International Journal of Frontier Technology and Engineering (IJFTE)
Vol. 02, No. 02, June 2024, pp. 87-97
ISSN: 2986-9064

Implementation of Data Parsing on Arduino Mega and NodeMCU ESP8266 for Web-Based Monitoring of E-Bike Charging Station

Aji Nugroho[1*], Mohammad Noor Hidayat[1], and Abdullah Faiq Munir [1]

[1]*Department of Electrical Engineering, Politeknik Negeri Malang, Malang, Indonesia*

Corresponding Author: Aji Nugroho, ajinugroho220891@gmail.com

Received Date: 10-03-2024 Revised Date: 16-05-2024 Accepted Date: 21-06-2024

Abstract

The growing trend for electric bicycles in Indonesia requires an efficient monitoring system for electric bicycle charging stations to ensure optimal performance and timely decision making. This study focuses on developing and testing a web-based monitoring system utilizing Arduino Mega and NodeMCU ESP8266 for real-time data transmission and analysis. The system integrates multiple sensors, including ACS712 current sensors, DC voltage sensors, and PZEM004T sensors, to monitor key parameters such as current, voltage, power, and energy. Data parsing, a crucial process for efficient transmission, is performed on the Arduino Mega before being sent to the NodeMCU ESP8266, which then forwards the data to a web server. Research uses the Research and Development (R&D) method, which includes stages of information gathering, system planning, implementation, and evaluation. Testing revealed that data parsing failures were due to data loss and format corruption caused by serial communication issues. Various transmission delays (0 ms, 100 ms, 200 ms and 500 ms) were tested to evaluate their impact on data loss, communication latency, and system responsiveness. The results indicate that a 200-ms delay offers an optimal balance between data transmission frequency and system stability, with minimal data loss and stable latency. The study also explored the effects of data reception delays on the web server, highlighting that lower transmission frequencies result in more stable reception delays and better overall performance. Efficient use of network resources and responsiveness of the system were achieved with appropriate delay configurations.

Keywords : Web-based Monitoring System, Data Parsing, Data Transmission, E-Bike Charging

1 Introduction

The trend for electric bicycles in Indonesia is increasing, as evidenced by the sale of electric motorcycles that reached 62,409 units in 2023 [1]. The popularity of electric bicycles is driven by various factors, including ease of use, exemption from taxes, and features such as a pedal assist that support battery efficiency. In addition, electric bicycles are associated with being environmentally friendly, equipped with advanced technologies such as digital speedometers and alarms, and are easier to maintain. Community support, particularly among women, also contributes to the growing popularity of electric bicycles [2]. Many cities around the world are beginning to provide electric bicycle rental stations as part of an integrated public transportation system [3]. However, to ensure the optimal operation of these electric bicycle stations, a reliable and efficient monitoring system is needed to track key parameters such as battery status, power consumption, and bike conditions in real-time.

One of the main challenges in the management of electric bicycle stations is to ensure that data collected from various sensors at the station can be efficiently processed and transmitted to a central control system or a web-based monitoring platform [4]. An effective approach is to integrate data from multiple sources to improve the efficiency of data-driven systems. This study employs the concept of data fusion from information science and proposes a framework for integrating data from Building Information Modeling (BIM) and the Internet of Things (IoT). Although these efforts are focused on the Architecture, Engineering and Construction (AEC) industry, they face challenges related to data heterogeneity and complex processing workflows. A similar concept can be applied to the monitoring of electric bicycle stations, where data from various sensors must be efficiently integrated and processed to ensure optimal operations and real-time monitoring [5].

Furthermore, the data parsing process plays a crucial role in transmitting sensor data to a central system. In Industrial Internet of Things (IIoT) devices, such as in manufacturing, parsing enables raw data from multiple sensors to be structured efficiently for further processing. A similar approach can be applied to electric bicycle charging stations, where the parsing of data from sensors such as current, voltage, and power is critical to reducing communication latency and accelerating decision-making processes [6].

Furthermore, the methods used to parse and classify large-scale texts on the Internet offer valuable information to handle large volumes of IoT data generated by electric bicycle stations. Research in the parsing and classification of large-scale data focuses on efficiently breaking down large data sets into manageable components for analysis [7]. Adopting similar techniques in IoT systems can help electric bicycle stations process vast sensor data streams more efficiently, enabling classification and prioritization of information for real-time decision making.

Embedding-based approaches, such as those used in Thai dependency parsing with character embedding, could also be adapted to handle complex sensor data. Embedding sensor data into multidimensional representations captures intricate relationships between sensor variables, such as voltage, current, and battery status, thus improving the system's ability to interpret and act on the data in real time [8]. In addition, advanced trie-based parsing algorithms can be used to optimize HTTP communication between microcontrollers (such as Arduino Nano) and web servers. By employing trie-based structures for HTTP parsing, the system can achieve faster and more efficient data transmission, thus minimizing communication overhead and improving real-time responsiveness [9]. A smart approach to electric vehicle optimization through IoT-enabled recommender systems could also be applied to electric bicycle monitoring stations [10]. Using IoT-enabled recommender systems, operators can optimize battery usage, charging patterns, and energy consumption through real-time recommendations based on sensor data and environmental conditions. This dynamic adjustment ensures maximum efficiency in bicycle operations while extending battery life and reducing energy consumption.

