Jurnal Nasional Teknik Elektro dan Teknologi Informasi
Volume 14 Number 2 May 2025

Translation of 10.22146/jnteti.v14i2.16272

Data Duplication Detection in IoT-based Air Quality Monitoring System

Dwi Ilham Maulana[1], Asep Andang[1], Ifkar Usrah[1], Agus Purnomo[2]

[1] Electrical Engineering Department, Faculty of Engineering, Siliwangi University, Tasikmalaya, Jawa Barat 46115, Indonesia
[2] Jurusan Teknologi Laboratorium Medis, Poltekkes Kemenkes Tanjungkarang, Bandar Lampung, Lampung 35145, Indonesia

[Submitted: 19 September 2024, Revised: 14 December 2024, Accepted: 16 April 2025]
Corresponding Author: Asep Andang (andhangs@unsil.ac.id)

ABSTRACT — The increasing volume of data on the Internet of things (IoT)-based systems has driven the need for efficiency in data management, particularly in air quality monitoring systems. One approach to address this challenge is data duplication detection, which works to eliminate redundant data to reduce storage requirements and power consumption. This study aims to develop an IoT-based air quality monitoring system incorporating a data duplication detection method as part of an effort to support the green IoT concept. The methodology involved a comparative analysis between systems with and without the implementation of data duplication detection, accompanied by a comprehensive evaluation of system performance. The data tested included the size of transmitted data and device power consumption during the transmission process. Testing was conducted under real operational conditions over a 24-hour period. The results indicate that the implementation of data duplication detection successfully reduced the size of transmitted data from 56 bytes to 11–44 bytes, depending on the level of data redundancy. Power consumption was reduced by 1.59% to 3.84% compared to the system without data duplication detection. This method was also proven not to affect the accuracy of the displayed data, thereby maintaining the system's functional requirements. In conclusion, the implementation of the data duplication detection method in an IoT-based air quality monitoring system not only optimizes data transmission processes but also supports energy efficiency in line with the principles of green IoT. This research provides a significant contribution to the development of more sustainable and energy-efficient IoT systems.

KEYWORDS — Duplicate Data Detection, Inline Duplicate Data Detection, IoT, Green IoT, Air Quality Monitoring System.

I. INTRODUCTION

An air quality monitoring system is an electronic device designed to detect air quality in both indoor and outdoor environments. For outdoor air quality monitoring, device placement must take into account the surrounding meteorological conditions, as these significantly influence the rate of air pollution dispersion [1]. Given such meteorological factors, it is possible to have two or more air quality monitoring devices operating within the same area. Concurrently, the increasing deployment of monitoring devices, particularly those based on the Internet of things (IoT), has drawn global attention, especially concerning the electrical energy consumption of each IoT device. As IoT technology continues to advance, it is projected that by 2025, the total electricity consumption of IoT devices will reach the annual electricity consumption level of Portugal, which is approximately 46 TWh [2]. Even though the information and communication technologies (ICT) sector is not among the largest contributors to carbon emissions and greenhouse gases, it accounts for more than 2.5% of the total global toxic emissions [3]. This has led to the emergence of the green IoT concept [4], [5].

Green IoT aims to improve energy efficiency in IoT devices through modifications to both hardware and software components [6]. On the software side, optimization can be achieved by altering the data transmission flow from sensor readings to the server, primarily by reducing the size of the data being transmitted.

Previous studies have shown that techniques such as dynamic subsampling, data fusion, and data scaling can reduce data size from 96 bytes to 50 bytes [7]. However, these methods compromise data accuracy. This highlights an urgent need for alternative approaches that can reduce data size without sacrificing accuracy.

Data duplication detection is a promising method to address this challenge. Unlike previous techniques, data duplication detection manipulates only identical segments of data, thereby preserving the integrity of the information [8]. This method can be applied either before or after data storage. When applied prior to storage, it enables the reduction of data transmitted by the sensor, whereas post-storage approaches only affect the server-side database.

Based on this research gap, the present study aims to develop an IoT-based air quality monitoring system that incorporates data duplication detection to minimize the continuous transmission of redundant data. This method is expected to reduce electricity consumption, thereby supporting the principles of green IoT [9]. The contribution of this study lies in offering an air quality monitoring solution that integrates the green IoT concept, with a particular emphasis on optimizing data transmission processes through the application of data duplication detection without compromising accuracy. Through this research, a green IoT-compliant air quality monitoring system is expected to be realized, with its primary focus on enhancing data transmission efficiency via duplication detection while maintaining data accuracy.

II. AIR QUALITY MONITORING SYSTEM

A. AIR POLLUTION

Air pollution refers to the entry or introduction of substances, energy, and/or other components into ambient air,

rendering it incapable of performing its intended functions effectively. Ambient air is the free air in the Earth's troposphere, which is essential for and affects human health, living organisms, and other environmental elements. Pollution is primarily caused by emissions resulting from human activities. Accordingly, emission standards have been established to regulate the amount of emissions that may be released into ambient air [10].

