PROGRESS IN ENGINEERING APPLICATION AND TECHNOLOGY
e-ISSN: 2773-5303

PEAT

Vol. 7 No. 1 (2026) 348-356
https://publisher.uthm.edu.my/periodicals/index.php/peat

Real-Time Phishing Detection Using Microcontroller and Machine Learning

Umie Maizatul Akma Kamarudin[1], Mohd Hezri Mohd Lwi@Mokhtar[1]*

[1] *Department of Electrical Engineering Technology, Faculty of Electrical Engineering Technology Universiti Tun Hussein Onn Malaysia, 84600, Pagoh, Johor, MALAYSIA*

*Corresponding Author: hezri@uthm.edu.my
DOI: https://doi.org/10.30880/peat.2026.07.01.040

Article Info

Received: 12 January 2026
Accepted: 20 February 2026
Available online: 25 May 2026

Keywords

Phishing Detection, Machine Learning, ESP32, Real-Time Monitoring, IoT.

Abstract

Phishing attacks pose a significant threat to online security by tricking users into revealing sensitive information through deceptive website. This project proposes a real-time phishing detection system that integrates machine learning with a microcontroller-based platform to provide a fast, low cost and automated solution. The system uses an ESP32 microcontroller to capture URLs in real-time and transmit them to a machine learning model hosted on a cloud-based Flask server. The model analyzes each URL to classify it as either phishing or legitimate based on learned patterns and features. Experiment results demonstrate the system's effectiveness in detecting phishing websites with high accuracy and responsiveness, using visual alerts via an OLED display and physical warning through LEDs and buzzers. The proposed approach offers a portable and scalable cybersecurity solution especially useful for environments with limited computing resources. Future enhancements may include the integration of deep learning models and mobile interfaces for wider usability.

1. Introduction

Phishing attacks are among the most common cybersecurity threats in the digital environment. These attacks were easy to create and highly effective. Attackers commonly design fake websites that closely resemble legitimate ones. This technique was known as social engineering, where users were manipulated into revealing sensitive information such as usernames, passwords, and banking details [1]. As online services increased, phishing continued to pose serious risks to individuals and organizations.

In recent years, phishing techniques have become more advanced with the use of automation and artificial intelligence. Automation refers to the use of software tools to generate phishing websites quickly and at a large scale. Artificial intelligence was used to imitate the layout, design, and behavior of legitimate websites with high accuracy. As a result, phishing websites became more convincing and harder to detect using visual inspection alone [2].

Traditional security solutions such as antivirus software, firewalls, and email spam filters were widely used to reduce phishing attacks. Most of these solutions relied on signature-based detection, which identified threats by matching known patterns. This approach was ineffective against newly created or unknown phishing websites. To address this limitation, machine learning techniques were applied in phishing detection. Machine learning refers to systems that learn patterns from data to make predictions. These methods analyzed features such as URL structure, website content, and HTML code to classify phishing websites [3].

Many machine learning-based phishing detection systems were implemented using cloud or server-based architectures. These architectures required high computational power, large memory, and continuous internet connectivity [5]. Due to these requirements, such systems were unsuitable for real-time deployment in resource-

constrained environments. Resource-constrained environments refer to systems with limited processing power, memory, and energy, such as embedded systems and IoT devices.

With advancements in embedded hardware, microcontroller-based platforms became capable of supporting lightweight machine learning inference. A microcontroller was a small, low-power computing unit commonly used in embedded applications. Frameworks such as TensorFlow Lite for Microcontrollers enabled trained models to run on devices like the ESP32 [6]. Therefore, this research aimed to develop a real-time phishing detection system using a lightweight machine learning model deployed on a microcontroller platform. The system was designed to operate independently without cloud support. The goal was to provide an efficient, low-cost, and energy-efficient phishing detection solution for embedded and IoT environments.

2. Methodology

This section describes the structured development of a real-time phishing detection system designed for resource-constrained environments. The methodology integrates hardware prototyping with embedded machine learning to identify malicious URLs without relying on heavy external infrastructure.