Furthermore, situation-aware IoT data generation, as discussed in research on the performance evaluation of IoT middleware platforms, can be applied in electric bicycle monitoring stations [11]. By generating IoT data that adapt to real-time conditions, such as changes in battery levels, user activity, or environmental factors, the system can adjust its monitoring to reflect current circumstances more accurately. This improves performance optimization and resource allocation, ensuring that the middleware efficiently processes diverse situational data inputs for system maintenance and decision making. In this regard, it is essential to consider the context features when parsing data, similar to how action parsing employs context to improve the understanding of actions in various environments [12]. In this regard, it is essential to consider the context features when parsing data, similar to how action parsing employs context to improve the understanding of actions in various environments [13]. Thus, integrating context-aware parsing, similar to the approach used in action parsing to interpret raw sensor data within specific contexts, can significantly improve the reliability and responsiveness of the system.

In this context, the use of microcontrollers such as the Arduino Nano and communication modules such as the NodeMCU ESP8266 becomes an ideal solution due to their ability to process data from various sensors and transmit them over the Internet [14]. The Arduino Nano, which is connected to various sensors, such as current sensors, voltage sensors, and power sensors, functions as a device to collect and process raw data. However, these data need to be parsed first to make them more compact and easier to transmit [15]. Data parsing is the process of breaking down raw data into manageable parts that can be sent more efficiently. This process is crucial in reducing the communication load and speeding up data transmission from the Arduino Nano to the NodeMCU ESP8266 [16]. The NodeMCU ESP8266, with its Wi-Fi connectivity, receives the parsed data from the Arduino Nano and then forwards it to a web server. Implementing this web-based system allows electric bicycle station operators to monitor the condition of the bicycles and the station in real-time from a remote location. In addition, the system can provide notifications or early warnings if there are conditions that require immediate action, such as low battery levels or system failures. Optimizing the data parsing process on the Arduino Nano and applying efficient communication protocols on the NodeMCU ESP8266 are critical steps to ensure the reliability and speed of data transmission. This not only reduces communication latency, but also minimizes network resource usage, which is essential in an Internet of Things

(IoT) environment.

Given the need for an efficient and real-time monitoring system, this study focuses on implementing data parsing between the Arduino Nano and NodeMCU ESP8266 for web-based monitoring of electric bicycle stations. This research aims to improve the reliability of the system, optimize data management, and support faster and more accurate decision making in the operation of electric bicycle stations.

2 Method

This research employs the Research and Development (R&D) method with the objective of developing and testing a new web-based monitoring system for electric bicycle charging stations. The R&D method is particularly suitable for this study, as it focuses on designing, implementing, and evaluating a system that integrates the latest technologies in data processing and transmission. The development process will begin with the collection of data from current, voltage, and power sensors connected to an Arduino Nano microcontroller. These raw data will be processed and parsed to enhance the efficiency of data transmission and storage. Subsequently, the parsed data will be transmitted to a NodeMCU ESP8266 module, which will relay the information to a web server. The monitoring system will undergo comprehensive testing to ensure data accuracy, communication efficiency, and system reliability under real-world conditions. The evaluations derived from these tests will provide critical feedback for system improvements and optimizations, ensuring effective implementation at electric bicycle stations. Using an iterative cycle of testing and evaluation, this research aims to produce a technological solution that enhances the efficiency of monitoring electric bicycle stations and supports better operational decision making through an integrated web-based system. The research stages utilized in this R&D methodology are illustrated in Figure 1.

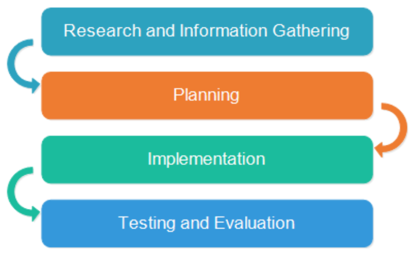


Figure 1: The research stages used in this research method

This research is carried out through several main structured stages, as illustrated in Figure 1. The first stage begins with gathering information on the problem at hand and evaluating the requirements for the devices used and data transmission techniques. This includes analyzing the hardware to be used, appropriate data transmission methods, and important parameters that need to be monitored, such as current, voltage, power, and battery status. This process aims to comprehensively understand the specific needs of the system to be developed, ensuring that each selected component can operate optimally under operating conditions. Following the information gathering, the next stage involves planning the design of the monitoring system based on the information collected. This design includes the selection of sensors (such as current, voltage, and power sensors), the microcontroller (Arduino Nano) and the communication module (NodeMCU ESP8266) that will be used to collect, process and send data to a web server. The planning phase also includes programming for data parsing to enhance the efficiency of data transmission and storage. Each step in this planning phase aims to ensure seamless integration between hardware and software. After the planning is completed, the implementation step is carried out by building a prototype of the electric bicycle charging station. This implementation includes connecting the selected sensors to the Arduino Nano, developing a data parsing program, and configuring the NodeMCU ESP8266 for data transmission to the Web server. The data collected by the sensors, such as the ACS712 (current sensor), DC voltage sensor, and PZEM004T, are parsed to allow more efficient transmission to the NodeMCU via a serial protocol. The NodeMCU then acts as a bridge between the hardware and the Internet, sending data to the designated web server. The final stage involves comprehensive testing and evaluation of the developed monitoring system. Testing is carried out under real-world conditions to ensure data accuracy, transmission efficiency, and overall system reliability. At this stage, various scenarios are applied, including simulations of low battery conditions, over current, and system failures, to determine how well the system processes and transmits information from the sensors to the web server.