To prevent air pollution, the government has established ambient air quality standards. These standards are divided into national and regional ambient air quality standards. Regional standards are determined based on the national standards and specific local environmental conditions. In cases where a regional government has not established its own standards, the national standards shall apply.

The Government of Indonesia, through its ambient air quality monitoring stations, operates a national air monitoring system. Monitoring results are presented in the air quality index (AQI). The AQI utilizes a color-coded status to indicate the air quality condition in a given location. This classification is based on its impact on human health, aesthetics, and other living organisms. The status color is derived from the AQI numerical range. AQI parameters include PM10 and PM2.5 particulate matter, carbon monoxide (CO), nitrogen dioxide (NO_2), sulfur oxides (SO_2), ozone (O_3), and hydrocarbons (HC) [11].

B. DATA DUPLICATION DETECTION

Data deduplication is a method for detecting and eliminating redundant data to improve storage efficiency [8], [12], as illustrated in Figure 1. This method is crucial in large-scale storage systems as it reduces data duplication, increases available storage capacity, and lowers operational costs [13].

Generally, data deduplication can be categorized based on its placement, timing, and deduplication algorithm [14]. Based on placement, deduplication is divided into client-based deduplication, in which the deduplication process is entirely performed on the client side; deduplication appliance, in which the process is handled by a third-party device; and storage arrays, where deduplication occurs on the server or storage location of the client's data. Based on timing, deduplication is categorized into synchronous/in-band deduplication, which is performed before data are written to storage, and asynchronous/out-of-band deduplication, which occurs periodically. In terms of algorithms, deduplication techniques include whole-file hashing, sub-file hashing, and delta encoding.

The data deduplication process generally consists of chunking, fingerprinting, mega chunk formation, duplication detection, index updating, and storing unique data [15]. Chunking refers to dividing data into segments of specific sizes. Fingerprinting is the process of generating hash values or hash signatures [14] from data chunks. Mega chunk formation involves aggregating data chunks that have undergone fingerprinting. The hash signatures of data chunks are then compared with the metadata index that stores unique data entries to detect duplication. If no duplication is found, the data chunk is considered unique. The unique data are stored and the index is updated as a reference for future duplication detection. Metadata is "data within data" that describes specific information [16].

Data deduplication is a vital aspect of modern storage technologies and is regarded as an efficient method for optimizing data storage capacity [14]. However, the development of deduplication techniques faces several challenges, such as the rationality of chunk segmentation, performance optimization, the necessity to ensure data reliability, system scalability, and the placement of deduplicated data [8]. Compared to other storage optimization methods, data deduplication entails relatively high overhead in both computation and storage processes. Therefore, continuous development of deduplication methods is essential.

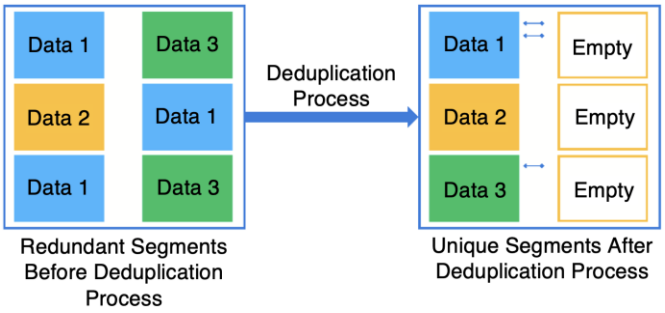

Figure 1. Data duplication detection process.

Low-overhead inline data deduplication is an in-band or inline deduplication technique [17]. This technique employs two levels of fingerprinting: a weak fingerprint for fast duplication detection, and a strong fingerprint for more accurate analysis. According to a prior study [17], low-overhead inline data deduplication demonstrates superior performance and satisfactory data write times compared to conventional data deduplication techniques. The performance improvement of the low-overhead inline data deduplication is supported by the implementation of adaptive sampling deduplication detection. Adaptive sampling deduplication detection is a constraint mechanism used during duplicate data detection. If no duplication is found within the initial *D%* of data blocks, the system may skip the remaining blocks. However, if duplication is detected before reaching the *D%* threshold, a full duplication detection process is executed. This approach accelerates deduplication by reducing the system's workload.

C. GREEN IOT

The term "green" in green IoT refers to environmentally friendly and energy-efficient characteristics, applied to both hardware and software components. In simple terms, green IoT is a low-power version of the Internet of Things (IoT), proposed to reduce energy consumption by IoT devices amid the growing global adoption of IoT technology [9].

IoT is a network connecting various devices through distinct identification elements, sensors, embedded intelligence, and ubiquitous internet connectivity [18]. The main idea of IoT is to interconnect physical objects and process their data through the internet for control or monitoring purposes. With the advancement of network technologies, IoT continues to gain attention for enabling seamless communication, connectivity, and controllability of objects anytime and anywhere.

