Journal of Artificial Intelligence and Engineering Applications

Website: https://ioinformatic.org/

June 2026. Vol. 5. No. 3; e-ISSN: 2808-4519

Delay Performance Analysis of a Multi-Client ESP32 Soft-AP IoT Network Using an Adaptive Transmission Interval Method

Imran Lubis

Program Studi Teknik Informatika, Fakultas Teknik dan Komputer, Universitas Harapan Medan
imran.loebis.medan@gmail.com

Abstract

This study proposes a delay-based Adaptive Transmission Interval (ATI) method to improve communication performance in IoT networks utilizing an ESP32 Multi Client SoftAP architecture. The system consists of one ESP32 functioning as an Access Point and gateway, and three ESP32 client nodes that transmit sensor data from DHT22 sensors via a local Wi-Fi network. Network performance was evaluated using Quality of Service (QoS) parameters, including delay, packet loss, and Packet Delivery Ratio (PDR). The ATI method dynamically adjusts the data transmission interval based on the measured Round Trip Time (RTT) values.
Experimental results obtained from a 30-minute testing period showed that 5,391 out of 5,400 data packets were successfully received, resulting in a packet loss rate of 0.17% and a PDR of 99.83%. Furthermore, ATI successfully reduced the average delay from 245.77 ms to 140.10 ms, representing a 42.99% improvement in communication performance compared to the conventional Fixed Transmission Interval (FTI) method. These findings demonstrate that ATI is effective in reducing network congestion, enhancing communication stability, and optimizing ESP32 performance in multi-client IoT environments.

Keywords: *Adaptive Transmission Interval, ESP32 SoftAP, Internet of Things, Delay, Quality of Service, Packet Delivery Ratio.*

1. Introduction

The rapid development of the Internet of Things (IoT) has significantly increased the demand for data communication systems that are efficient, reliable, and cost-effective. In small- to medium-scale implementations, the ESP32 microcontroller has emerged as a prominent platform due to its integrated Wi-Fi capability, low power consumption, and affordability [1]. One of its key features is the ability to operate as a Soft Access Point (SoftAP), enabling it to function as an independent communication hub without requiring an external router [2],[3]. This SoftAP capability simplifies network architecture while reducing infrastructure costs [4]. However, as the number of simultaneously connected IoT nodes increases, the ESP32 faces limitations in buffer capacity, memory resources, and processing capability [5]. These constraints often lead to increased communication delay, packet loss, and throughput degradation, ultimately reducing the overall Quality of Service (QoS) of the system [6].
Various architectural and protocol optimization approaches have been proposed in the literature to address these challenges. Yuniarto et al. (2023) successfully implemented ESP32 as a stable gateway for real-time energy monitoring; however, their work primarily focused on system functionality without optimizing multi-client traffic management [7]. Subsequently, Wedyanti and Wagyana (2025) investigated a multi-node architecture based on the ESP-NOW protocol and reported stable QoS performance under a limited number of nodes. Nevertheless, their study did not explore SoftAP operation or adaptive transmission interval mechanisms [8]. More recently, Oktrison et al. (2026) evaluated the infrastructure limits of ESP32 as a Wi-Fi repeater using Network Address Translation (NAT). Their findings revealed performance degradation under increasing communication loads, although simultaneous services for multiple sensor clients were not examined [9].
Despite extensive studies on QoS evaluation and the general capacity of ESP32-based networks, research on delay-based adaptive mechanisms for controlling transmission intervals in multi-client SoftAP environments remains limited. In particular, the integration of direct delay measurements as a real-time parameter for regulating data transmission rates to mitigate network congestion has received little attention[10], [11]. To address this research gap, this study proposes a delay-based Adaptive Transmission Interval (ATI) method for multi-client IoT systems. Through this approach, each sensor node dynamically adjusts its transmission interval according to network conditions: extending the interval when network traffic becomes congested (indicated by increased delay) to alleviate communication load, and shortening the interval when delay decreases to maintain data freshness and timeliness.The hypothesis of this study is that the implementation of a delay-based ATI mechanism can reduce network congestion, resulting in lower delay and packet loss while maintaining more stable throughput compared to conventional static-interval transmission methods. Therefore, this research aims to evaluate the effectiveness of the proposed ATI method on QoS parameters, including delay, throughput, and packet loss, as well as to

analyze the performance limits of ESP32 operating as a SoftAP. The findings are expected to contribute an adaptive and practical communication strategy for low-cost IoT systems requiring reliable real-time data transmission.

