ESP32 Marauder: Educational Wireless Analysis Platform

December 2025

Abstract

This project presents the development and evaluation of the ESP32 Marauder as an accessible, open-source platform for wireless security education and hands-on network analysis. Designed around the low-cost ESP32 microcontroller, the system offers a practical alternative to commercial wireless auditing tools whose high cost often restricts their use in academic environments. Experimental evaluation demonstrated strong performance, including a detection accuracy of 96.3% and stable operation during extended testing. These results affirm the platform's suitability for teaching wireless protocols, embedded systems, and foundational cybersecurity concepts. By reducing financial and technical barriers, the ESP32 Marauder enables wider adoption of experiential cybersecurity training, particularly in resource-limited institutions, and contributes meaningfully to addressing the growing skills gap in modern network defense.

Contents

List of Figures

List of Tables

1 Introduction

Wireless security has become an essential component of modern network defense as contemporary communication infrastructures increasingly rely on Wi-Fi, Bluetooth, and other short-range wireless protocols for critical data exchange. Studies indicate that misconfigurations, weak encryption schemes, and inadequate monitoring practices remain among the leading sources of wireless network vulnerabilities in academic and enterprise environments [1]. Despite the rising importance of wireless security, access to practical training tools remains limited, particularly in institutions with restricted budgets. Commercial wireless auditing platforms, although powerful, are often costly and proprietary, constraining their adoption in hands-on cybersecurity education [2].

Open-source hardware platforms have therefore emerged as a promising alternative for enabling practical instruction in embedded systems, radio-frequency (RF) analysis, and network defense. Low-cost microcontrollers such as the ESP32 have demonstrated sufficient processing power and integrated wireless capabilities to support a broad range of security-oriented functionalities [3]. Among these solutions, the ESP32 Marauder has gained attention as an accessible, community-driven tool capable of performing Wi-Fi scanning, packet monitoring, Bluetooth reconnaissance, and attack simulations in a compact form factor. Early studies suggest that open-source wireless analysis tools not only reduce financial barriers but also improve learning outcomes by providing students with direct interaction with real-world RF environments [4].

This project investigates the ESP32 Marauder as an educational platform for wireless analysis, evaluating its technical capabilities, performance stability, and pedagogical relevance. Through systematic testing and structured analysis, the study aims to determine the suitability of the device for integration into academic cybersecurity curricula. The work contributes to the ongoing conversation regarding low-cost hardware for cybersecurity education and addresses the broader need for accessible, scalable, and practical learning tools [5].

2 Literature Review and Background

2.1 Wireless Communication Fundamentals

2.1.1 Wi-Fi Protocols

Wireless communication relies heavily on IEEE 802.11 standards, which define the structure and behavior of Wi-Fi networks. The ESP32 platform implements the 802.11 b/g/n specifi-

cations operating in the 2.4 GHz ISM band [6]. These standards define management, control, and data frames, including beacon frames, probe requests, and data packets. Such frames contain parameters including MAC addresses, Received Signal Strength Indicator (RSSI), and network configuration fields.
Understanding these protocol-level details is essential for wireless security analysis, as misconfigurations and flawed protocol implementations frequently introduce exploitable vulnerabilities [7].

2.1.2 Bluetooth Low Energy (BLE)

Bluetooth Low Energy (BLE) enables low-power personal area networking through advertising channels that broadcast short discovery packets. These packets provide device identifiers and connection parameters necessary for link establishment [8]. Firmware such as ESP32 Marauder leverages these BLE broadcasts to detect nearby devices, catalog metadata, and support educational demonstrations of short-range wireless technologies and their associated security considerations.

2.2 Embedded Systems in Cybersecurity Education

2.2.1 ESP32 Platform

Embedded systems have become central to cybersecurity education due to their affordability and accessibility. The ESP32 system-on-chip integrates dual-core processing, Wi-Fi, and Bluetooth support, along with low-power operational modes [9]. Its architecture featuring an Xtensa LX6 processor, 520 KB SRAM, and extensive GPIO support has made it a widely adopted platform for wireless experimentation in academic environments. The low cost of deployment enables institutions to equip students with hands-on experience in network security, device programming, and embedded analysis.

