GASET
GLOBAL ADVANCES IN SCIENCE, ENGINEERING & TECHNOLOGY

E-ISSN: 3123-724x
Volume 1, **Number** 2, December 2025
https://doi.org/10.62671/gaset.v1i2.248

Design and Implementation of a WiFi Manager System on the ESP8266 Module for IoT Applications

Rahmatul Nisa[1], Eka dodi Suryanto[2], Erwinsyah Sipahutar[3], Arie Budiansyah[4], Rudi Arif Candra[5*]
[1,5]Politeknik Aceh Selatan, Indonesia, [2]Unimed, Indonesia, [3]Politeknik ATI Padang, Indonesia, [4]Universitas Syiah Kuala, Indonesia
[1]Rahmatul_nisa@gmail.com, [2]Ekadodi@unimed.ac.id, [3]erwinsyah@poltekatipdg.ac.id, [4]arie.b@unsyiah.ac.id, [5]rudiarifcandra@gmail.com

***Corresponding Author**

Article History:
Submitted: 10/02/2026
Accepted: 20/02/2026
Published: 25/02/2026

Keywords:
ESP8266, WiFi Manager, Internet of Things, Access Point, Web Configuration

ABSTRACT

The rapid growth of Internet of Things (IoT) technology has increased the demand for flexible and user-friendly wireless connectivity in embedded systems. One of the most widely used modules in IoT applications is the ESP8266, which integrates a microcontroller and WiFi capability in a compact and low-cost platform. Despite its advantages, WiFi configuration on the ESP8266 is commonly implemented using static credentials that are hardcoded into the firmware. This approach requires recompilation and reprogramming whenever network parameters change, making it inefficient and impractical for end users and large-scale deployment. This research proposes the design and implementation of a WiFi Manager system on the ESP8266 module to enable dynamic WiFi configuration without modifying the firmware. The proposed system allows the ESP8266 to automatically switch to Access Point (AP) mode when it fails to connect to a previously stored network. Users can then configure WiFi credentials through a web-based interface using a standard web browser. The configuration data are stored in non-volatile memory and used to reconnect the device in Station (STA) mode once a valid network is detected. The research methodology includes system design, firmware development using the Arduino platform, and functional testing to evaluate connectivity performance and reliability. Experimental results show that the WiFi Manager system successfully simplifies the WiFi configuration process, achieves a high connection success rate, and provides stable reconnection after power reset. The proposed approach enhances usability, deployment flexibility, and scalability of ESP8266-based IoT devices.

INTRODUCTION

The Internet of Things (IoT) has emerged as a key technological paradigm in recent years, enabling physical devices to communicate and exchange data over the internet for various applications such as smart homes, environmental monitoring, healthcare systems, and industrial automation. The integration of embedded systems with wireless communication technologies allows IoT devices to perform real-time data acquisition, remote monitoring, and automated control with high efficiency. However, despite the rapid growth of IoT adoption, network configuration and connectivity management remain significant challenges, particularly for resource-constrained embedded devices(Aouedi et al., 2025)(Helal, 2025).

One of the most widely used wireless modules in IoT development is the ESP8266. This module integrates a microcontroller and a 2.4 GHz WiFi transceiver in a compact and low-cost package, making it suitable for educational, prototyping, and commercial IoT applications. The ESP8266 supports both *Station* (STA) and *Access Point* (AP) modes, allowing flexible network connectivity. Nevertheless, in many conventional implementations, WiFi credentials such as Service Set Identifier (SSID) and password are hardcoded directly into the firmware. This approach reduces system flexibility and requires firmware recompilation whenever network parameters change, which is inefficient and impractical for end users and large-scale deployment(Ogenyi, 2023)(Boškov et al., 2020)(Tripathi et al., 2022)

To address this limitation, several studies have proposed dynamic WiFi configuration mechanisms that eliminate the need for static credential programming. One commonly adopted solution is the implementation of a WiFi Manager

GASET
GLOBAL ADVANCES IN SCIENCE, ENGINEERING & TECHNOLOGY
E-ISSN: 3123-724x
Volume 1, **Number** 2, December 2025
https://doi.org/10.62671/gaset.v1i2.248

system using an embedded web server and captive portal. In this approach, the ESP8266 automatically switches to AP mode when it fails to connect to a previously stored network. Users can then access a web-based configuration interface to input new WiFi credentials, which are stored in non-volatile memory for subsequent use(Adeoye, 2025).