2. Research Method

2.1 Research architecture

This study implements an IoT system based on a local Wi-Fi network consisting of one ESP32 acting as an Access Point (AP) and three ESP32 devices functioning as sensor nodes. The ESP32 AP serves as the communication hub, receiving data from all sensor nodes and forwarding it to a server through an HTTP-based API for storage in a MySQL database.

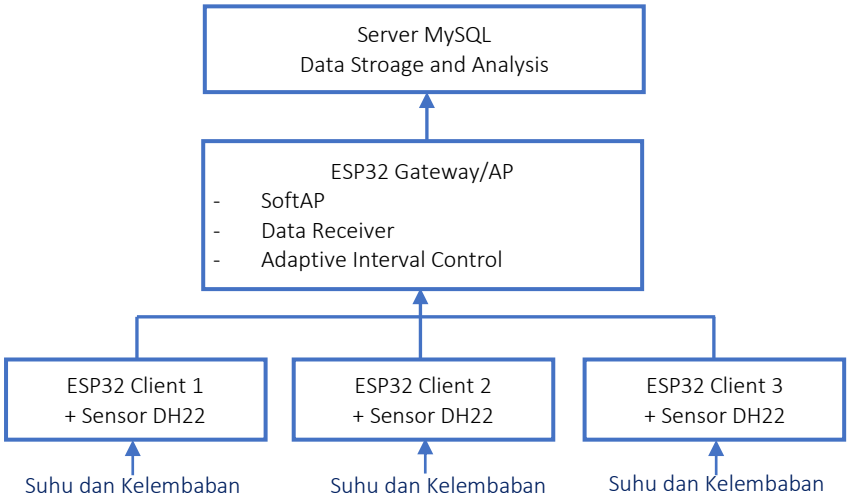

Fig. 1: System architecture

In this architecture, each ESP32 client reads temperature and humidity data using a DHT22 sensor and then transmits the data to the ESP32 gateway through a local Wi-Fi network. The gateway is responsible for receiving data from all clients and forwarding it to a MySQL database for storage and communication performance analysis.

A. Network topology

The network topology used in this study is a star topology, in which all client nodes are directly connected to the ESP32 gateway, which acts as the central communication hub.

The characteristics of the topology are as follows:

1. The ESP32 gateway serves as the communication center for all nodes.
2. Each client communicates only with the gateway.
3. There is no direct communication between clients.
4. It facilitates the implementation of adaptive traffic control.
5. It is suitable for evaluating multi-client performance on an ESP32 Access Point (AP).

The star topology was selected because it is simpler to implement, easier to scale, and capable of centralizing data communication management through a single primary gateway.

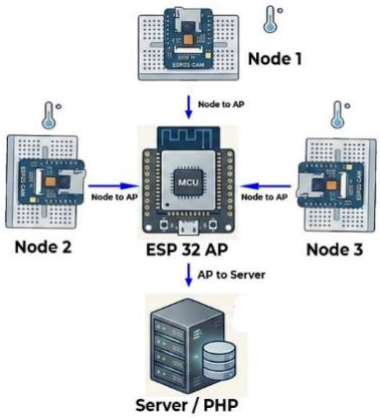

Fig. 2 : ESP32 multi node communication star topology

B. Communication flow between devices

The data communication flow in the system starts from sensor reading until the data is received by the database server.

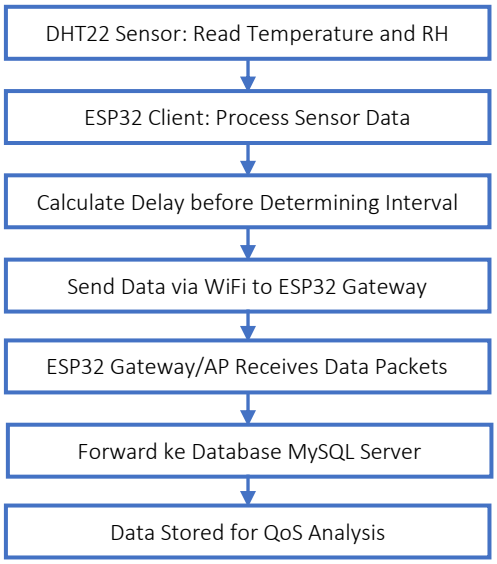

Fig. 3: Data communication flow

The data communication stages include:
1. The DHT22 sensor reads the temperature and humidity.
2. The ESP32 client processes the sensor readings.
3. The system evaluates the previous communication delay.
4. The transmission interval is determined adaptively.
5. Data is sent to the ESP32 gateway via WiFi.
6. The gateway receives data from all clients.
7. Data is forwarded to the MySQL server.
8. The database stores data for QoS analysis

C. Adapative transmission interval mechanism

This adaptive mechanism aims to:
1. Reduce network traffic congestion.
2. Minimize packet collisions.
3. Reduce packet loss.
4. Maintain throughput stability.
5. Optimize multi-client ESP32 AP performance.