Figure 2 illustrates a general architecture of green IoT. An IoT system—from planning to implementation—must adhere to "green" principles [19], which can be applied at both the hardware and software levels. A typical green IoT architecture incorporates technologies such as green cloud computing, green radio frequency identification (RFID), green wireless sensor networks (WSN), green machine-to-machine

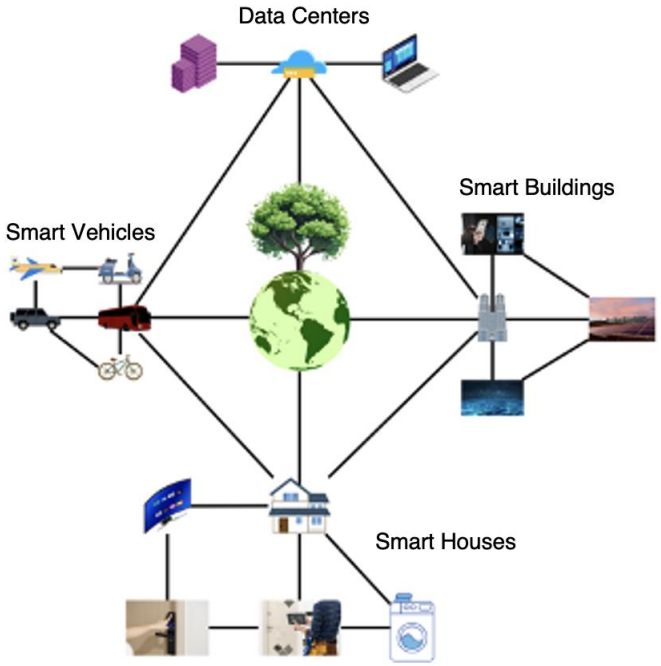

Figure 2. Green IoT architecture.

communication, and green data centers. However, there is currently no universally accepted architecture for green IoT, which makes it difficult to focus development efforts on fundamental aspects [6]. As a result, dedicated standardization efforts are required, and a committee has been formed to establish such standards. Presently, this committee is focusing on determining protocols to interconnect various types of networks and heterogeneous devices.

III. METHODOLOGY

The system operates by implementing a data duplication detection method applied to data to be transmitted by the node. The process began by dividing the data into several segments, each consisting of 32 characters. After segmentation, each part was processed in two fingerprinting steps: the first fingerprinting and the second fingerprinting.

The result of the first fingerprinting was compared with previously stored data. If a match was found, the comparison proceeded with the result of the second fingerprinting. If another match was found during the second comparison, the system retrieved the index corresponding to that segment. This comparison process continued until a match was found, as long as the previous data remained available.

Once the comparison process was completed, the next step was to recombine the segments using a semicolon (";") delimiter. Segments that matched previously existing data were represented by the index of the matching data. After reassembling all segments, the resulting data was transmitted to the server.

On the server side, the data received from the node was separated using the delimiter character. These segments were then examined to determine whether they represented server indexes or original data. If a segment was identified as a server index, the system retrieved the corresponding original data from the database, replacing the server index with the original data. If a segment contained original data, it was stored in the database and assigned a server index. After all segments were processed, they were reassembled without the delimiter, reconstructing the original data from the node. The server indexes corresponding to the original data segments were then sent back to the node. When the node received the server index data, it stored this information along with the segments that did not match any previously stored data.

The system functioned by applying a data duplication detection method. Figure 3 illustrates the workflow of the data duplication detection system in the IoT-based air quality monitoring system. The duplication detection process was conducted after the sensor performed its reading. The readings from each sensor were combined into a single JavaScript Object Notation (JSON)-formatted data object, including the node ID, which had been registered in the server database. JSON was used as an alternative data format to Extensible Markup Language (XML) for data exchange and storage. It consisted of key-value pairs, enabling efficient data communication.

Once the sensor reading data and node ID were combined into JSON format, the data underwent the duplication detection process. The result of this process was then transmitted to the server for storage in the server database.

Figure 3 illustrates the workflow of the data duplication detection system in the IoT-based air quality monitoring system. The detection mechanism operates on both the node and the server. Therefore, synchronization between the node and the server was crucial to ensure consistency between the sensor readings and the data displayed in the application or stored on the server.

Duplication detection on the JSON data was performed by dividing the JSON data into several segments referred to as chunks. Each chunk contained a maximum of 32 bytes. These chunks were then subjected to two fingerprinting processes: the first was applied to half of the chunk's data, and the second to the entire chunk. The fingerprinting was performed using the MurmurHash3 hashing algorithm. MurmurHash is a non-cryptographic hash function designed for general-purpose hash-based lookups, developed by Austin Appleby in 2008, with MurmurHash3 being its latest version [20]. The fingerprinting results were used as reference data to identify duplicates. The use of MurmurHash3 accelerated the fingerprinting process, as it was the fastest hashing algorithm [20] compared to other widely used algorithms, such as SHA-256 and SHA-512.