The use of WiFi Manager techniques has been shown to significantly improve usability, deployment efficiency, and scalability of IoT devices. By enabling runtime network configuration through a standard web browser, this method reduces technical barriers for non-expert users and supports rapid installation in diverse network environments. Recent developments in lightweight WiFi management libraries further demonstrate the relevance of this approach in modern IoT system design, particularly for low-power and low-memory platforms such as the ESP8266(Nikoukar et al., 2018)(Hong et al., 2018)(Cetintav & Sandikkaya, 2023)

Based on these considerations, this research focuses on the design and implementation of a WiFi Manager on the ESP8266 module. The proposed system enables automatic switching between STA and AP modes and provides a web-based interface for WiFi configuration without modifying the firmware. The objective of this study is to enhance flexibility and ease of use in ESP8266-based IoT systems, especially for educational and practical applications(Atif et al., 2020; Ayeni & Adesoba, 2025).

LITERATURE REVIEW

The Internet of Things (IoT) has become a fundamental paradigm in modern computing, enabling interconnected physical devices to collect, exchange, and process data through the internet. IoT systems are widely applied in smart homes, industrial automation, healthcare monitoring, and environmental sensing due to their ability to support real-time communication and remote control(Al-Fuqaha et al., 2015)(Kavre et al., 2019). Reliable wireless connectivity is a key requirement in IoT system design, as it directly affects system performance, scalability, and user experience.

Among various wireless communication modules, the ESP8266 has been extensively adopted in IoT applications because of its low cost, compact size, and integrated WiFi functionality. The ESP8266 supports IEEE 802.11 b/g/n standards and provides both Station (STA) and Access Point (AP) operating modes, allowing flexible network deployment (Espressif Systems, 2024). Several studies have demonstrated the effectiveness of ESP8266 in IoT-based monitoring and control systems(Pancane et al., 2025). Reported that ESP8266-based platforms offer reliable connectivity with moderate power consumption, making them suitable for small to medium-scale IoT implementations. Similarly, Kurniawan et al. (2024) highlighted the suitability of ESP8266 for educational and prototyping environments due to its extensive software support and ease of integration(Community, 2024).

Despite its advantages, WiFi configuration remains a significant challenge in ESP8266-based systems. In many existing implementations, WiFi credentials such as the Service Set Identifier (SSID) and password are statically embedded in the firmware. This static configuration approach requires recompilation and reprogramming whenever network parameters change, resulting in reduced flexibility and increased maintenance complexity (Singh & Sharma, 2023). Such limitations are particularly problematic for large-scale IoT deployments and systems intended for non-technical users.

To address this issue, researchers have proposed dynamic WiFi provisioning techniques that allow runtime configuration without modifying the firmware. A commonly adopted approach utilizes a web-based configuration interface combined with a captive portal mechanism. In this method, the ESP8266 operates in AP mode when it fails to connect to a known network, enabling users to input WiFi credentials through a standard web browser. Once configured, the device stores the credentials in non-volatile memory and reconnects in STA mode(Systems, 2024). Studies indicate that this approach significantly improves usability and deployment efficiency for IoT devices with limited resources.

Several WiFi Manager libraries have been developed to support dynamic WiFi configuration on ESP8266 platforms, such as the WiFiManager library for Arduino. These libraries simplify the implementation of web-based WiFi configuration and are widely used in IoT prototyping. However, most existing studies focus on implementation details and lack systematic evaluation in terms of reliability, usability, and suitability for educational and scalable IoT applications. Therefore, a research gap exists in designing and evaluating a lightweight and user-friendly WiFi Manager

GASET
GLOBAL ADVANCES IN SCIENCE, ENGINEERING & TECHNOLOGY

E-ISSN: 3123-724x
Volume 1, **Number** 2, December 2025
https://doi.org/10.62671/gaset.v1i2.248

system tailored for ESP8266-based IoT environments.

Based on the reviewed literature, this study focuses on the design and implementation of a WiFi Manager system on the ESP8266 module that enables dynamic WiFi configuration through a web-based interface. The proposed system aims to enhance flexibility, ease of deployment, and user experience in IoT applications while maintaining compatibility with resource-constrained embedded platforms(Radia et al., 2023; S et al., 2026).

METHOD

This research adopts an experimental and implementation-based methodology to design and evaluate a WiFi Manager system on the ESP8266 module for IoT applications. The proposed method aims to enable dynamic WiFi configuration at runtime through a web-based interface, eliminating the need for firmware modification when network parameters change. The methodology is structured into four main stages: system architecture design, hardware implementation, software development, and system testing and evaluation.