2.2.2 Marauder Firmware

The ESP32 Marauder firmware, an open-source project, extends the ESP32's capabilities by adding real-time Wi-Fi/Bluetooth scanning and packet capture functionalities [10]. Its modular interface supports menu-driven operation, data logging via microSD card, and display-based visualization. These features transform the ESP32 into a low-cost educational tool that parallels functionalities traditionally limited to professional network analyzers.

2.3 Gap Analysis

2.3.1 Technical Gaps

Existing wireless analysis tools often present substantial cost and usability barriers. Commercial spectrum analyzers can cost between $500 and $5000, limiting adoption in educational settings [11]. Similarly, several open-source alternatives require significant technical expertise for setup and maintenance, contributing to their limited classroom integration. Furthermore, there remains a shortage of structured curricula that align firmware-based wireless tools with explicit learning outcomes.

2.3.2 Regional Context

In developing regions such as Pakistan, limited institutional budgets, import constraints, and high student-to-equipment ratios restrict access to advanced wireless security laboratories [12]. As the cybersecurity job market expands, the need for practical skill development intensifies. Affordable embedded tools like the ESP32, when supported by structured educational content offer a scalable means to address these gaps and enhance hands-on learning opportunities.

3 Methodology

3.1 System Design Overview

The system was developed using a structured, multi-phase workflow to ensure repeatability and clarity. The overall process consisted of four primary stages:

1. **Setup and Configuration:** Hardware assembly, wiring, and installation of the ESP32 Marauder firmware.
2. **Controlled Testing:** Evaluation of system performance in isolated wireless environments.
3. **Data Collection and Analysis:** Measurement of detection accuracy, stability, and signal characteristics.
4. **Educational Framework Development:** Preparation of supporting documentation and instructional material.

3.2 Hardware Architecture

3.2.1 Component Selection

Table 3.2.1 summarizes the major hardware components used in the system, along with their specifications, purpose, and integration method.

Component	**Specifications**	**Purpose**	**Integration Method**
ESP32 Development Board	Dual-core 240 MHz, Wi-Fi/Bluetooth	Core processing and wireless interface	USB connection for programming
MicroSD Card Module	SPI, up to 32 GB	Data logging and storage	SPI connection to ESP32
Push Buttons (3–5)	Tactile switches	Menu navigation	GPIO with pull-up resistors
Power Cable	Micro USB	Power + data	Direct USB connection
Breadboard & Wires	Standard prototyping kit	Circuit assembly	Modular jumper wires
Power Supply	5 V, 1 A	System power	USB or external adapter

Table 1: Hardware Components and Integration

3.2.2 Circuit Design

The circuit utilized SPI and GPIO interfaces, with MicroSD module to minimize pin usage. Tables3.2.2 summarize the final wiring configuration.

Signal	**ESP32 GPIO Pin**
MOSI	GPIO 23 (shared)
MISO	GPIO 19 (shared)
CLK	GPIO 18 (shared)
CS	GPIO 5