The comparison data refers to the data utilized by the duplicate detection method to determine whether a particular piece of data is a duplicate. In this case, the data being detected are data chunks.

The processed data chunks were then converted into fingerprint one and fingerprint two. These two fingerprints were compared with the previously processed fingerprint data. If there was a matching fingerprint, the corresponding data chunk was identified as a duplicate. Conversely, if there was no match, the data chunk was considered unique. If, during the process, two identical data chunks existed but only one was included in the comparison set, the system did not recognize the chunk as a duplicate. After the comparison process, data chunks identified as duplicates were replaced with server index data, whereas nonduplicate data chunks remained unchanged. All compared data chunks were then compiled into a single list, separated by the ";" character, and transmitted to the server.

The server index data refers to numeric values obtained after a data chunk is stored in the database. These numerical values represent the position of the chunk within the database

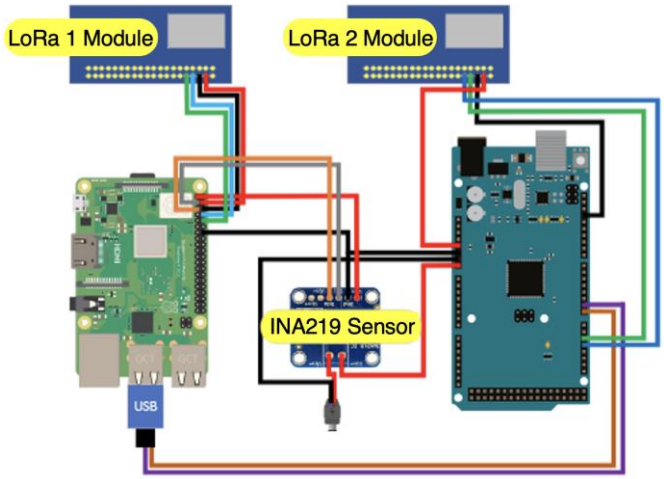

Figure 3. Software testing schematic.

so that when the chunk is needed, it can be retrieved based on its position. In the list generated at the node, if any of the list elements was a data chunk, the server first stored that chunk into the database, and then the resulting server index was stored and sent back to the node. If th e element was already identified as a server index, this step was skipped.

The determination of whether an element in the list is a data chunk or a server index is based on its data type. If the data type is string, the element is a data chunk; if the data type is integer, the element is a server index. After all detected data chunks were stored in the database, the server index values were replaced with their corresponding data chunks, resulting in a list composed entirely of data chunks.

This complete list of data chunks was then combined to form the latest sensor reading data. The latest sensor reading data here referred to the JSON data previously generated by the node before undergoing the duplicate detection process.

After the latest sensor reading data was stored, the newly generated server index data was sent back to the node in the form of a list, separated by commas (“,”) and ending with a semicolon (“;”). These server index data were stored at the node as new comparison data and were merged with the corresponding fingerprint one and fingerprint two data.

When data are transmitted from the node to the server and vice versa, a gateway serves as an intermediary to facilitate communication between the node and the server. This is necessary because the node only supports LoRa communication. LoRa communication cannot directly connect to the internet and requires an intermediary device, namely a gateway. The gateway in the system transmits data to the server using the message queuing telemetry transport (MQTT) protocol.

On the Android application, the latest sensor reading data are obtained by periodically sending Hypertext Transfer Protocol (HTTP) requests to the server. The received data are then parsed into individual sensor readings: PM 1.0, PM 2.5, PM 10, temperature, humidity, and wind speed. The latest sensor reading data received by the Android application corresponds to the device ID of the IoT device, from which the latest sensor data is retrieved.

The latest sensor reading data consisted of the IoT device ID and the sensor reading data. The sensor reading data were composed of two parts, separated by the “|” character. The first part contained labels representing the combined sensor data. These labels included the characters “h,” “t,” “1,” “2,” “0,” and “v.” Label “h” represented humidity data, “t” represented temperature, “1” represented PM 1.0, “2” represented PM 2.5, “0” represented PM 10, and “v” represented wind speed. The second part contained the actual sensor readings, with values separated by commas (“,”). The sequence of the readings followed the label order in the first part. If a sensor did not transmit any data, that sensor’s label was not included, and the number of values in the second part was reduced accordingly. For example, if the wind speed sensor did not provide any reading, the label “v” would be absent in the first part, and the second part would contain one fewer value.

IV. RESULTS AND DISCUSSION

A. SOFTWARE TESTING WITHOUT DATA DUPLICATION DETECTION

Software testing without data duplication detection was conducted over a duration of three hours. The data collected included the data transmitted by the node along with its transmission time, the data received by the server along with its reception time, and the power consumption data of the node. The data transmitted during the test consisted of PM 1.0, PM 2.5, PM 10, temperature, humidity, and wind speed, with each data type generated as random values.