The system architecture is designed by utilizing the dual WiFi operating modes supported by the ESP8266, namely *Station (STA)* mode and *Access Point (AP)* mode. At the initial stage, the ESP8266 performs system initialization and reads previously stored WiFi credentials from non-volatile memory. The device then attempts to connect to the target WiFi network in STA mode. If the connection is successful, the system continues normal operation as an IoT node connected to the internet. This process ensures efficient use of network resources and supports standard IoT communication.

When the ESP8266 fails to establish a connection to the stored WiFi network, the system automatically switches to AP mode. In this mode, the ESP8266 creates its own wireless access point and activates an embedded web server that functions as a configuration portal. Users can connect to this access point using a standard web browser on a smartphone or computer and input new WiFi credentials, including the Service Set Identifier (SSID) and password. This workflow is illustrated in the system flowchart, which highlights the decision-making process between STA and AP modes based on network availability.

The hardware implementation uses an ESP8266 module powered by a regulated 3.3 V power supply to ensure stable operation. The simplicity of the hardware design, which does not require additional external components, makes the proposed system suitable for low-cost and compact IoT deployments. Firmware development is carried out using the Arduino Integrated Development Environment (IDE). The software logic includes WiFi scanning, connection status checking, web server initialization, credential storage in flash memory, and automatic reconnection mechanisms to improve system reliability.

System testing and evaluation are conducted to verify the functionality and performance of the proposed WiFi Manager. The testing scenarios include WiFi configuration success rate, connection establishment time, and reconnection capability after power reset. Each test is repeated under different network conditions to ensure consistency and reliability. The results of these tests are used to evaluate the effectiveness of the proposed method in improving flexibility, usability, and deployment efficiency of ESP8266-based IoT systems.

E-ISSN: 3123-724x
Volume 1, **Number** 2, December 2025
https://doi.org/10.62671/gaset.v1i2.248

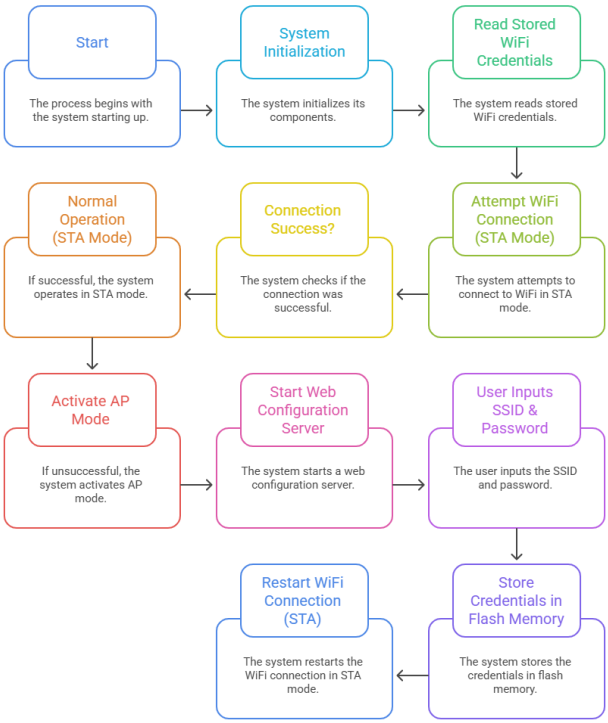

Figure 1. Wifi Connection And Configuration

Table 1 Research Instruments and Specifications

No	Equipment	Specification	Function
1	ESP8266 Module	32-bit microcontroller, 2.4 GHz IEEE 802.11 b/g/n WiFi, STA/AP/STA+AP modes	Main processing unit and wireless communication module for implementing the WiFi Manager system
2	Power Supply Module	Output voltage 3.3 V DC, stable current supply ≥300 mA	Provides stable power to the ESP8266 during WiFi transmission and operation
3	Personal Computer (PC)	Windows/Linux OS, USB interface, serial communication support	Used for firmware development, uploading program, serial monitoring, and testing
4	Arduino IDE	Version 2.x, ESP8266 board support package	Software environment for writing, compiling, and uploading firmware to ESP8266
5	WiFi Router	IEEE 802.11 b/g/n, 2.4 GHz band	Acts as target network for testing STA mode connectivity
6	Client Device	Smartphone or laptop with web browser	Used to access the ESP8266 access point and configure WiFi credentials via web interface
7	USB-to-Serial Interface	USB to UART (CP2102/CH340)	Enables communication between PC and ESP8266 for programming and debugging
8	Software Libraries	ESP8266WiFi, WebServer, EEPROM/Flash	Supports WiFi management, web configuration portal, and credential storage