Based on the results of the evaluation, improvements and optimizations are made to the system until the expected performance is achieved.

2.1 System Design

The electric bicycle charging station prototype will be equipped with several sensors to monitor various important parameters. The sensors used include three ACS712 current sensors, three DC voltage sensors, and two PZEM004T sensors. The functions of each sensor are as follows:

a. **ACS712 Current Sensors**: These three current sensors are used to monitor the current at three points:
   (1) The current from the Solar Charge Controller (SCC) output to the battery.
   (2) The current flowing from the battery to the inverter.
   (3) The current generated by the solar panel.

b. **DC Voltage Sensors**: These three voltage sensors are used to monitor the voltage at three points:
   (1) The voltage from the SCC output to the battery.
   (2) The voltage flowing from the battery to the inverter.
   (3) The voltage generated by the solar panel.

c. **PZEM004T Sensors**: These two sensors will be used to monitor various electrical parameters such as current, voltage, power, energy, frequency and power factor from two sources:
   (1) Electricity from PLN (state-owned electricity company).
   (2) Electricity from the inverter output.

The proposed system is integrated with the Internet for real-time data transmission. Data from all sensors will be sent and displayed through a web server, which will present a visualization of the monitoring data from the electric bicycle charging station. A block diagram of the proposed electric bicycle charging station system can be seen in Figure 2. Based on Figure 2, Data processing (data parsing) is performed using Arduino Mega and NodeMCU ESP8266, and then displayed on a web interface. The monitoring system for the electric bicycle charging station starts by measuring various parameters using installed sensors. The measured data are initially processed by the Arduino Mega before being sent to the NodeMCU ESP8266. The NodeMCU acts as a bridge between the hardware and the internet network, sending data to the web server. The data received by the web server are then stored in a database for further analysis. Finally, the web server displays the data in an easy-to-understand visual format, such as graphs or tables, through a web browser, allowing users to monitor the charging station's condition in real time. In this study, a serial parsing approach is applied to parse data from various sensors connected to the electric bicycle charging station. Serial parsing is chosen for its simplicity and ability to handle sequential data processing from each sensor. Each sensor, such as the voltage, current, and power sensors, sends data sequentially through the Arduino Nano, which is then processed and forwarded to the NodeMCU ESP8266 for transmission to the server. The serial parsing approach implemented in this system offers efficiency in bandwidth usage as well as the capability to handle data from sensors sequentially, minimizing the risk of data loss during transmission. This enhances the efficiency of real-time monitoring, which is a novel element compared to previous approaches that relied on parallel parsing methods in a similar system. Figure 3 shows the circuit diagram of the proposed system.

Based on Figure 3, the Arduino Mega is used as the central controller to connect various sensors and modules. The voltage sensors are connected to the analog pins of the Arduino Mega, specifically pins A0, A1, and A2, each linked to the signal pins (S) of the three voltage sensors. For current sensors, three sensors are connected to the analog pins A3, A4, and A5, with the sensor OUT pins connected to these respective pins, and the GND and VCC pins of the sensors connected to the corresponding GND and VCC pins of the Arduino. Two PZEM004T modules, used for power measurement, are connected to digital pins 10 (TX) and 11 (RX) for the first module, and pins 12 (TX) and 13 (RX) for the second module, with additional VCC and GND connections from each module to the appropriate pins on the Arduino. The BH1750 light sensor and the 20x4 LCD display, which use I2C communication, are both connected in parallel to the SDA (pin 20) and SCL (pin 21) pins of the Arduino Mega. The GND and VCC pins of these sensors and modules are also connected to the GND and VCC pins of the Arduino. The NodeMCU ESP8266, a WiFi module used for network connectivity, is connected to digital pins 2 (TX) and 3 (RX) for serial communication with the Arduino Mega, with its GND pin connected to the Arduino GND pin.