To monitor the power consumption during data transmission, the LoRa module on the node side was connected in series with the INA219 sensor, as illustrated in Figure 3. The INA219 sensor readings were obtained from the Serial Monitor of the Arduino IDE.

Figure 4 presents the power consumption graph of the node during the three-hour test. The average power consumption of the node over the three-hour period was 962.93 mW, and each data transmission required an average power of 2,266.04 mW.

B. SOFTWARE TESTING WITH DATA DUPLICATION DETECTION

Software testing with data duplication detection was performed following the configuration illustrated in Figure 4. This test evaluates the updated system software which incorporates a data duplication detection method. The purpose of the test was to assess the accuracy of the data and the electrical power required for data transmission by the LoRa module, resulting from the application of data duplication detection in the air quality monitoring system. The test was conducted under three different scenarios. In the first scenario, identical or static data were used throughout the three-hour test duration. The second scenario involved the use of random data—sensor readings generated as random values by Arduino Mega. The third scenario utilized server index data transmitted by the server in large sizes. Each scenario was executed over a three-hour period, resulting in a total of nine hours of combined testing.

Figure 5 shows the results of the software testing with data duplication detection where the sensor data transmitted was consistently identical in each reading period. Consequently, the data transmitted remained the same across each transmission interval. The results indicate that during the three-hour test, the node consumed an average power of 950.211 mW, with an average power consumption of 2,230.51 mW during each data transmission. This represents a 35.49 mW or 1.59% reduction compared to data transmission without the use of the data duplication detection method.

Jurnal Nasional Teknik Elektro dan Teknologi Informasi
Volume 14 Number 2 May 2025

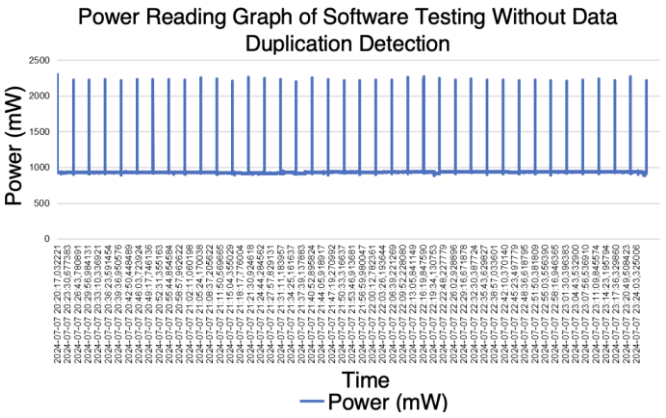

Figure 4. Power reading graph of software testing without data duplication detection.

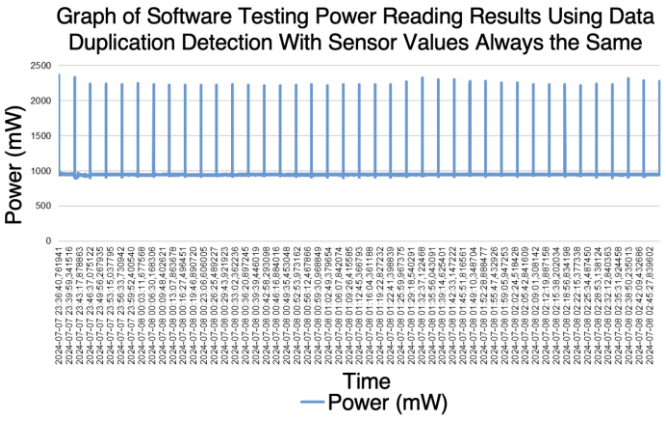

Figure 5. Graph of power consumption test results using data duplication.