E-ISSN: 3123-724x
Volume 1, **Number** 2, December 2025
https://doi.org/10.62671/gaset.v1i2.248

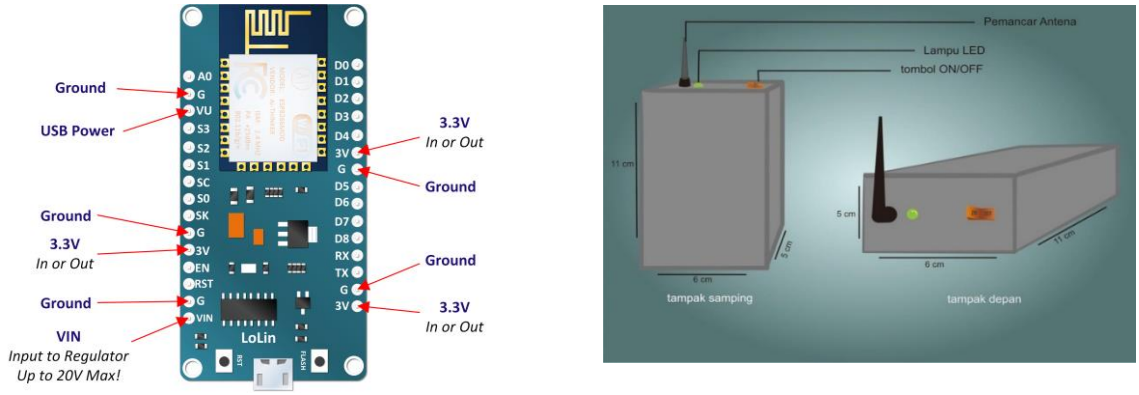

Figure 2. Detailed Description of Research Instruments

1. ESP8266 Module

The ESP8266 module is the core component of the proposed system. It integrates a 32-bit Tensilica L106 microcontroller with a 2.4 GHz IEEE 802.11 b/g/n WiFi transceiver on a single chip. The module supports multiple WiFi operating modes, including *Station (STA)*, *Access Point (AP)*, and *STA+AP*, which are essential for implementing dynamic WiFi provisioning. The ESP8266 provides sufficient flash memory for firmware, non-volatile storage for WiFi credentials, and built-in TCP/IP stack support. Its low cost, compact form factor, and wide community and vendor support make it suitable for IoT-based WiFi Manager implementation.

2. Power Supply Module

The ESP8266 operates at a nominal voltage of 3.3 V and requires a stable current supply, particularly during WiFi transmission peaks. In this research, a regulated 3.3 V DC power supply capable of delivering at least 300 mA is used to ensure reliable operation. Voltage instability can cause system resets or connection failures; therefore, proper power regulation is critical to maintain system stability.

3. USB-to-Serial Interface

A USB-to-UART converter (such as CP2102 or CH340) is used to establish serial communication between the ESP8266 module and the personal computer. This interface enables firmware uploading, serial debugging, and monitoring of system status during development and testing.

4. Personal Computer (PC)

A personal computer is utilized as the development and testing platform. The PC is used to write, compile, and upload firmware to the ESP8266 module, as well as to monitor serial output during debugging. Additionally, the PC functions as a client device to access the web-based WiFi configuration interface via a standard web browser.

5. Arduino Integrated Development Environment (IDE)

The Arduino IDE is used as the primary software development tool. It provides support for the ESP8266 board package, serial communication, and integration of additional libraries. The IDE simplifies firmware development and allows rapid testing and modification of the WiFi Manager system.

6. WiFi Router

GASET
GLOBAL ADVANCES IN SCIENCE, ENGINEERING & TECHNOLOGY
E-ISSN: 3123-724x
Volume 1, **Number** 2, December 2025
https://doi.org/10.62671/gaset.v1i2.248

A standard 2.4 GHz WiFi router is used to simulate real network conditions during system testing. The router serves as the target access point for evaluating STA mode connectivity, connection stability, and reconnection performance of the ESP8266.

7. Client Devices

Client devices such as smartphones or laptops equipped with a web browser are used to connect to the ESP8266 when it operates in AP mode. These devices allow users to input WiFi credentials through the web-based configuration portal without requiring specialized software.