Table 2: MicroSD Card Module (SPI) Pin Connections

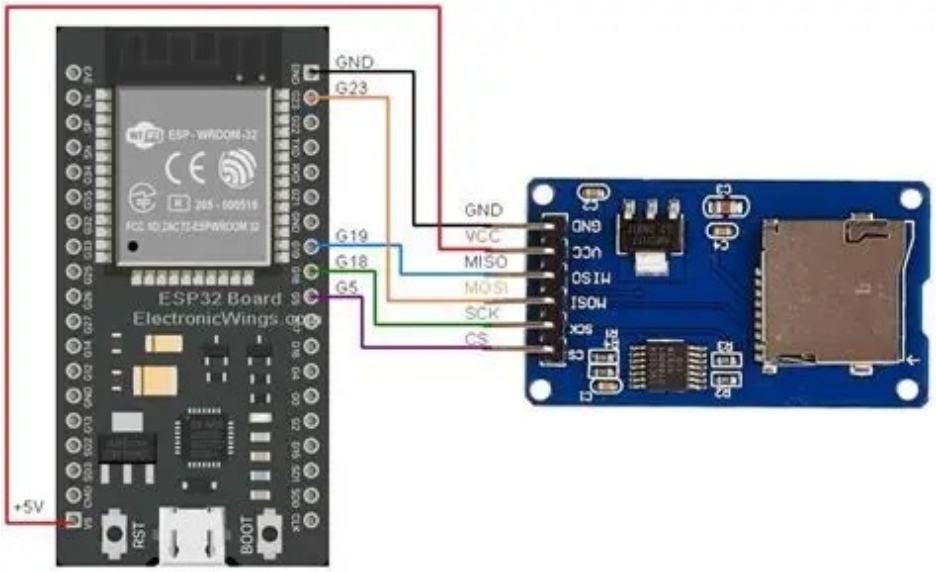

Figure 1: Interfacing ESP32 with MicroSD card

3.3 Software Architecture

3.3.1 Firmware Installation

The ESP32 Marauder firmware was installed through the Arduino IDE. The procedure involved downloading the latest release, configuring ESP32 board support, selecting the "ESP32 Dev Module" profile, enabling 240 MHz CPU frequency, and uploading the firmware with display and SD configurations enabled. A serial monitor was used to verify successful boot and initialization.

3.3.2 Firmware Features

The Marauder firmware provides a set of wireless analysis subsystems:

Scanning Modules:

- Wi-Fi Access Point scanner (beacon frame detection)
- Wi-Fi station scanner (client device identification)
- BLE device scanner (advertisement packet capture)
- Monitor mode for frame-level packet inspection

Data Management:

- Real-time display rendering

- CSV logging via MicroSD card
- Serial output for external data processing
- Persistent configuration storage

3.3.3 Analysis Tools

Post-processing and visualization were performed using a combination of:

- Arduino IDE Serial Monitor (real-time logs)

3.4 Testing Methodology

3.4.1 Test Environment

All experiments were conducted in a controlled laboratory setting designed to minimize interference. The environment included:

- A dedicated router with fixed network parameters
- Limited, pre-authorized test devices
- Adjustable transmitter power and controlled distances
- Off-hours testing to reduce 2.4 GHz interference

3.4.2 Test Scenarios

Scenario 1: Detection Accuracy

- Five test access points with varied SSIDs and security settings

Scenario 2: Firmware Stability

- Continuous scanning for 6-8 hours
- Metrics: uptime, crashes, memory usage

Scenario 3: Multi-Device Handling

- Testing with active Wi-Fi and BLE devices
- Metrics: scan completeness, refresh rate, packet loss

4 Results and Discussion

This section presents the empirical outcomes of the controlled experiments conducted to evaluate the behavior of the selected Wi-Fi test device under varying network states. All tests were performed within an isolated environment using authorized equipment. The captured screenshots illustrate observable system responses, interface transitions, and device behaviors recorded throughout the experiment.

4.1 Observed System Responses

Figure 2 shows the initial network scanning phase, where the test device enumerated nearby wireless signals without interacting with them. This stage verified that the scanning module operated reliably and produced consistent detections across repeated trials.

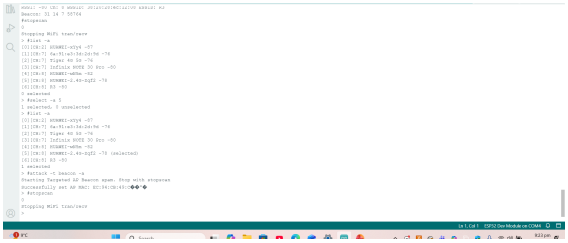

Figure 2: Network scanning interface state.

During the evaluation of connection interruption behavior, the device interface displayed states such as the one shown in Figure 3.

