm{PLR}(\%) = \frac{\text{n. messages lost}}{\text{n. messages sent}} \cdot 100$$

This metric could be extended with an anomaly detection feature to identify and alert the user to connection losses.

These measurements were made possible by leveraging RTC Fast Memory storage, which allows certain variables, such as latency accumulation and missed transmission counter, to persist across deep sleep cycles, where the RAM is typically flushed. This capability ensured reliable data retention and accurate latency evaluation even under low-power operating conditions.

2) Predictive model: The performance of the predictive model was evaluated using the Mean Squared Error (MSE) between the predicted and observed values, calculated as:

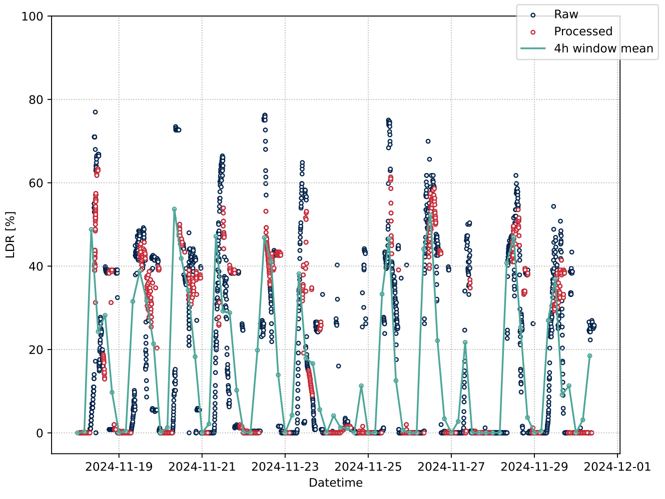
$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

To enable a meaningful comparison between the predicted and sensed signals, which had significantly different sampling rates, a rolling window mean of the sensed values was computed.

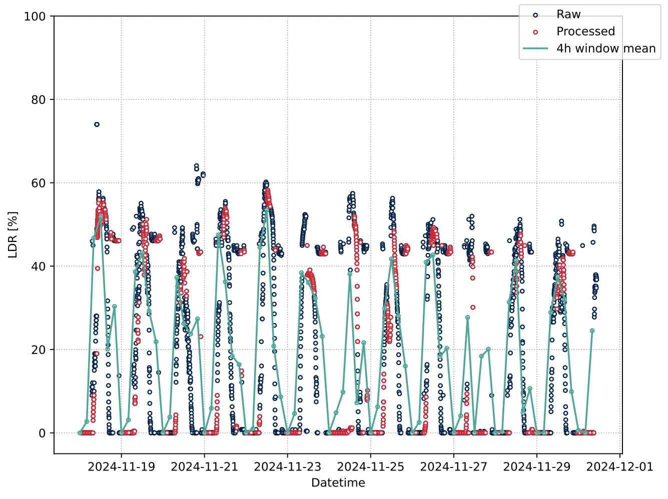
IV. Results

As shown in Fig. 6a, the average mean latency is approximately 690 ms. Although the expected mean latency was around 200 ms, this result ensures the system's reliability, as the sensor's latency remains well below the sampling rate. The relatively high latency may be attributed to the complexity of the Wi-Fi network, which relies on Network Mesh Extenders.

As highlighted in Fig. 6b, the Packet Loss Rate (PLR) is kept under control by the network. The accumulation of PLR over the 30-minute period was primarily caused by the continuous alternation between deep sleep mode and wake-up mode. Notably, the endpoint frequently encountered difficulties when attempting to reconnect to the Wi-Fi network. Furthermore, the PLR also accounts for the failed CoAP steps, which were covered by the direct connection to the InfluxDB instance. This



(a) LDR 1



(b) LDR 2

Fig. 7. LDR raw data vs. pre-processed. In green the rolling mean of the raw data is plot, using a window of 4 hours.

means that the effective packet loss was much lower than the values depicted in the figure.

Nevertheless, the accumulated PLR never exceeded 40% of the wake-up cycles. Since rapid changes in light did not significantly affect the task, and the primary control component relied on the mean value over 6 hours of acquisitions, having more than 50% of the sampling data correctly transmitted was sufficient for a reliable solution. No correlation was observed between latency and PLR, as missing packets, represented by an infinite latency, were excluded during the pre-processing phase.

The pre-processing stage was effective in eliminating high-frequency variations in light, as only the mean values over hourly periods were relevant for the application. The results of the pre-processing stage are shown in Fig. 7.

Using this pre-processed data, the model performed better. Specifically, the resulting MSE was reduced to 109.33% compared to over 200% for LDR1, and 122.12% for LDR2. However, the model's performance remains suboptimal, primarily due to the limited amount of data collected (approximately two weeks). A longer data collection period is expected to improve performance, particularly by enabling the inclusion of additional features such as yearly seasonality and the holiday factor, which were not fully exploited during the analyzed period. Nevertheless, the model appears to fit the data trend, as shown in Fig. 8.

V. Conclusion

The WSN demonstrated excellent performance in terms of reliability and latency, validating the decision to make the model redundant and send sensed data directly to the InfluxDB instance if the CoAP service failed. Furthermore, latency remained relatively low despite the complexity of the network, which included router devices such as Wi-Fi Mesh Extenders.