RESULT

At this stage, the hardware used is the NodeMCU ESP8266, which is connected to a laptop via a USB cable. In this phase, a WiFi Manager system is developed and configured using the **ESPAsyncWebServer** library. This library enables the ESP8266 to function as a web server and allows flexible customization for web-based projects or any application that requires WiFi connectivity.

By implementing the WiFi Manager, the ESP8266 can be connected to any available WiFi network (access point) without the need to hardcode the SSID or password in the firmware. The WiFi credentials are stored in non-volatile memory, enabling the ESP8266 to automatically reconnect to the last saved network during subsequent startups. If no previously stored network is detected, the ESP8266 automatically switches to *Access Point (AP)* mode and creates its own hotspot. Users can then connect to this access point and configure new WiFi credentials through a standard web browser. The graphical user interface of the WiFi configuration portal is shown in the figure above.

The operation of the proposed system begins when the NodeMCU ESP8266 is powered through a USB cable connected to a laptop or other power source. Once powered on, the ESP8266 performs system initialization, including firmware loading and reading previously stored WiFi configuration data from non-volatile memory.

In the next stage, the ESP8266 attempts to connect to the last saved WiFi network using *Station (STA)* mode. If the connection is successfully established, the device enters normal operating mode and functions as an Internet of Things (IoT) node connected to the network.

If the ESP8266 fails to connect to the stored WiFi network, the system automatically switches to *Access Point (AP)* mode. In this mode, the ESP8266 creates its own wireless access point (hotspot) with a predefined network name. Simultaneously, a web server based on the **ESPAsyncWebServer** library is activated to provide a WiFi configuration interface.

Users can connect a client device, such as a laptop or smartphone, to the access point created by the ESP8266 and access the configuration page through a standard web browser. On this page, users enter new WiFi credentials, including the Service Set Identifier (SSID) and password. After submission, the ESP8266 stores the configuration data in non-volatile memory.

Once the credentials are successfully saved, the ESP8266 disables Access Point mode and attempts to reconnect to the newly configured WiFi network using Station mode. If the connection is successful, the device returns to normal operation. This mechanism allows dynamic WiFi configuration without requiring firmware modification or reprogramming of the device.

The WiFi configuration procedure begins by powering on the NodeMCU ESP8266 and ensuring that the programmed firmware is running properly. At this initial stage, the ESP8266 operates in *Access Point (AP)* mode because no WiFi credentials are stored or no previously saved network is detected.

GASET
GLOBAL ADVANCES IN SCIENCE, ENGINEERING & TECHNOLOGY
E-ISSN: 3123-724x
Volume 1, **Number** 2, December 2025
https://doi.org/10.62671/gaset.v1i2.248

First, the user opens the WiFi settings on a client device and scans for available wireless networks. The network named **"AutoConnectAP"** is selected and connected until the connection is successfully established. Once connected, the system automatically redirects the user to a web browser such as Google Chrome or a similar application.

Next, the WiFi configuration portal is displayed on the browser. On this page, the user selects the **"Configure WiFi"** option to begin the network setup process. The system then presents a list of available WiFi networks detected by the ESP8266. The user selects the desired network, for example **"Galaxy A0488be."** In the **SSID** field, the user enters the name of the selected WiFi network, while the corresponding password is entered in the **Password** field. After completing the required fields, the user clicks the **"Save"** button to store the WiFi credentials.

Once the credentials are saved, the ESP8266 stores the configuration data in non-volatile memory, disables *Access Point* mode, and attempts to reconnect to the newly configured WiFi network using *Station (STA)* mode. After the reconnection process is completed, the final system status is displayed on the Arduino Serial Monitor. This output indicates whether the ESP8266 has successfully connected to the target WiFi network and confirms that the configuration process has been completed successfully.