2.1 Materials

The hardware development for this real-time phishing detection system is centered on the ESP32 microcontroller, which serves as the core processing unit. This device was selected specifically for its dual Wi-Fi and Bluetooth capabilities, low power consumption, and its ability to handle lightweight machine learning inference locally. To facilitate user interaction and provide immediate feedback, a 0.96-inch OLED display is utilized to present classification results such as "Phishing Detected" or "URL is Safe" in real time.

The physical alert mechanism is further reinforced by an active buzzer for audio warnings and a set of Red and Green LED indicators. The buzzer provides an auditory signal that ensures user awareness even when the display is not being monitored. The LEDs function as high-visibility indicators of the URL's safety status, where a green light signifies a safe link and a red light warns of a potential threat. Supporting components such as 220-ohm resistors, breadboards, and jumper wires are used to ensure stable electrical connections and regulate current flow to prevent component burnout.

From a software perspective, the Arduino IDE serves as the primary development environment for writing, compiling, and uploading the system logic to the ESP32. To enable intelligent classification, TensorFlow Lite for Microcontrollers is employed to run an optimized Random Forest model directly on the hardware. Additionally, the system integrates Google Apps Script and a cloud-based Flask API to provide a hybrid detection layer, allowing the device to cross-reference global databases and log scanning events to a remote Google Sheets dashboard for continuous monitoring.

2.2 Procedure and System Integration

The development process began with a structured design phase, utilizing Wokwi simulation software to validate the circuit architecture and programming logic. This simulation allowed for rigorous testing of input and output behaviors, such as LED and buzzer activation, before the physical implementation of the prototype. By identifying logical errors in a virtual environment, the research ensured a higher degree of stability and reduced the risk of hardware failure during the actual assembly phase. The overall sequence of these research steps is documented in Fig. 1.

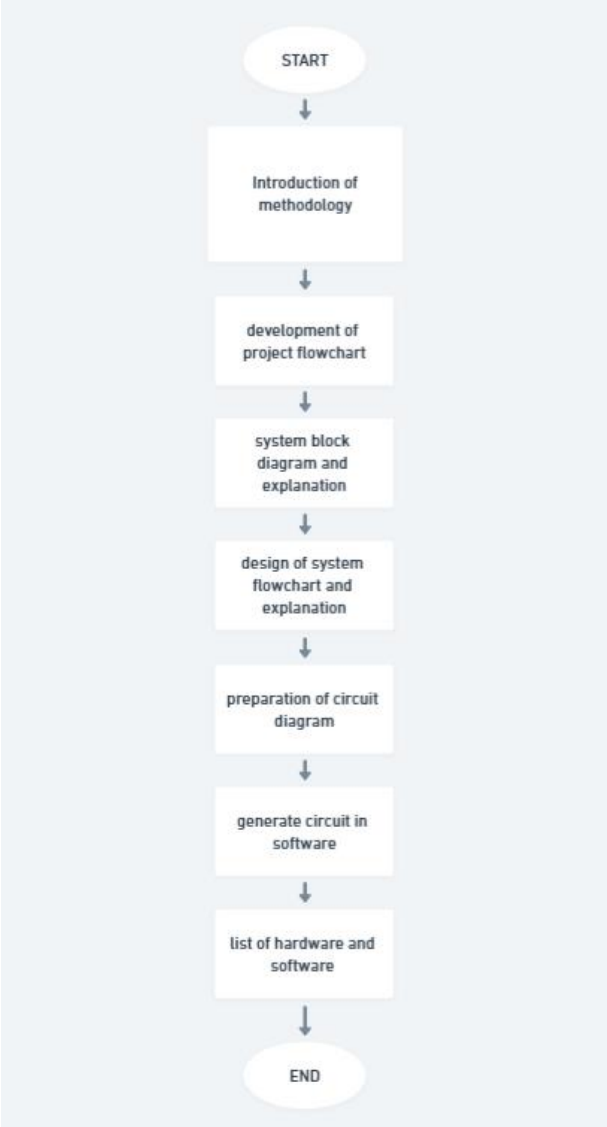
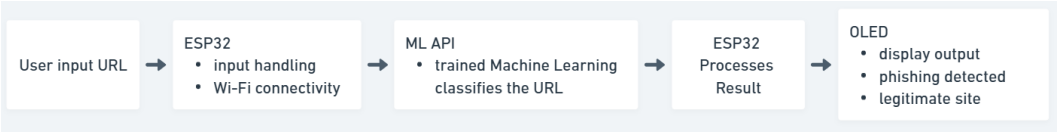