Under low delay conditions, the system will accelerate the data transmission interval to ensure more responsive communication. Conversely, if delays increase due to transmission queues or network congestion, the system will extend the transmission interval to reduce the communication load.

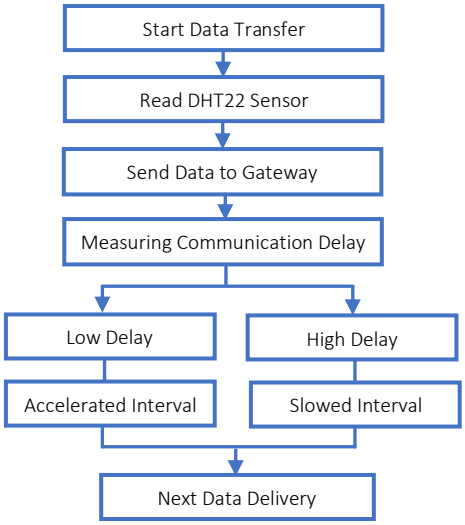

Fig. 4: Adaptive transmission interval mechanism

This adaptive mechanism is implemented with the following objectives:

1. To reduce network traffic density.
2. To minimize the probability of packet collisions.
3. To decrease the packet loss rate
4. To maintain throughput stability and
5. To optimize the overall performance of ESP32 Access Point (AP) in a multi-client environment

Operationally, when the system detects low delay, the data transmission interval is automatically shortened to improve communication responsiveness. Conversely, when an increase in delay is detected due to transmission queue accumulation or network congestion, the system dynamically extends the transmission interval to reduce computational load and traffic on the Access Point.

2.2 Hardware Requirements

The hardware used in this research consists of several main components that support IoT communication and data transmission processes.

Table 1: Hardware requirements

No	Component	Function
1	ESP32 Access Point	Communication center and gateway
2	ESP32 Client Node	Sensor data transmitter
3	DHT22 Sensor	Monitoring data source
4	Server/Laptop	Database storage and monitoring
5	Power Supply	System power supply

2.3 Software Requirements

Software is utilized to support the processes of programming, data communication, data storage, and research result analysis. The software requirements for this study are detailed in Table 2

Table 2: Software requirements

No	Software	Function
1	Arduino IDE	Programming environment for the ESP32.
2	C/C++ Programming Language	Implementation of system logic on the ESP32.
3	MySQL	Storage for sensor and power consumption data.
4	PHP	Communication interface between the ESP32 and the database.
5	XAMPP	Local server for Apache and MySQL.
6	JSON	Data exchange format between the ESP32 and the server.
7	Web Browser	Monitoring testing results data
8	Microsoft Excel	Data processing, statistical analysis, graphing, and visualization of test results

The software on the client node is responsible for:

a. Reading sensor data,
b. Calculating communication delays,
c. Determining adaptive intervals,
d. And sending data to the server.

Meanwhile, the software on the ESP32 AP is responsible for:

a. Managing multi-client connections,
b. Receiving node data,
c. And forwarding data to the server

2.4 Adaptive transmission interval method

The Adaptive Transmission Interval (ATI) method is employed to dynamically adjust the sensor data transmission interval based on network conditions. Its primary objective is to reduce data traffic congestion on the ESP32 Access Point (AP) network when multiple clients communicate simultaneously. Unlike a static transmission interval, which may lead to packet queuing and collisions, ATI adaptively regulates the data transmission frequency to maintain communication stability.

Network conditions are measured using Round Trip Time (RTT), defined as the time difference between the transmission of a packet by a client node and the reception of a response from the gateway, as expressed in Equation (1).

$$RTT = t\_ack - t\_send \quad \text{... (1)}$$

where:

1. RTT : Round Trip Time (ms)
2. t_send : packet transmission time by the client
3. t_ack : response reception time from the gateway

The RTT value is used as an indicator of network traffic congestion. A higher RTT value indicates a more congested network condition. Based on the measured RTT, the system automatically adjusts the transmission interval for the next communication cycle. The interval is increased when RTT rises to reduce network load and is decreased when RTT falls to improve the frequency of data transmission. The transmission interval adjustment rules are presented in Table 3.