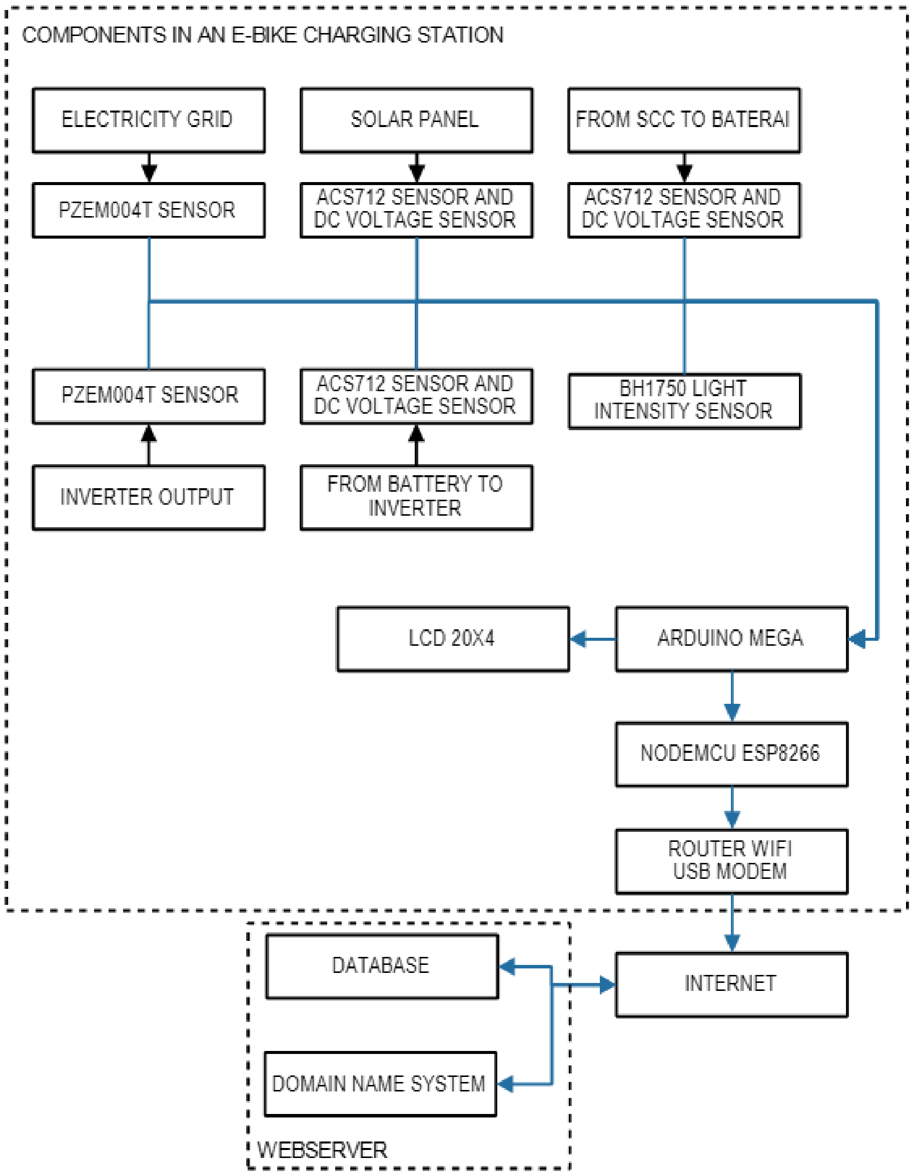


Figure 2: Block diagram of the electric bicycle charging station system.

Arduino Mega acts as a data collector for various sensors, while the NodeMCU ESP8266 is responsible for parsing and sending the data to a web server. Communication between Arduino Mega and NodeMCU ESP8266 is established through a serial protocol using the SoftwareSerial library. In the code, pin 2 (RX) and pin 3 (TX) on the Arduino Mega are designated as data communication lines with the NodeMCU. The data sent from the Arduino Mega are packaged in a string format organized using a delimiter. In this case, the chosen delimiter is a comma (,), which serves as a separator between data elements within the string. The choice of this delimiter aims to facilitate the parsing process on the NodeMCU, allowing each data element to be identified and processed more efficiently. With this format, data communication between the Arduino Mega and NodeMCU becomes more structured, enabling faster and more accurate information processing on the NodeMCU side. An example of the code implementation is as follows :

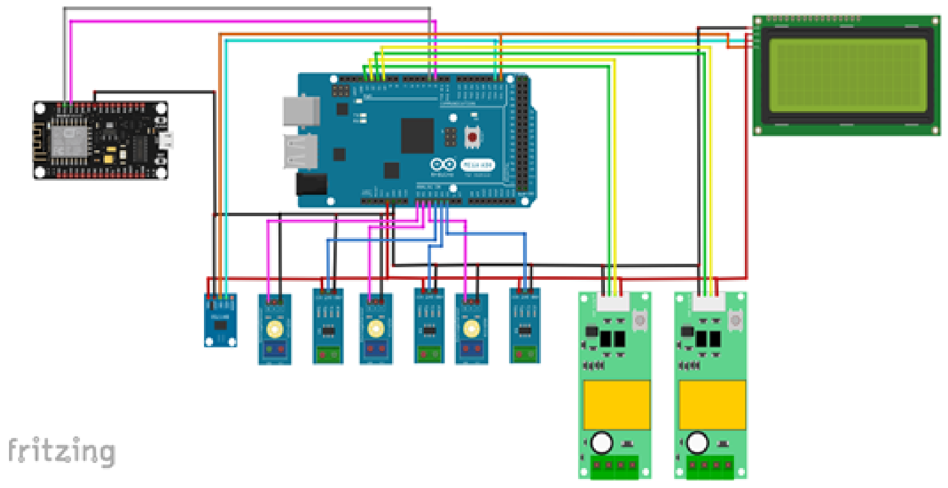


Figure 3: Wiring diagram of the E-bike charging station monitoring system design.

```
    void kirimdata3() String dataKirim = String(kalibrasitegangan1) + "," + String(
        arussensor1) + "," + String(voltage) + "," + String(current) + "," + String(
        power);
  senddata.println(dataKirim);
}
```