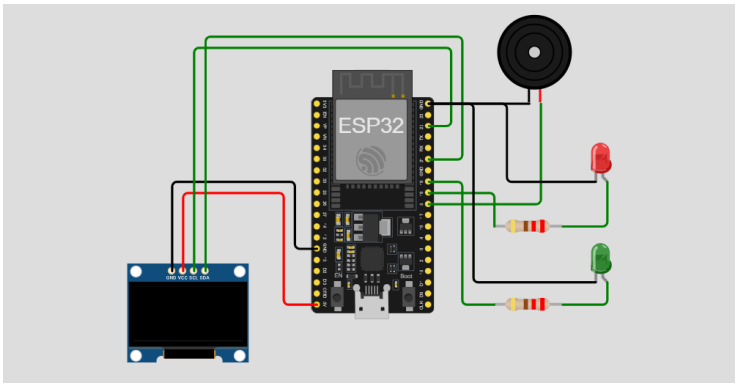
Fig. 1 *Flowchart of Methodology*

Following validation, the physical circuit was assembled by interfacing the OLED via the I2C bus using GPIO 21 and 22, while the status indicators were mapped to digital outputs GPIO 18, 19, and 5. The integration of these hardware and software components is detailed in Fig, 2. The system is designed to accept URL inputs through two primary methods, a serial monitor for developer testing or a dedicated web interface for general user interaction.



(a)

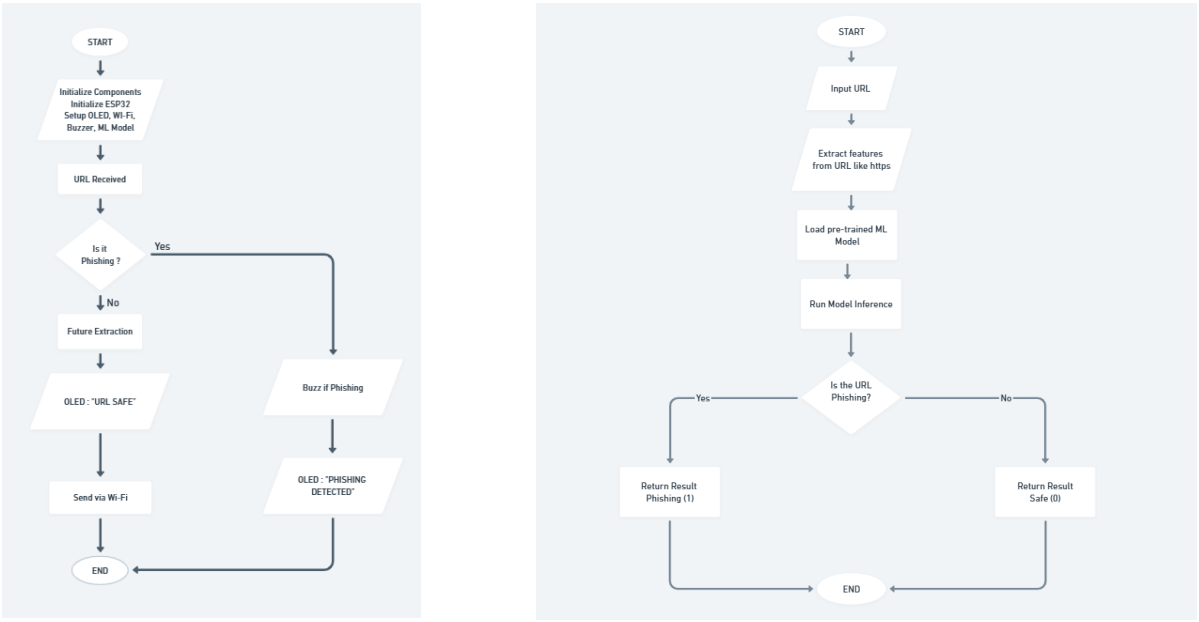
Progress in Engineering Application and Technology Vol. 7 No. 1 (2026) p. 348-356



(b)