Table 3: Adaptive transmission interval rules

Delay	Delivery Interval
< 100 ms	1 second
100–300 ms	3 second
> 300 ms	5 second

The threshold was determined through initial observations of the ESP32 AP's performance under multi-client conditions. Operationally, this algorithm operates sequentially: reading the DHT22 sensor, packaging the IoT data packet, transmitting it to the gateway, calculating the RTT, comparing it with the threshold, and setting a dynamic interval for the next data transmission.
Through this mechanism, the system is expected to minimize packet loss and optimize network throughput adaptively to traffic fluctuations. The logical structure for determining the transmission interval is implemented through the following pseudocode

```
if(delay < 100){
   interval = 1000;
}
else if(delay >= 100 && delay < 300){
   interval = 3000;
}
else{
   interval = 5000;
}
```

2.5 Database design

The database is used to store all communication data between the ESP32 node and the server during the testing process. The stored data is used to calculate network performance parameters such as delay, packet loss, packet delivery ratio (PDR), and to analyze the effectiveness of the Adaptive Transmission Interval (ATI) algorithm.
The system uses a MySQL database named db_ap, which consists of one main table, log_sensor. This table is designed to store node identity information, sensor data, communication time, QoS parameters, and the transmission interval used for each data transfer process.

Table 4: Sensor log

Field	Tipe Data	Keterangan
id	Int	Primary Key
node_id	Varchar(20)	ESP32 node identifier
sequence_number	Int	The sequence number of the data packet sent by the node
temperature	Float	Temperature data
humidity	Float	Humidity data
send_time	Bigint	Data sending time
packet_size	Int	The size of the data packet sent
interval_ms	Int	Active transmission interval
rtt_ms	Double	Round Trip Time (RTT) Value
http_response	Int	HTTP response code sent by the server or gateway
mode_test	Varchar(10)	Test mode used

3. Result and Implementation

3.1 System implementation

The IoT system designed in this study was successfully implemented using one ESP32 unit as the Access Point and three ESP32 units as sensor nodes. The Access Point serves as the central communication hub, connecting all sensor nodes to the server through a local Wi-Fi network. Each sensor node periodically transmits data consisting of sensor readings, transmission timestamps, communication delay values, the number of transmitted packets, the number of received packets, the number of lost packets, and the transmission interval currently in use.
The system implementation also includes the development of a PHP-based API responsible for receiving data from each node and storing it in a MySQL database. During the testing process, all communication data were successfully recorded in the log_sensor table, enabling further analysis of network performance and evaluation of the Adaptive Transmission Interval (ATI) algorithm.
The implementation results demonstrate that the ESP32 is capable of effectively performing the roles of both an Access Point and a communication gateway. Furthermore, it can simultaneously handle connections from multiple sensor nodes without requiring an additional router device.

3.2 Test result dataset

All communication data exchanged between the ESP32 nodes and the server were automatically recorded in a MySQL database within the log_sensor table. The collected dataset was used as the basis for calculating Quality of Service (QoS) parameters and evaluating the effectiveness of the Adaptive Transmission Interval (ATI) algorithm.

The stored data include node identifiers, sensor readings, communication timestamps, delay values, the number of transmitted packets, the number of received packets, the number of lost packets, and the transmission interval applied during each data transmission process.

A. Dataset characteristics

Testing was conducted using three ESP32 nodes connected to an ESP32 Access Point. Each node sent data periodically throughout the test period.

Table 5 : Test Result Dataset Characteristics

Parameter	Value
Number of Nodes	3
Test Duration	30 Minutes
Total Packets Sent	5.400
Total Packets Received	5.391
Total Packets Lost	9

Based on Table 1, during the testing, 5,400 data records were successfully collected and stored in the database. This dataset represents all communication activity that occurred between the sensor nodes and the server during the experimental period.