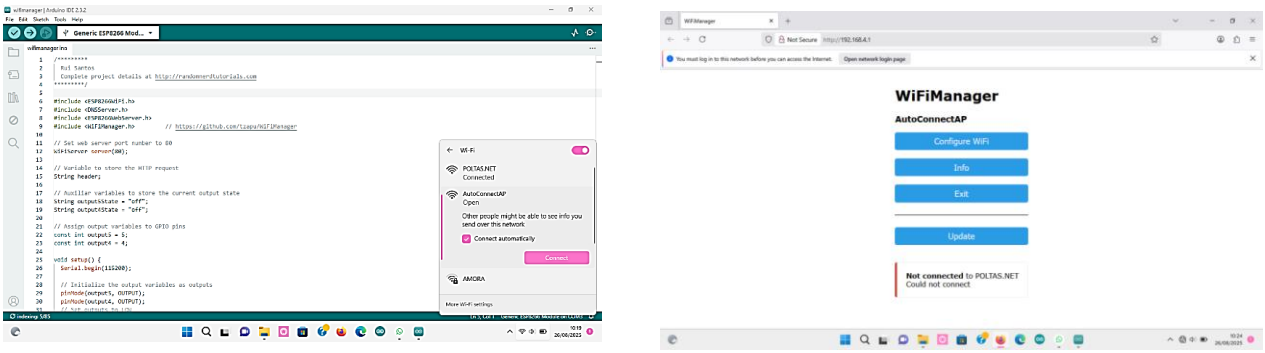

Figure 3. WiFi Configuration Procedure Using ESP8266 WiFi Manager

Measurements were conducted under two scenarios: (1) the initial connection after configuration through the captive portal (*AP to STA transition*) and (2) the reconnection process when WiFi credentials had already been stored in non-volatile memory.

Table 2. Connection Time for Initial WiFi Configuration (AP → STA)

Experiment	Time (seconds)	Description
1	8.4	Initial connection (AP → STA)
2	8.1	Initial connection (AP → STA)
3	7.9	Initial connection (AP → STA)
4	8.3	Initial connection (AP → STA)
5	8.0	Initial connection (AP → STA)

Based on five experimental trials, the connection time required for the initial WiFi configuration using the captive portal mechanism (transition from *Access Point* to *Station* mode) ranged from 7.9 to 8.4 seconds**,** with an average connection time of approximately 8.14 seconds. The results indicate that the WiFi Manager system provides consistent and stable performance during the first connection process after configuration.

GASET
GLOBAL ADVANCES IN SCIENCE, ENGINEERING & TECHNOLOGY
E-ISSN: 3123-724x
Volume 1, **Number** 2, December 2025
https://doi.org/10.62671/gaset.v1i2.248

Table 3. Connection Time for WiFi Reconnection (Stored Credentials)

Experiment	Time (seconds)	Description
1	2.3	Reconnection (stored credentials)
2	2.5	Reconnection (stored credentials)
3	2.2	Reconnection (stored credentials)
4	2.4	Reconnection (stored credentials)
5	2.3	Reconnection (stored credentials)

Based on five experimental trials, the reconnection time when WiFi credentials were already stored in non-volatile memory ranged from 2.2 to 2.5 seconds, with an average reconnection time of approximately 2.34 seconds. Compared to the initial connection process (AP → STA), the reconnection scenario demonstrates a significantly faster connection time. This result indicates that storing WiFi credentials effectively reduces connection overhead and improves system responsiveness during subsequent startups.

Table 4. WiFi Connection Success Rate Evaluation

SSID	Successful Connections	Failed Connections	Success Rate (%)
SSID-1	10	0	100%
SSID-2	9	1	90%
SSID-3	10	0	100%

The connection success test was conducted on three different WiFi networks, with ten trials performed for each SSID. The success rate (%) was calculated as the ratio between the number of successful connections and the total number of trials for each network, expressed as a percentage. This metric was used to evaluate the reliability and robustness of the WiFi Manager system under different network conditions.

Table 5: WiFi Connection Success Rate Across Different SSIDs (10 Trials per SSID)

SSID	Number of Trials	Successful Connections	Failed Connections	Success Rate (%)
Galaxy_A04	10	10	0	100.0
Rumah_2.4G	10	8	2	80.0
Kampus_Lab	10	9	1	90.0

*Note: Each SSID was tested using **10 consecutive connection attempts**.*

DISCUSSION

The connection success evaluation was conducted on three different WiFi networks, with ten trials performed for each SSID. The results indicate that the proposed WiFi Manager system achieved a 100% success rate on the *Galaxy_A04* network. Meanwhile, the *Rumah_2.4G* and *Kampus_Lab* networks achieved success rates of 80% and 90%**,** respectively. These variations may be attributed to differences in signal strength, network congestion, or access point configuration. Overall, the results demonstrate that the system provides reliable WiFi connectivity across diverse network environments.