Fig. 2 *Integration (a) Block Diagram; (b) Circuit Design*

The operational workflow of the real-time phishing detection system follows a continuous cycle aimed at immediate threat mitigation, which is effectively visualized in Fig.3 by combining the overall system process and the specific internal classification logic. The system flowchart in part (a), illustrates the high-level operational sequence starting from component initialization and Wi-Fi setup to the reception of a URL input. Once a URL is received, the system enters a decision phase to evaluate its safety, if classified as a threat, the ESP32 immediately triggers localized alert mechanisms, activating the buzzer and red LED while displaying a "PHISHING DETECTED" warning on the OLED screen. Complementing this, part (b) provides the details of the decision-making logic where the ESP32 extracts key lexical and host-based features, such as SSL status, domain age, and suspicious keywords, to be evaluated by the pre-trained Random Forest model. This model runs a real-time inference to assign a binary result, where a return value of **1** indicates a phishing attempt and **0** indicates a safe site, thereby ensuring that the hardware responses are driven by accurate, data-driven patterns. Any detected incident data is then logged and transmitted via Wi-Fi to a remote database for further monitoring and model improvement, completing the cyclical defense process.



(a) **(b)**

Fig. 3 *Logic and Operational of the Phishing Detection System (a) System Flowchart; (b) ML Model Flowchart*

A hybrid processing approach was implemented to balance speed and accuracy. Lightweight inference is performed locally on the ESP32 for immediate response, while complex URLs are offloaded to a cloud-based server for deeper analysis. This dual-layer strategy allows the device to function in various settings, including environments with limited network access.

2.3 Machine Learning and Characterization

System characterization involved a series of experimental tests to verify the performance and reliability of the hardware components. The ESP32 was extensively characterized based on its power stability, Wi-Fi connectivity range, and its ability to process data entry without delay. Testing confirmed that the indicator components, the OLED, LEDs, and buzzer responded correctly according to the system logic, providing both visual and auditory confirmation of the detection results.

The software's classification capabilities were characterized by testing the system against various URL categories, including whitelisted domains (e.g., google.com), adult content, and known malicious links. Characterization metrics included detection accuracy, response latency, and memory consumption on the ESP32 platform. These tests proved that the system could effectively distinguish between safe and suspicious content using a combination of local heuristics and cloud-based verification.

A critical aspect of characterization was the identification of system limitations in different network environments. Evaluations conducted on the UTHM campus Wi-Fi revealed that strict institutional security policies could block outbound HTTP requests, leading to database log errors (HTTP code -1), as shown in Fig. 4. Conversely, testing on private hotspots demonstrated full functionality, confirming that while the system is highly effective, its cloud-dependent features require an unrestricted network to operate at maximum capacity.

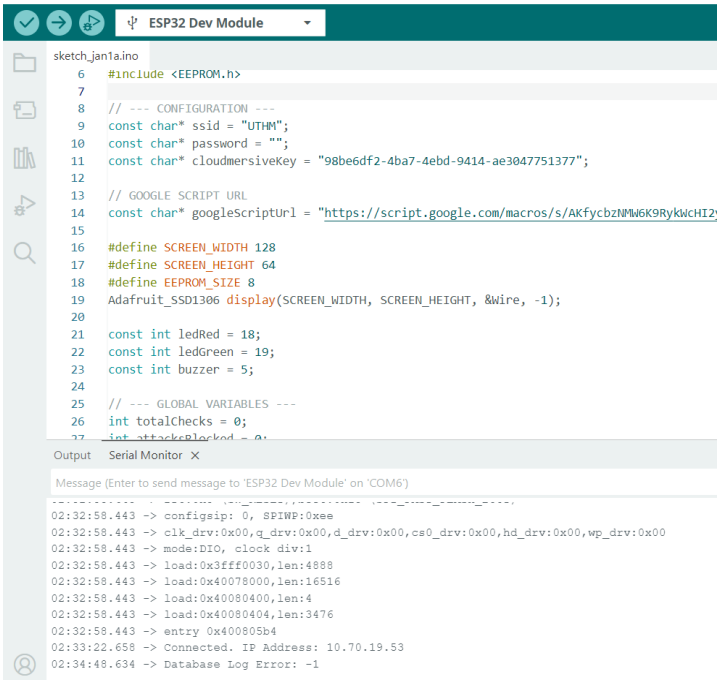


Fig. 4 *Serial monitor output demonstrating the HTTP code -1 error caused by network restrictions*