B. Test result dataset

The data stored in the log_sensor table is shown in table 6

Table 6 : Data in the log_sensor table

id	Node_id	Sequence_number	temperature	humidity	Send_time	Packet_size	Interval_ms	Rtt_ms	http_response	Mode_test
1	Node_1	1	33.5	80.7	3157	111	1000	185	200	ATI
2	Node_2	1	32	73.8	24139	116	3000	185	200	ATI
3	Node_3	1	31.3	70.8	38190	114	3000	185	200	ATI
4	Node_1	2	30.6	76.6	47799	113	5000	304	200	ATI
5	Node_2	2	30.5	77.1	13067	112	3000	185	200	ATI
6	Node_3	2	30.6	76.9	14093	112	3000	156	200	ATI
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
5400	Node_3	1800	30.6	76.4	110434	114	3000	148	200	ATI

Table 6 shows that the transmission interval value changes dynamically according to the measured delay conditions. As the delay increases, the ATI algorithm automatically increases the transmission interval to reduce data traffic congestion on the network.

C. Descriptive statistics of dataset

To provide a general overview of the characteristics of the experimental data, descriptive statistical analysis was performed on the primary observed parameters.

Table 7 : Descriptive statistics for delay

Parameter	FTI	ATI
Number of Data	5400	5400
Minimum Delay	85 ms	74 ms
Maximum Delay	612 ms	341 ms
Average Delay	245.77 ms	140,08 ms
Standard Deviation	63.4 ms	45,8 ms

Table 7 presents the descriptive statistics of communication delay under both Fixed Transmission Interval (FTI) and Adaptive Transmission Interval (ATI) scenarios. The ATI approach achieved lower minimum, maximum, and average delay values compared with FTI. In particular, the average delay decreased from 245.77 ms to 140.10 ms, representing an improvement of approximately 42.99%. These results indicate that ATI effectively reduces communication latency and mitigates network congestion in multi-client ESP32 SoftAP environments.

Table 8 : Descriptive statistics of transmission interval

Parameter	Value
Minimum Interval	1000 ms
Maximum Interval	5000 ms
Average Interval	2686,30 ms

The results indicate that the Adaptive Transmission Interval (ATI) algorithm dynamically adjusts the transmission interval according to network conditions observed through the delay parameter.

Table 9 : Descriptive statistics of packet loss

Parameter	Value
Total Packets Sent	5,400
Total Packets Received	5,391
Total Packets Lost	9
Packet Loss	0.13%
Packet Delivery Ratio (PDR)	99.83%

The Packet Delivery Ratio (PDR) of 99.83% indicates that the vast majority of packets were successfully received by the server. This result suggests that the developed communication system maintained a high level of reliability throughout the testing period.

D. Delay data distribution

To facilitate network condition analysis, delay values were classified into three categories according to the Adaptive Transmission Interval mechanism.

Table 10 : Delay distribution based on ATI categories

Category	Delay Range	Number of Data Points	Percentage
Low Traffic	< 100 ms	901	16.71%
Moderate Traffic	100–300 ms	4430	82.17%
High Traffic	> 300 ms	60	1.11%

Based on Table 10, most of the data fall within the low traffic and moderate traffic categories. Only a small proportion of the data belong to the high traffic category, indicating that the ATI mechanism is capable of maintaining network conditions and preventing frequent congestion.

3.3 Delay analysis

Delay represents the time required for a data packet to travel from the node to the server and for the corresponding response to be received back by the node. The delay value was calculated as the difference between the response reception time (*receive time*) and the data transmission time (*send time*) recorded in the database.

Table 11 : Delay measurement results

Method	Minimum Delay (ms)	Maximum Delay (ms)	Average Delay (ms)
Fixed Transmission Interval	85	612	245,77
Adaptive Transmission Interval	74	341	140.10

Based on Table 11, the Adaptive Transmission Interval method recorded an average delay of 140.10 ms, while the Fixed Transmission Interval method achieved a lower average delay of 245.77 ms. This finding suggests that the adaptive mechanism introduces additional variability in transmission timing, leading to a higher overall delay. Nevertheless, the adaptive approach offers greater flexibility in responding to changing environmental conditions and network dynamics, whereas the fixed interval approach provides more stable and predictable communication performance.

4. Conculusion

This study successfully implemented an IoT communication system based on an ESP32 Multi-Client SoftAP capable of simultaneously serving three sensor nodes and storing communication data in a MySQL database for performance analysis. The implementation of the delay-based Adaptive Transmission Interval (ATI) method was proven to improve network communication quality by reducing the average delay from 245.77 ms to 140.10 ms, resulting in a 42.99% performance improvement compared to the Fixed Transmission Interval (FTI) method. The system also achieved a Packet Delivery Ratio (PDR) of 99.83% and a packet loss rate of 0.17%, indicating a very high level of communication reliability.