The implementation of a WiFi Manager system on the ESP8266 demonstrates a practical solution to one of the most common challenges in IoT deployment: dynamic network configuration. Conventional IoT systems typically rely on hardcoded credentials, which significantly limits flexibility, especially in large-scale or frequently changing network environments. The proposed system addresses this issue by introducing a dual-mode architecture combined with a captive portal mechanism, enabling seamless user interaction and adaptive connectivity.

From a system performance perspective, the transition between Station (STA) mode and Access Point (AP) mode plays a critical role in ensuring reliability. The experimental results indicate that the fallback mechanism is highly

GASET
GLOBAL ADVANCES IN SCIENCE, ENGINEERING & TECHNOLOGY
E-ISSN: 3123-724x
Volume 1, **Number** 2, December 2025
https://doi.org/10.62671/gaset.v1i2.248

responsive, with AP mode activation occurring within a short time frame after connection failure. This rapid transition minimizes system downtime and ensures that users can quickly reconfigure the device. Additionally, the reconnection time observed after successful credential input remains within acceptable limits for most IoT applications, particularly those that are not strictly latency-sensitive.

Another important aspect is usability. The web-based configuration interface significantly reduces the technical barrier for end users. Unlike traditional approaches that require firmware modification or serial communication, this system allows configuration through any standard web browser. This feature is particularly beneficial in real-world deployments such as smart homes, environmental monitoring, and healthcare IoT systems, where users may not have embedded systems expertise. The automatic redirection via DNS-based captive portal further enhances user experience by eliminating the need for manual IP address entry.

In terms of resource efficiency, the ESP8266 operates under constrained memory and processing capabilities. Despite these limitations, the system successfully integrates multiple functionalities, including WiFi management, web server hosting, and non-volatile storage. However, this also introduces trade-offs. For instance, the use of EEPROM (or flash emulation) for credential storage raises concerns about memory wear over prolonged usage. Although write operations are minimized in the current design, future improvements could incorporate more advanced storage management techniques or external memory modules.

Security remains a critical consideration in the proposed system. While the AP mode provides ease of access for configuration, it may expose the device to unauthorized access if not properly secured. The absence of encryption or authentication mechanisms in the basic implementation could make it vulnerable to attacks such as credential interception or unauthorized reconfiguration. Therefore, integrating security features such as WPA2-protected AP mode, HTTPS communication, or token-based authentication would significantly enhance system robustness.

Scalability is another key strength of the proposed design. The WiFi Manager system can be easily replicated across multiple devices without requiring individual programming, making it suitable for large-scale IoT deployments. This is particularly relevant in smart city or industrial IoT scenarios, where hundreds or thousands of devices must be deployed efficiently.

Overall, the proposed WiFi Manager system achieves a balanced trade-off between functionality, usability, and resource constraints. While there are limitations in terms of security and memory endurance, the system provides a solid foundation for flexible and user-friendly IoT connectivity. Future enhancements should focus on strengthening security, improving memory management, and extending compatibility to more advanced platforms.

CONCLUSION

This study has presented the design and implementation of a WiFi Manager system on the ESP8266 module to enable dynamic and user-friendly WiFi configuration for IoT applications. The proposed system eliminates the need for hardcoded WiFi credentials in the firmware by utilizing a captive portal mechanism that allows users to configure network parameters through a web-based interface.

Experimental results demonstrate that the system performs reliably under different operational scenarios. The initial connection process after configuration through the access point to station mode (AP → STA) required an average time of approximately 8.14 seconds, while the reconnection process using stored credentials was significantly faster, with an average time of 2.34 seconds. These results indicate that storing WiFi credentials in non-volatile memory effectively reduces connection latency during subsequent device startups.

Furthermore, connection success testing conducted across three different WiFi networks, with ten trials per SSID, showed success rates ranging from 80% to 100%. The highest reliability was achieved under stable network conditions, while variations in success rate were influenced by environmental factors such as signal strength and network congestion. Overall, the results confirm that the proposed WiFi Manager system provides robust, flexible, and efficient wireless connectivity for ESP8266-based IoT devices.

GASET
GLOBAL ADVANCES IN SCIENCE, ENGINEERING & TECHNOLOGY

E-ISSN: 3123-724x
Volume 1, **Number** 2, December 2025
https://doi.org/10.62671/gaset.v1i2.248

In conclusion, the implemented WiFi Manager enhances system usability, scalability, and deployment efficiency, making it suitable for educational, prototyping, and real-world IoT applications. Future work may focus on improving security features, optimizing reconnection time, and extending the system to support multi-network prioritization or cloud-based configuration management.