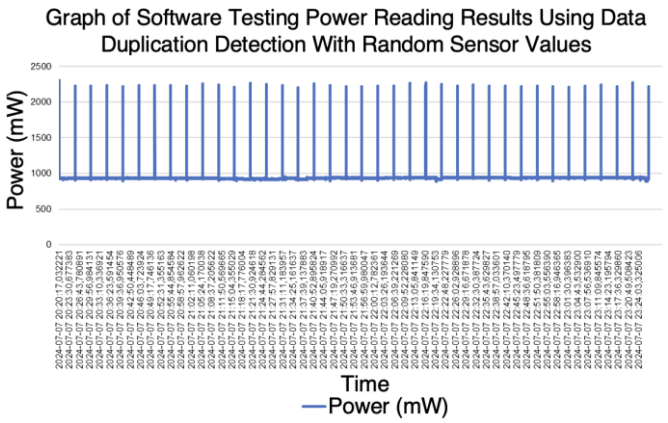

Figure 6. Graph of power consumption test results using data duplication

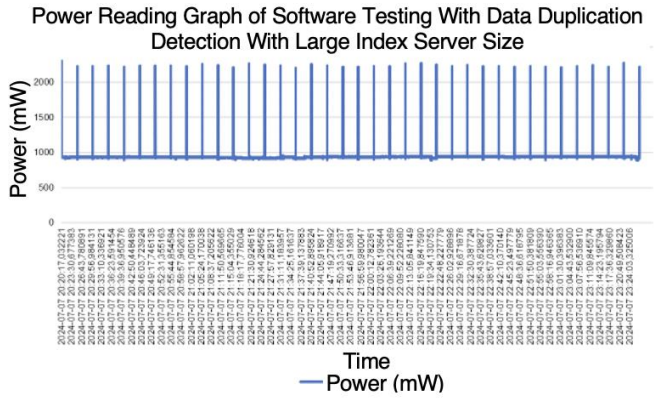

Figure 7. Graph of power consumption test results using data duplication detection with large-sized server index data.

Figure 6 presents the power consumption data from the test where the software incorporated data duplication detection and the sensor readings were randomly generated. As a result, the data transmitted by the node varied in each transmission period. The results show that during the three-hour test, the node consumed an average power of 937.039 mW, and the average power required for data transmission was 2,195.026 mW. This represents a reduction of 71.01 mW or 3.24% compared to the power consumption during software testing without data duplication detection.

Figure 7 displays the node's power consumption data over a three-hour test using server index data, which was transmitted in large sizes. The results indicate that the average power consumption of the node during the test was 807.30 mW, and the average power used during data transmission was 2,182.038 mW. This value is 3.85% lower than the power consumption without the use of data duplication detection.

Based on the results presented in Figure 4 through Figure 7, it can be concluded that the data duplication detection method effectively reduces power consumption. This reduction occurs because the method minimizes the amount of data received by the LoRa module, leading to a lighter processing load. With fewer data to process, the power consumption of the LoRa module decreases accordingly.

C. SYSTEM TESTING

System testing was conducted to determine whether the developed system functions as intended. This was verified by the successful display of sensor readings—including PM 1.0, PM 2.5, PM 10, temperature, humidity, and wind speed—on the Android application. The test involved comparing the readings obtained from the node with the data displayed in the Android application over a 24-hour testing period. If the displayed data are consistent between the two, it indicates that the system is functioning properly.

Table I shows the first ten and last ten data records from the 24-hour system testing. The table demonstrates that the sensor readings from the node and the data received by the Android application are identical. This confirms that the data duplication detection system operates correctly, as evidenced by the consistency between the sensor data and the data displayed in the Android application.

The system testing demonstrates that, even when data manipulation occurs prior to transmission from the node, the IoT system is still capable of delivering the latest sensor readings accurately and without error. Errors in this context may arise if the server index used is incorrect, leading to a mismatch between the transmitted server index and the intended chunk position within the database. Additional errors may result from an improper sequence of data during the duplication detection process. Such misalignment can cause the reconstruction of data chunks in an incorrect order, rendering the data received by the Android application incoherent. For instance, a data chunk that should appear at the beginning may be positioned at the end, or a chunk intended for the middle may appear at the front. Consequently, the Android application is unable to process the data due to its invalid structure, preventing it from displaying the corresponding sensor readings.

V. ANALYSIS

Based on Figure 8, it can be observed that the use of data duplication detection with a server index data result in a larger

TABLE I
SYSTEM TESTING RESULTS

PM 1,0		PM 2,5		PM 10		Temperature		Air Humidity		Wind Speed	
Node	*Android*	*Node*	*Android*	*Node*	*Android*	*Node*	*Android*	*Node*	*Android*	*Node*	*Android*
25	25	34	34	41	41	26	26	77	77	0	0
28	28	36	36	45	45	26	26	77	77	20	20
28	28	36	36	45	45	26	26	78	78	0	0
25	25	34	34	43	43	26	26	78	78	0	0
22	22	30	30	38	38	26	26	79	79	0	0
…	…	…	…	…	…	…	…	…	…	…	…
26	26	34	34	39	39	35	35	55	55	0	0
26	26	34	34	39	39	35	35	55	55	0	0
23	23	31	31	35	35	35	35	55	55	0	0
23	23	31	31	35	35	35	35	56	56	0	0
23	23	32	32	35	35	35	35	54	54	0	0

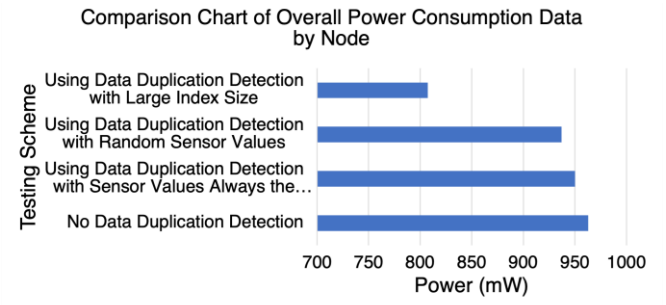

Figure 8. Comparison of overall power consumption between systems without and with data duplication detection.