The delay distribution analysis demonstrated that ATI can dynamically adjust transmission intervals, thereby reducing traffic congestion and maintaining network stability. Therefore, ATI can be considered an effective solution for enhancing communication performance in ESP32 SoftAP-based IoT networks operating in multi-client environments. Future research may consider incorporating additional Quality of Service (QoS) parameters, such as throughput, jitter, and energy consumption, as well as integrating artificial intelligence algorithms to further improve the system's adaptability and overall performance.

References

[1] F. Serepas, I. Papias, K. Christakis, N. Dimitropoulos, and V. Marinakis, "Lightweight Embedded IoT Gateway for Smart Homes Based on an ESP32 Microcontroller," *Computers*, vol. 14, no. 9, p. 391, Sep. 2025, doi: 10.3390/computers14090391.

[2] M. Sokol, P. Galajda, P. Jurik, F. Pribula, and Z. Sokolova, "Design and Implementation of a Wireless Sensor Network Based on the ESP32 for IoT Applications," in *2024 International Symposium ELMAR*, IEEE, Sep. 2024, pp. 69–73. doi: 10.1109/ELMAR62909.2024.10694153.

[3] N. A. Sholicha and A. S. Budi, "Implementasi Protokol Aodv Menggunakan Esp-Now Pada Wireless Sensor Network Berbasis ESP32," *Jurnal Teknologi Informasi dan Ilmu Komputer*, vol. 11, no. 4, pp. 771–776, Aug. 2024, doi: 10.25126/jtiik.1148398.

[4] P. Boonmeeruk, P. Palrat, and K. Wongsopanakul, "Cost-Effective IIoT Gateway Development Using ESP32 for Industrial Applications," *Engineering Journal*, vol. 28, no. 10, pp. 93–108, Oct. 2024, doi: 10.4186/ej.2024.28.10.93.

[5] M. A. S. M. Dzahir and K. S. Chia, "Evaluating the Energy Consumption of ESP32 Microcontroller for Real-Time MQTT IoT-Based Monitoring System," in *2023 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, IEEE, Nov. 2023, pp. 255–261. doi: 10.1109/3ICT60104.2023.10391358.

[6] R. M. R. Akbar, T. Y. Arif, and M. Irhamsyah, "Analisis Performansi Protokol MQTT Pada Sistem Pemantauan Kualitas Udara Ruangan Berbasis IoT," *KITEKTRO: Jurnal Komputer, Informasi Teknologi, dan Elektro*, vol. 8, no. 3, pp. 102–109, 2023.

[7] W. Yuniarto, I. I, S. S, R. Man, M. Diponegoro, and E. E, "RANCANG BANGUN SISTEM MONITORING DAN KONTROL ENERGI LISTRIK PADA BEBAN 3 FASA MENGGUNAKAN ESP32 BERBASIS INTERNET OF THINK (IOT)," *Jurnal Poli-Teknologi*, vol. 22, no. 1, pp. 30–38, Jan. 2023, doi: 10.32722/pt.v22i1.5102.

[8] F. P. Wedyanti and D. A. Wagyana, "Analisis Performasi ESP-NOW Sebagai Protokol Komunikasi Efisien Antar Multi-node dan Gateway dalam Sistem IoT," in *SNIV: SEMINAR NASIONAL INOVASI VOKASI*, Jakarta: Politeknik Negeri Jakarta, Jun. 2025, pp. 607–614.

[9] Oktrison, D. N. Ilham, R. A. Candra, and E. Sipahutar, "Design and Performance Analysis of a Low-Cost ESP32-Based NAT WiFi Repeater for Indoor IoT Networks," *Global Advances in Science, Engineering & Technology (GASET)*, vol. 1, no. 2, pp. 61–71, Feb. 2026, doi: 10.62671/gaset.v1i2.249.

[10] L. P. Verma, G. Kumar, O. I. Khalaf, W.-K. Wong, A. A. Hamad, and S. S. Rawat, "Adaptive congestion control in IoT networks: Leveraging one-way delay for enhanced performance," *Heliyon*, vol. 10, no. 22, p. e40266, Nov. 2024, doi: 10.1016/j.heliyon.2024.e40266.

[11] I. A. Renaldy, R. Hidayati, and K. Sari, "Optimalisasi Transmisi Lora Melalui Mqtt Mosquitto dengan Metode Data Chunking," *Simetris: Jurnal Teknik Mesin, Industri, Elektro dan Ilmu Komputer*, vol. 17, no. 1, pp. 247–260, Apr. 2026, doi: 10.24176/simet.v17i1.15820.