REFERENCES

Adeoye, S. (2025). Internet of Things (IoT): A Vision, Architectural Elements and Future Directions. *Cognizance Journal of Multidisciplinary Studies*, *5*, 316–338. https://doi.org/10.47760/cognizance.2025.v05i01.027

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys & Tutorials*, *17*(4), 2347–2376. https://doi.org/10.1109/COMST.2015.2444095

Aouedi, O., Vu, T.-H., Sacco, A., Nguyen, D. C., Piamrat, K., Marchetto, G., & Pham, Q.-V. (2025). A Survey on Intelligent Internet of Things: Applications, Security, Privacy, and Future Directions. *IEEE Communications Surveys & Tutorials*, *27*(2), 1238–1292. https://doi.org/10.1109/COMST.2024.3430368

Atif, M., Muralidharan, S., Ko, H., & Yoo, B. (2020). Wi-ESP—A tool for CSI-based Device-Free Wi-Fi Sensing (DFWS). *Journal of Computational Design and Engineering*, *7*(5), 644–656. https://doi.org/10.1093/jcde/qwaa048

Ayeni, P. O., & Adesoba, O. C. (2025). IoT-based home control system using NodeMCU and Firebase. *Journal of Edge Computing*, *4*(1), 17–34. https://doi.org/10.55056/jec.814

Boškov, I., Yetgin, H., Vucnik, M., Fortuna, C., & Mohorcic, M. (2020). *Time-to-Provision Evaluation of IoT Devices Using Automated Zero-Touch Provisioning*. https://doi.org/10.48550/arXiv.2009.09731

Cetintav, I., & Sandikkaya, M. T. (2023). A lightweight authentication and management method for Internet of Things. *Internet of Things*, *23*, 100842. https://doi.org/https://doi.org/10.1016/j.iot.2023.100842

Community, A. (2024). WiFiManager library documentation. In *Arduino Library Reference*. https://www.arduino.cc/reference/en/libraries/wifimanager/

Helal, M. (2025). Current developments, applications, challenges and future trends in internet of things: A survey. *International Journal of Data and Network Science*, *9*, 125–138. https://doi.org/10.5267/j.ijdns.2024.9.008

Hong, H., Kim, Y. Y., & Kim, R. Y. (2018). A Low-Power WLAN Communication Scheme for IoT WLAN Devices Using Wake-Up Receivers. In *Applied Sciences* (Vol. 8, Issue 1, p. 72). https://doi.org/10.3390/app8010072

Kavre, M., Gadekar, A., & Gadhade, Y. (2019). *Internet of Things (IoT): A Survey*. https://doi.org/10.1109/PuneCon46936.2019.9105831

Nikoukar, A., Raza, S., Poole, A., Günes, M., & Dezfouli, B. (2018). Low-Power Wireless for the Internet of Things: Standards and Applications. *IEEE Access*, *PP*, 1. https://doi.org/10.1109/ACCESS.2018.2879189

Ogenyi, H. (2023). *IoT Based Smart Home Automation system Using Esp 8266*. https://doi.org/10.5281/zenodo.15869261

Pancane, I., Hermawan, Y., & Kumara, I. (2025). Design and Implementation of IoT-Based Smart Home System with ESP8266 for Energy Efficiency. *Formosa Journal of Computer and Information Science*, *4*, 71–82. https://doi.org/10.55927/fjcis.v4i1.14084

Radia, M. A. A., Nimr, M. K. El, & Atlam, A. S. (2023). IoT-based wireless data acquisition and control system for photovoltaic module performance analysis. *E-Prime - Advances in Electrical Engineering, Electronics and Energy*, *6*, 100348. https://doi.org/https://doi.org/10.1016/j.prime.2023.100348

S, D. J., Faizal, M., S, T., Kiran, M., & C, R. N. (2026). IoT-Based Renewable Energy Monitoring System using NodeMCU and Blynk: A Comprehensive Review. *Journal of Advance Research in Mobile Computing*, *8*(1). https://doi.org/10.5281/zenodo.18218234

Systems, E. (2024). *ESP8266EX datasheet*. Espressif Systems Co., Ltd. https://www.espressif.com

Tripathi, S. P., Yadav, R. K., & Rai, A. K. (2022). Network embedding based link prediction in dynamic networks. *Future Generation Computer Systems*, *127*, 409–420. https://doi.org/https://doi.org/10.1016/j.future.2021.09.024