Listing 1: Sending data code

The next step is the data parsing process on the NodeMCU ESP8266. The NodeMCU receives the string data sent from the Arduino Mega via the serial port, where these data are first stored in a buffer to ensure that the entire data have been received completely before parsing is performed. This parsing process involves breaking down the string based on the previously specified delimiter, which is the comma (,). Each separated data element is then identified and broken down into smaller parts. Each of these parts of the data is stored in the appropriate variables, allowing the data to be accessed and processed according to the needs of the application. With this approach, the NodeMCU can manage and interpret data more effectively, ensuring that the information received can be used for decision making or device control with high precision. An example of a code implementation is as follows:

```
String receivedData = ""; // buffer data
String voltage, current, power;

void loop() {
  if (senddata.available()) {
    receivedData = senddata.readStringUntil('\n'); /
    int index1 = receivedData.indexOf(',');
    int index2 = receivedData.indexOf(',', index1 + 1);

    voltage = receivedData.substring(0, index1);
    current = receivedData.substring(index1 + 1, index2);
    power = receivedData.substring(index2 + 1);

  }
}
```

Listing 2: Parsing data code

After the data are successfully parsed from various sensors connected to the Arduino Mega, the next step is to transfer the data to the server for monitoring and further processing. The data transfer process is carried out using an HTTP GET request, which is one of the primary methods in the HTTP protocol for sending data from a client to a server. To facilitate and automate this data transfer, a specific function called kirimKeServer()

was created. This function is responsible for initiating and executing the process of sending the parsed data to the web server. Within the kirimKeServer() function, the data generated by the sensors, which has been parsed into the appropriate format, is organized into a query string that conforms to the GET request format. This query string is then sent to the server through a pre-defined URL. The use of HTTP GET requests within the kirimKeServer() function allows data to be sent quickly and efficiently, and also simplifies debugging and monitoring processes, as the sent data can be viewed directly through a browser or other monitoring tools. An example of the code implementation is as follows :

```
void kirimkeserver()  HTTPClient http;
  getData = "?data3=" + data3 + "\&data4=" + data4 + "\&data5=" + data5 + ... + "\&
      data25=" + data25;
  Link = "http://sistemmonitoringspsl.com/receive" + getData;
  http.begin(client, Link);

  int httpCode = http.GET();
  String payload = http.getString();
  Serial.println(httpCode);
  Serial.println(payload);
  http.end();
```

Listing 3: Server sending data code

The success of the parsing process is crucial because it determines the accuracy of the data sent to the web server for monitoring purposes. Properly parsed data ensure that every piece of information received by the server is valid and can be processed correctly. Once the data are received, the web server stores and displays them in an easily understandable data visualization format, as shown in Figure 3. In this figure, data from the monitoring devices are presented in an interactive dashboard that includes information about voltage, current, power, energy, and the status of various resources, such as photovoltaic (PV), batteries, PLN power source, and inverters. This visualization helps users monitor system performance in real time and make quick decisions based on current operational conditions. To enhance novelty, this system also implements an adaptive parsing method, in which the parsing algorithm dynamically adjusts the parsing method based on network conditions. When the system detects high latency, the data transmission frequency is automatically reduced to lessen the network load without compromising the overall accuracy of the monitoring data. Figure 4 shows the interface of the web server.

3 Result and Discussion

3.1 Data parsing failure

In the testing of the web-based E-Bike charging station monitoring system, several failures were identified in the data parsing process between the Arduino Mega and NodeMCU ESP8266. These failures occurred primarily during data transmission from the Arduino Mega to the NodeMCU ESP8266 via serial communication. An example of a parsing failure is shown in Figure 5.