2.4 Data Analytic Strategy

The data analytic strategy utilizes a quantitative method to evaluate the overall effectiveness of the phishing detection system. All scanning events are recorded in real time and visualized through a Live Monitoring Dashboard, which tracks critical parameters such as the device IP, access timestamp, scanned URLs, and the specific classification reason. These data logs are systematically synchronized with a cloud-based Google Sheet, as illustrated in Fig. 5, which serves as the primary data repository for further statistical analysis and performance tracking. In addition to classification results, the strategy includes monitoring the Received Signal Strength Indicator (RSSI), measured in dBm, to analyze the impact of signal quality on real-time data transmission reliability. Furthermore, an iterative refinement strategy is employed where user feedback from a push-button interface and manual override data are analyzed to retrain the model, thereby improving detection accuracy and reducing false-positive rates over time.

Progress in Engineering Application and Technology Vol. 7 No. 1 (2026) p. 348-356

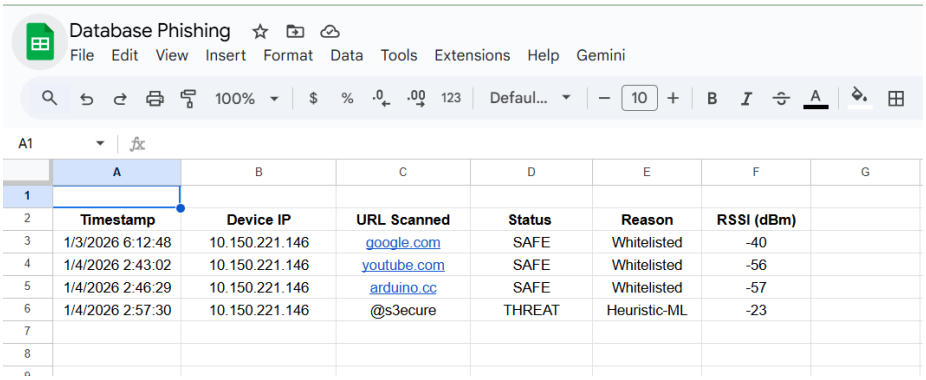


Fig. 5 *Google Excel Data Spreadsheet*

3. Result

The physical implementation of the phishing detection system demonstrates that an ESP32 microcontroller can effectively host intelligent security mechanisms for edge-based protection. Initial validation through Wokwi simulation ensured that the program structure and I/O behaviors were correct before the physical assembly of the prototype. Hardware results confirm that the ESP32 successfully manages Wi-Fi connectivity and processes URL inputs with high responsiveness, maintaining stable power throughout the operation. The system distinguishes between security conditions by providing immediate visual and auditory feedback. For instance, as summarized in Table 1 legitimate URLs trigger a green LED and a "SAFE" message on the OLED, while malicious links activate a red LED, a continuous buzzer alert, and a "BLOCKED" warning.

Table 1 *LED Indicator and Buzzer Status Based on System Condition*

Condition	LED Color	Buzzer Action	OLED Status
Wi-Fi Successfully Connected	Green (Blinks)	Silent	Displays IP Adress & “SYSTEM READY”
Whitelisted URL (e.g., google.com)	Green	Silent	Displays “SAFE” & “Whitelisted”
Clean Analysis Result (via Cloud API)	Green	Silent	Displays "SAFE" & "Verified Content"
Wi-Fi Failed / DANGER (Open)	Red	Beep Alert	Displays "DANGER" or "OFFLINE"
Adult Content (e.g., porn, gamble)	Red	Continuous Beep	Displays "BLOCKED!" & "Adult Content"
Malicious URL Detected (Cloud API)	Red	Continuous Beep	Displays "BLOCKED!" & "Cloud API"
Suspicious URL (High Heuristic Score)	Red	Continuous Beep	Displays "BLOCKED!" & "Heuristic-ML”

The hardware development setup, as shown in Fig, 6, serves as the primary testing environment where the device transitions through several operational states. When a user inputs a link, the system differentiates results using specific hardware responses. Fig. 7 illustrates these physical outcomes: a “URL Safe” result is marked by the activation of a green LED as seen in part (a), whereas a “Suspicious URL” identified through heuristic-based detection triggers a red LED and the alert buzzer as shown in part (b). The OLED display accurately presents system information, including network status, detection results, and system indicators, confirming proper integration between the physical components and the underlying detection logic.