Future implementations could consider a more advanced WSN topology, transitioning from a star to a mesh configuration. This change would enhance system reliability by enabling multiple redundant communication paths. Additionally, as the number of endpoints increases, a BLE-based approach should be considered to further reduce energy consumption and extend the battery life of individual devices. However, the maximum reliable range for BLE is approximately 10 meters, meaning that sensors would need to be strategically distributed throughout the environment, potentially increasing costs significantly.

Despite the added cost, a BLE mesh topology could improve data sensing capabilities, as each device could handle both routing and sensing during wake-up cycles. This would result in more data being passed to the model, thereby improving its performance.

A final enhancement to the system could involve implementing remote updates via a bootloader. This would enable the entire system to be consistently updated without requiring physical access to individual end devices, a task that is not always straightforward.

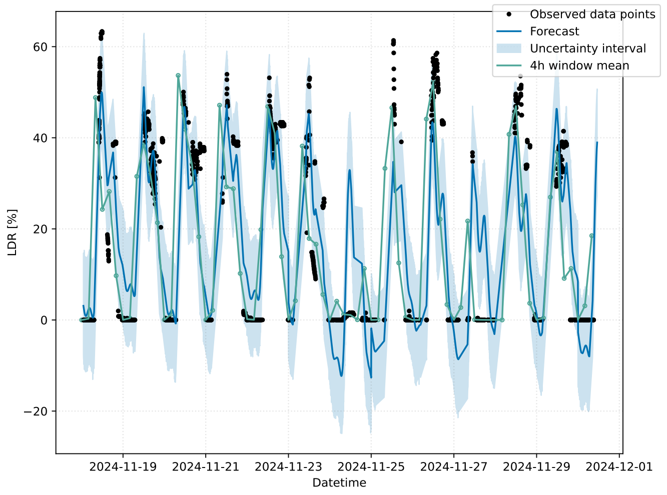
The forecasting unit did not behave as expected. The big variability of the signal, due to human presence in the environment which produce oscillation related to moving shadows, made the predictive model to be really bad performing. A system of this kind may require a bigger amount of data to be processed, or a much more complex predictive model should be required, such as DNN ones.

However, the alert system was well behaving, making the data-driven methodology valid for the application.

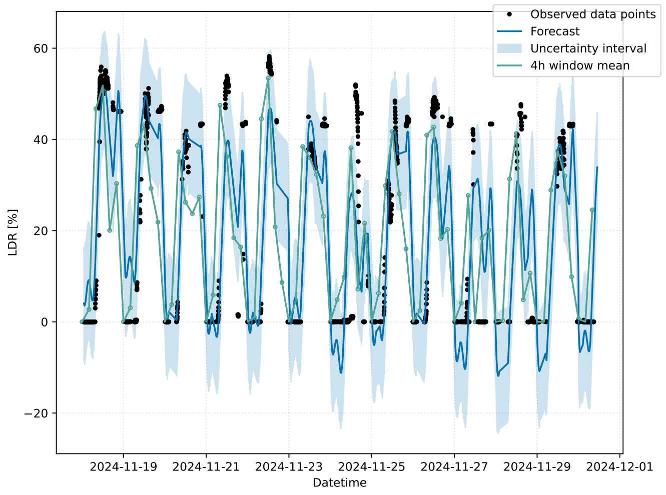
References

[1] Kris Kuciel, *The Truth on the Benefits of House Plants*

[2] Girman, John and Phillips, Tom and Levin, Hal, *Critical review: how well do house plants perform as indoor air cleaners*, Proceedings of healthy buildings, 2009



(a) LDR1, MSE: 109.33%



(b) LDR2, MSE: 122.12%

Fig. 8. LDR predictions

[3] Kim, Ho-Hyun and Yang, Ji-Yeon and Lee, Jae-Young and Park, Jung-Won and Kim, Kwang-Jin and Lim, Byung-Seo and Lee, Geon-Woo and Lee, Si-Eun and Shin, Dong-Chun and Lim, Young-Wook, *House-plant placement for indoor air purification and health benefits on asthmatics*, Environmental health and toxicology, 2014

[4] Schwenzer, M., Ay, M., Bergs, T. et al. *Review on model predictive control: an engineering perspective*. Int J Adv Manuf Technol 117, 1327–1349, 2021.

[5] Holkar, K. S.; Waghmare, Laxman M. *An overview of model predictive control*. International Journal of control and automation, 2010, 3.4: 47-63.

[6] Bieker, Katharina, et al. *Deep model predictive control with online learning for complex physical systems*. arXiv preprint arXiv:1905.10094, 2019.

[7] De Leon, Eduardo; Nabi, Majid. *An experimental performance evaluation of bluetooth mesh technology for monitoring applications*. In: 2020 IEEE Wireless Communications and Networking Conference (WCNC). IEEE, 2020. p. 1-6.

[8] Espressif Systems, *Analog to Digital Converter (ADC) Calibration Driver*. Available: https://docs.espressif.com/projects/esp-idf/en/v5.3.1/esp32c6/api-reference/peripherals/adc_calibration.html

[9] Espressif Systems, *ESP-DL: Espressif deep-learning library for AIoT*. Available: https://docs.espressif.com/projects/esp-dl/en/latest/esp32/introduction.html