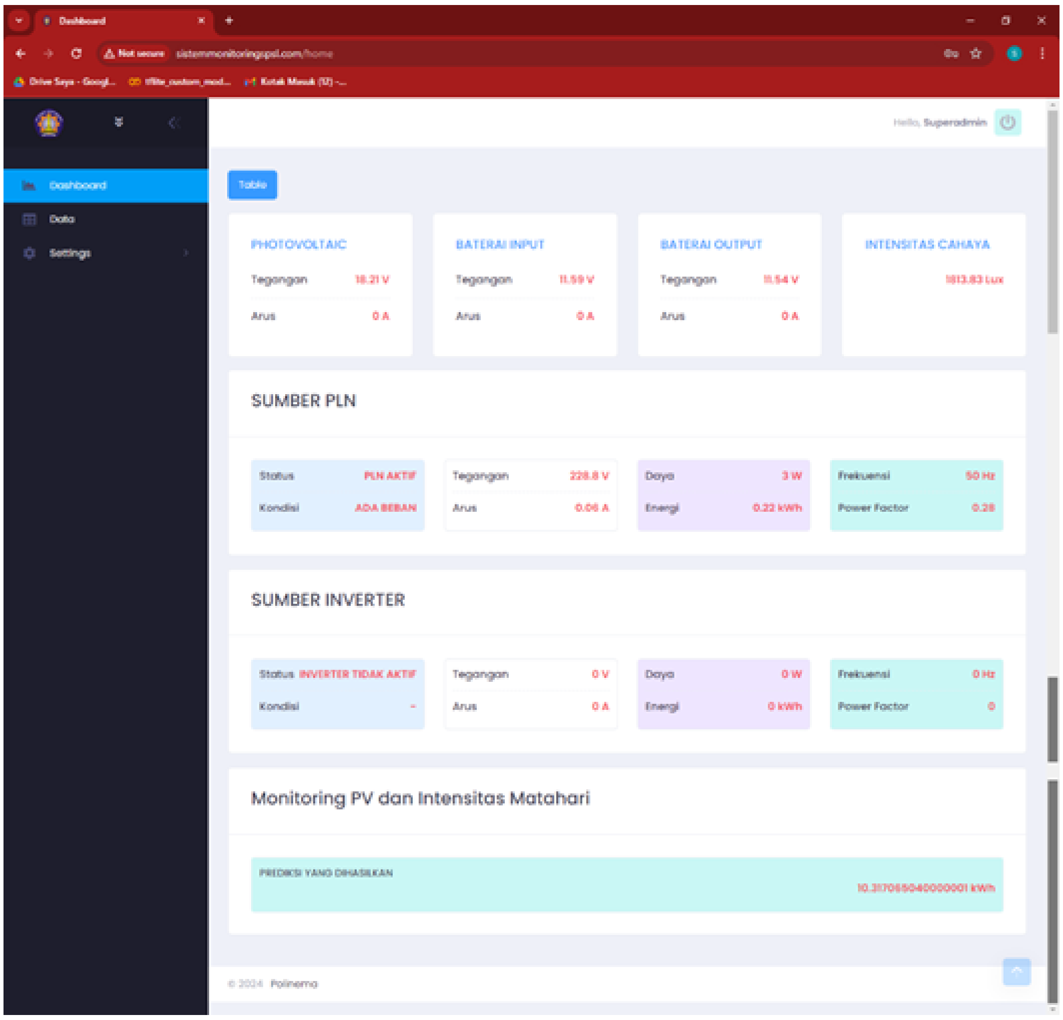
The following are the specific findings identified during the testing:

(1) **Data Loss During Transmission**
There were several instances where data sent from the Arduino Mega was not completely received by the NodeMCU ESP8266. This occurred due to interference in the serial connection or a full communication buffer on the NodeMCU. As a result, the data received by the NodeMCU did not match the expected format, leading to parsing errors.

(2) **Parsing Errors Due to Corrupted Data**
Parsing errors often occurred when the data received by the NodeMCU ESP8266 did not match the desired format. For example, when the delimiter character used to separate each data element was not properly detected, the parsing process produced incorrect values or failed to extract the data entirely. This error was caused by signal interference or mismatched baud rates between the two devices.

3.2 The Impact of Data Transmission Delays

In the testing of the Web-based E-Bike charging station monitoring system, experiments were conducted to measure the impact of applying delays in data transmission between Arduino Mega and NodeMCU ESP8266.



PHOTOVOLTAIC

Tegangan	18.21 V
Arus	0 A

BATERAI INPUT

Tegangan	11.59 V
Arus	0 A

BATERAI OUTPUT

Tegangan	11.54 V
Arus	0 A

INTENSITAS CAHAYA

1813.83 Lux

SUMBER PLN

Status	PLN AKTIF
Kondisi	ADA BEBAN
Tegangan	228.8 V
Arus	0.06 A
Daya	3 W
Energi	0.22 kWh
Frekuensi	50 Hz
Power Factor	0.28

SUMBER INVERTER

Status	INVERTER TIDAK AKTIF
Kondisi	-
Tegangan	0 V
Arus	0 A
Daya	0 W
Energi	0 kWh
Frekuensi	0 Hz
Power Factor	0