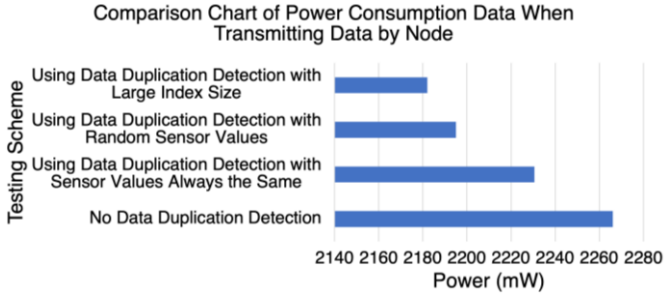

Figure 9. Comparison of power consumption during transmission for systems without data duplication detection and systems using duplication detection.

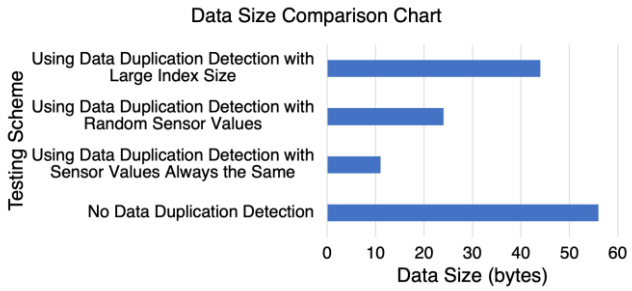

Figure 10. Comparison of data sizes sent by software without duplicate data detection and software with duplicate data detection.

data size but records the lowest average power consumption compared to the other test scenarios. This is attributed to lower power consumption during the node's idle process, which only consumes approximately 800 mW, compared to around 900 mW in the other tests.

Figure 9 presents a comparison of the average power consumption by the node in each test during data transmission, measured in mW. The test employing data duplication detection using the large index data server scenario shows a reduction in power consumption by 3.85% compared to the test without data duplication detection. Other tests involving data duplication detection with random sensor values and constant sensor values demonstrate power consumption reductions of 3.24% and 1.59%, respectively. These results confirm that data duplication detection can reduce power consumption by decreasing the amount of data transmitted from the node to the server, with power reductions ranging from 1.59% to 3.84%.

Figure 10 illustrates a comparison of the data size transmitted by the node for systems implementing data duplication detection and those without, measured in bytes. It is evident that the implementation of data duplication detection reduces the data size to between 11 and 44 bytes, compared to 56 bytes in systems without duplication detection. When the index data approaches its maximum capacity, the size of the index data is 19 bytes. Therefore, it can be concluded that the use of data duplication detection reduces the data size from 58 bytes to approximately 40–45 bytes.

System testing further confirms that the data duplication detection method functions correctly when the node is connected to the sensors. This is evidenced by the consistency of sensor reading values between the node and the Android application. These values include readings of PM 1.0, PM 2.5, PM 10, temperature, humidity, and wind speed.

VI. CONCLUSION

The data duplication detection system has been successfully implemented in the air quality monitoring system, as evidenced by the identical values of PM 1.0, PM 2.5, PM 10, temperature, humidity, and wind speed recorded by both the Android application and the node. Furthermore, the data duplication detection mechanism effectively reduces the size of transmitted data, from 56 bytes to a range between 11 and 44 bytes. This reduction occurs when the transmitted data contains segments that are similar to previously sent data chunks. In such cases, the existing data chunks are replaced by their corresponding indices from the server, rather than retransmitting the original data, resulting in more efficient data transmission. If the detected data are new, it is transmitted in full and processed by the server. Once more than ten unique data entries are identified during the initial transmission, subsequent data are consistently treated as unique, even if duplicates are present. Under ideal conditions with no communication errors, the implementation

of data duplication detection can reduce data usage by 11 to 44 bytes and save power consumption by 1.59% to 3.84% compared to systems without data duplication detection.

AUTHORS' CONTRIBUTIONS

Conceptualization, Dwi Ilham Maulana, Agus Purnomo, and Asep Andang; methodology, Asep Andang; software, Dwi Ilham Maulana; validation, Dwi Ilham Maulana, Asep Andang, and Agus Purnomo; formal analysis, Agus Purnomo; investigation, Dwi Ilham Maulana, and Asep Andang; resources, Ifkar Usrah; data curation, Asep Andang, and Agus Purnomo; writing-original drafting, Dwi Ilham Maulana; writing-reviewing and editing, Dwi Ilham Maulana and Asep Andang; visualization, Dwi Ilham Maulana; supervision, Ifkar Usrah; project administration, Ifkar Usrah; funding acquisition, Agus Purnomo.