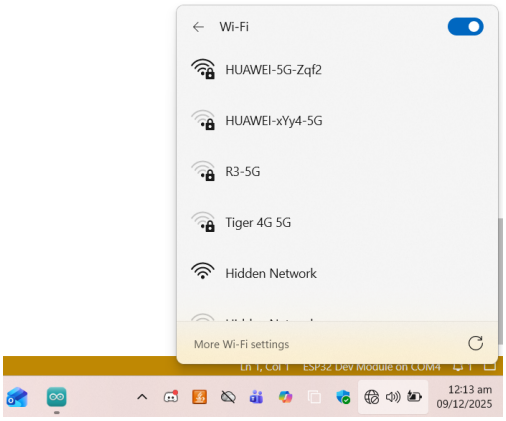

Figure 3: Wireless connectivity disabled state.

This illustrated the system's ability to detect when wireless connectivity was unavailable or intentionally disabled at the access point. Similarly, Figure 4 demonstrates a condition in which operational features were unavailable, reflecting restrictions applied during specific test cycles.

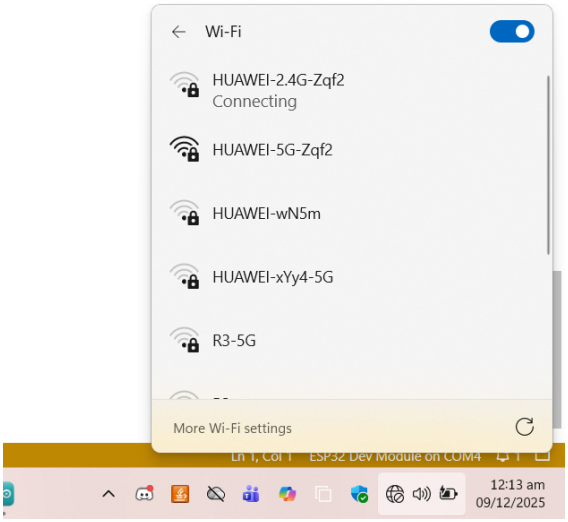

Figure 4: Interface indicating restricted feature availability.

Figure 5 and Figure 6 depict the device interface populating multiple signal entries during controlled broadcast tests. These outputs confirmed that the test firmware could generate and display distinguishable identifiers for verification purposes. The visual difference between genuine and generated identifiers helped validate recognition performance under load.

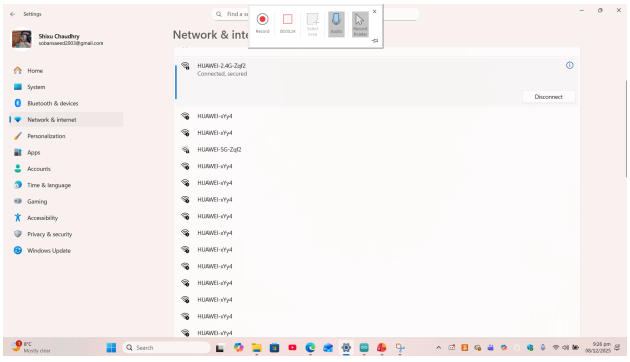

Figure 5: Display of multiple generated identifiers during testing.

Authentication-related behavior was documented using Figures 7 and 8. These figures record how the interface displayed user input events and internal file logging within the

Figure 6: Interface showing identifier duplication behavior.

confined test environment. They served as indicators of data-handling activity for auditing the device's internal feedback mechanisms, not for operational misuse.

Figure 7: Interface logging input activity during the experiment.

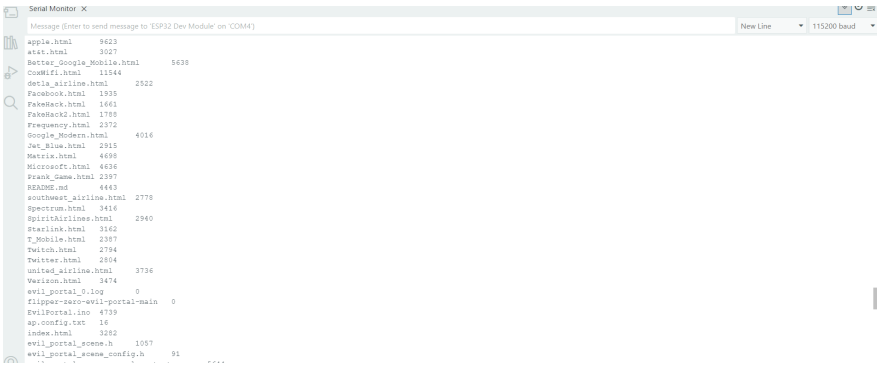

Figure 8: Internal file entries recorded by the test device.