Monitoring PV dan Intensitas Matahari

PREDIKSI YANG DIHASILKAN 10.317065040000001 kWh

© 2024 Polinema

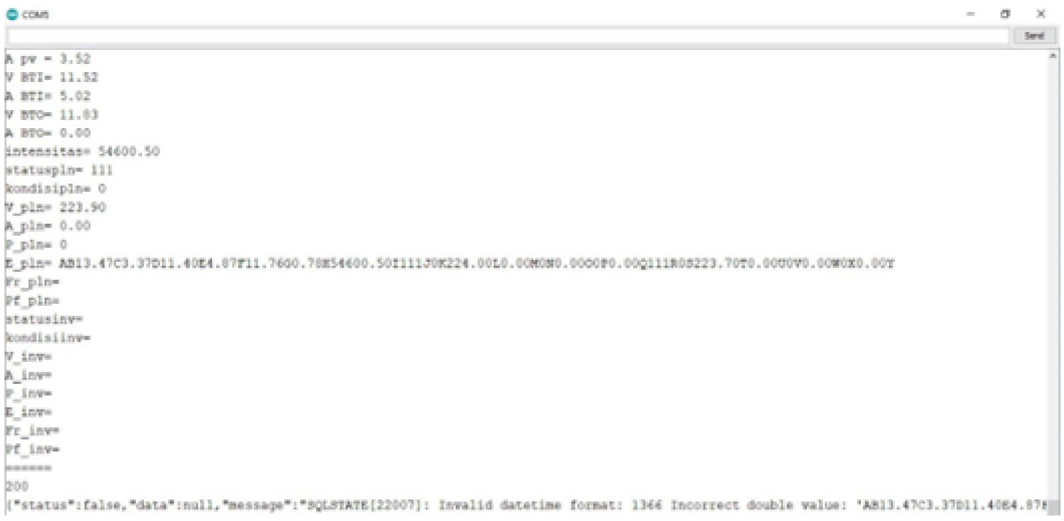
Figure 4: Web server interface visualizing the data received for monitoring electric bicycle stations.

The aim of this experiment was to determine the optimal delay configuration to reduce communication latency, prevent data loss, and ensure efficient use of network resources. Several delay scenarios were applied to the Arduino Mega code before transmitting the data to the NodeMCU ESP8266. The delays ranged from 0 ms (without delay), 100 ms, 200 ms, to 500 ms. The test was carried out by sending data from various sensors (ACS712, DC voltage sensor, PZEM004T) at different time intervals, monitoring the impact on data transmission speed, accuracy, and network utilization. The test data is presented in Table 11.

Table 1: Test Results on the Impact of Data Transmission Delays

Delay (ms)	Data loss	Communication Latency (ms)	Data Transmission	System responsiveness
0	15%	High	Very High	Very low, frequent data loss
100	10%	High	High	Low
200	<2%	Stable (50 ms)	Optimal	Optimal for real-time monitoring
500	<1%	Stable	Low	Slow, less responsive

Testing also revealed that with a longer delay (500 ms), the utilization of network bandwidth became more efficient. The transmitted data placed less stress on the Wi-Fi network and the use of network resources, such as buffers and memory on the NodeMCU ESP8266, was reduced by up to 30%. However, despite the lower



```
A pv = 3.52
V BTI= 11.52
A BTI= 5.02
V BTO= 11.83
A BTO= 0.00
intensitas= 54600.50
statuspln= 111
kondisipln= 0
V_pln= 223.90
A_pln= 0.00
P_pln= 0
E_pln= AB13.47C3.37D11.40E4.87F11.76G0.78H54600.50I111J0K224.00L0.00M0N0.00O0P0.00Q111R0S223.70T0.00U0V0.00W0X0.00Y
Fr_pln=
Pf_pln=
statusinv=
kondisiinv=
V_inv=
A_inv=
P_inv=
E_inv=
Fr_inv=
Pf_inv=
======
200
{"status":false,"data":null,"message":"SQLSTATE[22007]: Invalid datetime format: 1366 Incorrect double value: 'AB13.47C3.37D11.40E4.87F
```

Figure 5: Example of a parsing failure in the data received by NodeMCU ESP8266.

resource usage, the system's responsiveness to changes in conditions became slower. In contrast, with delays of 0 or 100 ms, the usage of network resources increased by up to 45% compared to delays of 200 or 500 ms, leading to a higher risk of buffer overflow and data loss. This indicates that overly short delays are not ideal for applications requiring stable and reliable data transmission.

3.3 Delay in Data Reception on the Web Server

Testing was conducted to measure the data reception delay on the Web server after the data were sent by the NodeMCU ESP8266. The results indicated that several factors influenced this delay, such as the amount of data transmitted, the processing capacity of the Web server, and the efficiency of the network. The test results are presented in Table 2

Table 2: Test Results of Data Reception Delays on the Web Server

Transmission delay	Average Reception delay (ms)	Maximum fluctuation (ms)
0 ms	120	180
100 ms	90	150
200 ms	30	Minimal
500 ms	20	Almost no fluctuation

Based on Table 2, as the data transmission frequency increases (with 0 ms and 100 ms transmission delays from the NodeMCU), there is a significant rise in data reception delays on the web server. In this test, the average data reception delay reached 120 ms with fluctuations up to 180 ms during data surges for a 0 ms transmission delay. For a 100-ms transmission delay, the average reception delay was 90-ms, with fluctuations up to 150-ms. When examining the impact of the processing capacity of the web server, the lower transmission frequencies (200 ms and 500 ms) resulted in more stable reception delays on the web server. With a 200 ms transmission delay, the average reception delay was 30 ms with minimal fluctuation, and for a 500 ms transmission delay, the average delay was 20 ms with almost no fluctuation. In addition, the efficiency of the network and connection conditions was tested to assess their effect on data reception delays. Under stable and low-latency network conditions, the average reception delay was 20-30 ms. However, in high-latency or unstable network environments, reception delays increased to between 120 and 200 ms.