ACKNOWLEDGMENT

The authors would like to express their gratitude to the Ministry of Health of the Republic of Indonesia for funding this research through the 2024 SIMLITABKES grant administered by the Health Polytechnic of the Ministry of Health, Tanjung Karang, Bandar Lampung. Appreciation is also extended to the Department of Electrical Engineering, Siliwangi University, for their collaboration in this research.

REFERENCES

[1] J. Wang and S. Ogawa, "Effects of meteorological conditions on PM2.5 concentrations in Nagasaki, Japan," *Int. J. Environ. Res. Public Health*, vol. 12, no. 8, pp. 9089–9101, Aug. 2015, doi: 10.3390/ijerph120809089.

[2] R. Kyburz, "Energy efficiency of the Internet of things," 2016. [Online]. Available: https://www.iea-4e.org/wp-content/uploads/publications/2016/08/160704_EE-IoT-Policy-Options_v1.8_-_FINAL_with_cover.pdf

[3] A.S.H. Abdul-Qawy and T. Srinivasulu, "Greening trends in energy-efficiency of IoT-based heterogeneous wireless nodes," in *Int. Conf. Electr. Electron. Comput. Commun. Mech. Comput. (EECCMC)*, 2018, pp. 1–10.

[4] R. Raut *et al.*, *Green Internet of Things and Machine Learning*. Hoboken, NJ, USA: John Wiley & Sons, 2022.

[5] B. Mahapatra and A. Nayyar, *Green Internet of Things*. Boca Raton, FL, USA: CRC Press, 2022.

[6] F.K. Shaikh, S. Zeadally, and E. Exposito, "Enabling technologies for green Internet of things," *IEEE Syst. J.*, vol. 11, no. 2, pp. 983–994, Jun. 2017, doi: 10.1109/JSYST.2015.2415194.

[7] J. Botero-Valencia, L. Castano-Londono, D. Marquez-Viloria, and M. Rico-Garcia, "Data reduction in a low-cost environmental monitoring system based on LoRa for WSN," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3024–3030, Apr. 2019, doi: 10.1109/JIOT.2018.2878528.

[8] X. Zhang and M. Deng, "An overview on data deduplication techniques," in *Inf. Technol. Intell. Transp. Syst.*, 2016, pp. 359–369, doi: 10.1007/978-3-319-38771-0_35.

[9] X. Liu and N. Ansari, "Toward green IoT: Energy solutions and key challenges," *IEEE Commun. Mag.*, vol. 57, no. 3, pp. 104–110, Mar. 2019, doi: 10.1109/MCOM.2019.1800175.

[10] "Pengendalian Pencemaran Udara," Regulation of Government of the Republic Indonesia, No. 41, 1999.

[11] "Pengendalian Pencemaran Udara," Regulation of the Minister of Environment and Forestry of the Republic of Indonesia, No 14, 2020.

[12] P. Abbareddy, S. Bhukya, C. Narsingoju, and B. Narsimhulu, "A novel methodology for secure deduplication of image data in cloud computing using compressive sensing and random pixel exchanging," *J. Theor. Appl. Inf. Technol.*, vol. 102, no. 4, pp. 1608–1618, Feb. 2024.

[13] R. Kaur, I. Chana, and J. Bhattacharya, "Data deduplication techniques for efficient cloud storage management: A systematic review," *J. Supercomput.*, vol. 74, pp. 2035–2085, May 2018, doi: 10.1007/s11227-017-2210-8.

[14] S. Michiels, Ed. *Companion '08: Proceedings of the ACM/IFIP/USENIX Middleware '08 Conference Companion*. New York, NY, USA: Association for Computing Machinery, 2008.

[15] J. Malhotra and J. Bakal, "A survey and comparative study of data deduplication techniques," in *2015 Int. Conf. Pervasive Comput. (ICPC)*, 2015, pp. 1–5, doi: 10.1109/PERVASIVE.2015.7087116.

[16] J. Riley, Understanding Metadata. Baltimore, MD, USA: National Information Standards Organization, 2017.

[17] W. Chen *et al.*, "Low-overhead inline deduplication for persistent memory," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 8, pp. 1–13, Aug. 2021, doi: 10.1002/ett.4079.

[18] A. Rayes and S. Salam, "Internet of things (IoT) overview," in *Internet of Things from Hype to Reality*. Cham, Switzerland: Springer, 2019, pp. 1–35.

[19] R. Arshad *et al.*, "Green IoT: An investigation on energy saving practices for 2020 and beyond," *IEEE Access*, vol. 5, pp. 15667–15681, Jul. 2017, doi: 10.1109/ACCESS.2017.2686092.

[20] B. Pan *et al.*, "Study on image encryption method in clinical data exchange," in *2015 7th Int. Conf. Inf. Technol. Med. Educ. (ITME)*, 2015, pp. 252–255, doi: 10.1109/ITME.2015.98.