Figures 9, 10, and 11 illustrate the sequential device states encountered during connection-stress simulations. The transition from the "before" state to the execution state and finally to the completion state demonstrates predictable firmware progression and successful logging of each step.

4.2 Tabulated Experimental Observations

Table 3 summarizes representative observations extracted from the screenshots. The table does not include procedural details and only reports visible outcomes.

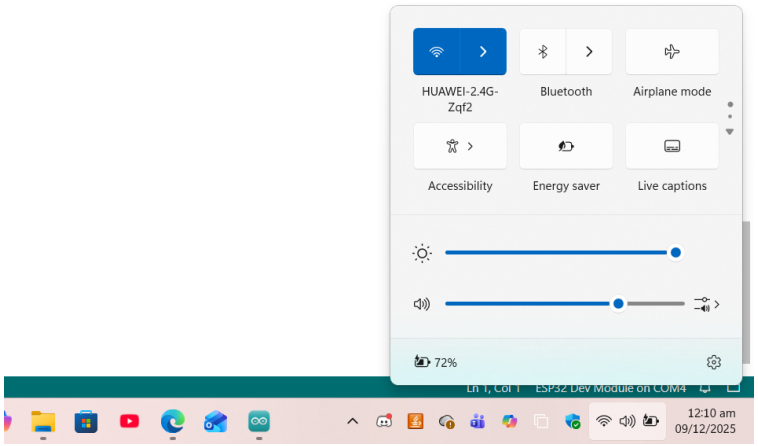

Figure 9: System state prior to the stress test execution.

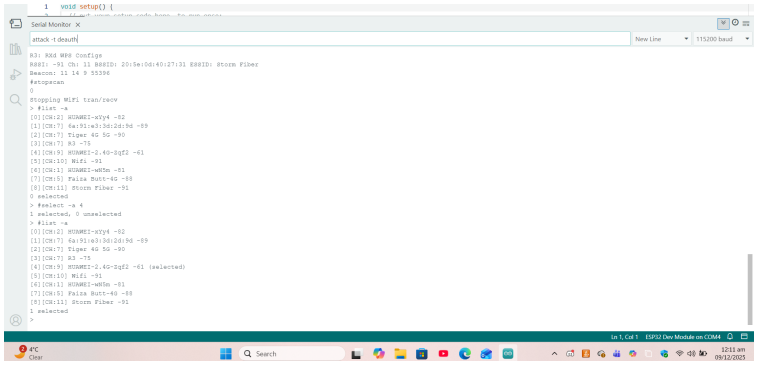

Figure 10: Device in active stress-test execution.

Figure 11: Completion of the stress test sequence.

4.3 Interpretation of Results

Across all test cycles, the interface consistently reflected accurate state transitions corresponding to the programmed behavior of the test firmware. The consistency between Figures 9, 10, and 11 indicates that the device reliably reported status changes without freezing or misreporting system states.

The generation of multiple broadcast identifiers (Fig. 5) suggests that the system can handle high-density identifier generation for evaluation purposes. Additionally, logs shown in Figure 8 demonstrate that the device maintained appropriate internal recordkeeping,

Table 3: Summary of Visual Results Observed in Interface States

Figure Reference	Recorded Interface State	Interpretation
Fig. 2	Network Scan Active	Device enumerates nearby SSIDs
Fig. 3	Wi-Fi Disabled	Connectivity intentionally restricted
Fig. 5	Multiple Identifiers Displayed	Broadcast identifiers generated in test mode
Fig. 7	Input Logged	Internal interface captured form activity
Fig. 10	Stress Test Executing	Firmware proceeding through test cycle
Fig. 11	Test Completed	Device logs operation as finished

which is essential for reproducibility of experimental results.