Data reception delays on the web server affect the update of data displays on the monitoring dashboard. When reception delays exceed 100 ms, there is a noticeable lag in the information updates visible to users, particularly under less stable network conditions. However, with an average reception delay of 30 ms or less, the system is able to display data nearly in real time, allowing users to respond more quickly and accurately to emergency conditions.

4 Conclusion

The conclusions of this research indicate that optimizing a web-based monitoring system for electric bicycle charging stations can be achieved through appropriate data transmission delay configurations and the use of an efficient serial parsing method. With an optimal transmission delay of 200 ms, data loss was significantly reduced to below 2%, while communication latency stabilized at 50 ms, ensuring that the system remains responsive under real-time monitoring conditions. Furthermore, higher transmission delays enhanced network resource utilization efficiency, although it slightly reduced system responsiveness. The test results also showed that lower transmission delays increased data reception latency at the web server, whereas the optimal delay ensured more stable and nearly real-time data reception. Overall, this study contributes to the development of a more efficient and reliable monitoring system, with broader implications for the application of Internet of Things (IoT) technology in real-time monitoring systems.

References

[1] N. P. Nugroho. (2024) Populasi kendaraan listrik meningkat pesat, roda dua naik 262 persen. Accessed: 03-Sep-2024. [Online]. Available: https://bisnis.tempo.co/read/1839797/populasi-kendaraan-listrik-meningkat-pesat-roda-dua-naik-262-persen

[2] A. Yulianti. (2023) Maraknya sepeda listrik di indonesia? Accessed: 03-Sep-2024. [Online]. Available: https://serbasepeda.com/blogs/popularitas-sepeda-listrik/

[3] T. Bieliński, A. Kwapisz, and A. Ważna, "Electric bike-sharing services mode substitution for driving, public transit, and cycling," *Transportation Research Part D: Transport and Environment*, vol. 96, 2021.

[4] S. Dasi, S. M. Kuchibhatla, M. Ravindra, K. S. Kumar, S. S. Chekuri, and A. K. Kavuru, "Iot-based smart energy management system to meet the requirements of ev charging stations," *Journal of Theoretical and Applied Information Technology*, vol. 102, no. 5, pp. 2116–2127, 2024.

[5] X. Huang, Y. Liu, L. Huang, E. Onstein, and C. Merschbrock, "Automation in construction bim and iot data fusion: The data process model perspective," *Automation in Construction*, vol. 149, p. 104792, 2023.

[6] R. Mharaj, V. Balyan, and M. T. Kahn, "Design of iiot device to parse data directly to scada systems using lora physical layer," *International Journal of Smart Sensing and Intelligent Systems*, vol. 15, no. 1, 2022.

[7] M. Song, H. Zheng, Z. Tao, J. Jiang, and B. Pan, "Research on methods of parsing and classification of internet super large-scale texts," in *Proceedings of an International Conference*, 2021.

[8] S. Singkul and K. Woraratpanya, "Thai dependency parsing with character embedding," in *11th International Conference on Information Technology and Electrical Engineering (ICITEE)*, vol. 7, 2019, pp. 1–5.

[9] A. Li, D. He, and H. Wang, "An advanced trie-based http parsing algorithm," in *Sixth International Conference on Information Science and Technology*, 2016.

[10] P. Amudhavalli, R. Zahira, S. Umashankar, and X. N. Fernando, "A smart approach to electric vehicle optimization via iot-enabled recommender systems," *Technologies*, vol. 12, no. 137, pp. 1–21, 2024.

[11] S. Mondal, P. P. Jayaraman, P. D. Haghighi, A. Hassani, and D. Georgakopoulos, "Situation-aware iot data generation towards performance evaluation of iot middleware platforms," *Sensors*, vol. 23, no. 7, pp. 1–32, 2023.

[12] D. Evans, "Energy-efficient transaction serialization for iot devices," *Journal of Computer Science Research*, vol. 2, no. 2, pp. 1–16, 2020.

[13] N. Mehrseresht, "Action parsing using context features," in *International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, 2017.

[14] F. Zaro, A. Tamimi, and A. Barakat, "Smart home automation system," *International Journal of Engineering and Innovative Research*, vol. 3, no. 1, pp. 66–88, 2021.

[15] F. Palaha, E. Ermawati, M. Machdalena, and E. H. Arya, “Analisa traffic data esp8266 pada kontrol dan monitoring daya listrik menggunakan aplikasi blynk berbasis arduino nano,” *Jurnal Nasional Komputasi dan Teknologi Informasi*, vol. 4, no. 6, pp. 480–489, 2021.

[16] A. Herlan, I. Fitri, and R. Nuraini, “Rancang bangun sistem monitoring data sebaran covid-19 secara real-time menggunakan arduino berbasis internet of things (iot),” *Jurnal Teknologi dan Informasi*, vol. 5, no. 2, pp. 206–212, 2021.