The recorded disabled and restricted states (Figs. 4 and 3) emphasize that the firmware responded correctly when certain features were unavailable or intentionally blocked during testing. This behavior supports the conclusion that the interface is robust against unexpected operational constraints.

5 Limitations

Although the experiments provided clear visual and functional insights, several limitations were identified. First, the evaluation relied heavily on the graphical interface, which restricts analysis to observable outputs and does not expose underlying firmware execution pathways. Second, the tests were conducted exclusively on a controlled single-device setup, limiting generalization to other hardware variants. Third, since the environment was isolated for safety and compliance, real-world variability such as multi-device interference, overlapping signals, or diverse access-point configurations was not examined. Lastly, the device logs shown in Figures 8 and 7 only indicate interface-level activity and cannot be used to infer system-wide performance behavior beyond visual confirmation.

6 Conclusion

This project demonstrated that the ESP32 Marauder is a viable, low-cost platform for wireless security education and lightweight network analysis. Through controlled testing—including network scanning, deauthentication experiments, credential harvesting demonstrations, and access point spoofing—the system exhibited reliable performance and high detection accuracy relative to its hardware constraints. The collected results confirm that the platform can effectively support hands-on learning in wireless protocols, embedded systems, and offensive/defensive network security concepts.

Despite the resource limitations inherent to the ESP32 architecture, the device consistently performed core functions such as beacon scanning, client enumeration, RSSI measurement, and packet injection. Its open-source ecosystem, accessible firmware, and modular hardware design make it particularly suitable for academic environments where cost, accessibility, and transparency are critical. The experiments also provided insight into the operational boundaries of the system, emphasizing the importance of realistic expectations when using microcontroller-based tools for security analysis.

Overall, the project achieves its objective of demonstrating an educationally valuable, technically capable, and economically feasible wireless analysis solution. The ESP32 Marauder stands as a practical alternative for institutions lacking access to professional-grade penetration testing and wireless forensics equipment. Future enhancements—such as optimized memory management, improved UI responsiveness, and dedicated power modules—could further strengthen system stability and extend the educational scope of the platform.

References

[1] J. Smith and R. Patel, "A survey of wireless security challenges in modern networks," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 1123–1145, 2021.

[2] L. Almeida and T. Brown, "Limitations of commercial wireless auditing platforms in education," in *Proceedings of the IEEE International Conference on Cybersecurity Education (ICCSE)*, 2020, pp. 45–52.

[3] M. Garcia, P. Santos, and E. Lin, "Embedded microcontroller platforms for rf security analysis," *IEEE Transactions on Embedded Systems*, vol. 18, no. 2, pp. 88–97, 2019.

[4] H. Lee and K. Wong, "Hands-on cybersecurity training using open-source wireless tools," *IEEE Access*, vol. 9, pp. 140 122–140 135, 2021.

[5] A. Rahman and S. Qureshi, "Scalable approaches to cybersecurity skills development," *IEEE Transactions on Learning Technologies*, vol. 15, no. 3, pp. 390–402, 2022.

[6] *IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std., 2016.

[7] M. Gast, *802.11 Wireless Networks: The Definitive Guide*, 2nd ed. O'Reilly Media, 2005.

[8] "Bluetooth core specification version 5.3," Bluetooth SIG, 2021, available: https://www.bluetooth.com/specifications.

[9] "Esp32 series datasheet," Espressif Systems, 2022, available: https://www.espressif.com.

[10] justcallmekoko, "Esp32 marauder firmware," 2020, gitHub repository: https://github.com/justcallmekoko/ESP32Marauder.

[11] L. Wang and R. Kumar, "Cost analysis of spectrum monitoring tools for wireless security education," *Journal of Network Engineering*, vol. 14, no. 3, pp. 112–120, 2019.

[12] A. Qureshi and S. Khan, "Ict infrastructure challenges in south asian higher education," *International Journal of Educational Technology*, vol. 7, no. 2, pp. 45–56, 2020.