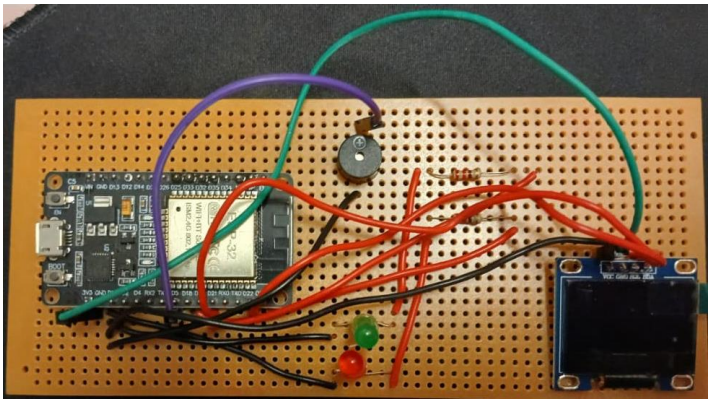
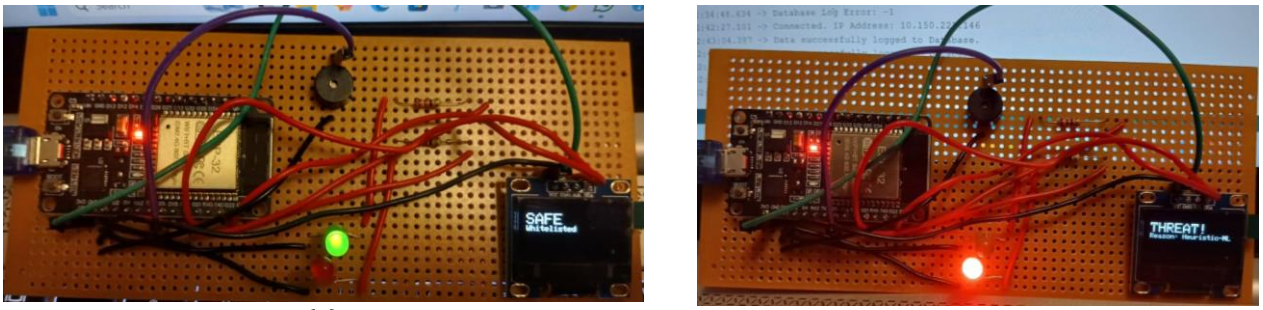


Fig. 6 *Hardwrae Development*

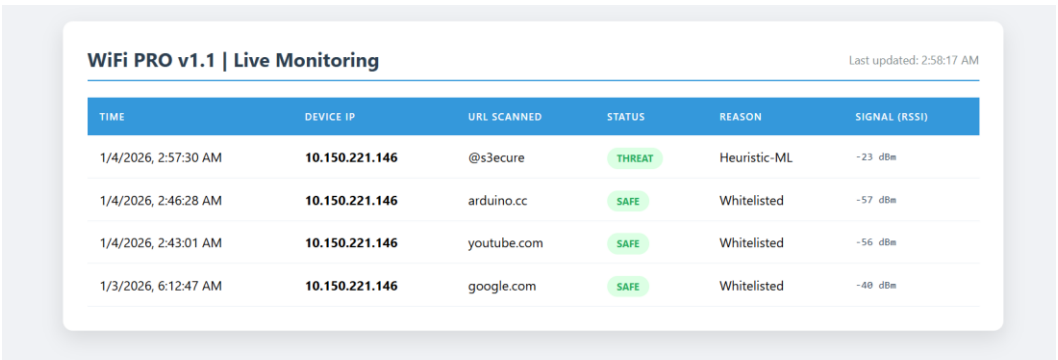


(a) **(b)**

Fig. 7 *Hardware response to classification results* (a) *URL Safe;* (b) *Suspicious URL*

Software testing validates that the system can classify diverse URL types, including whitelisted, adult, and malicious content, using a hybrid of local heuristic analysis and cloud-based verification. However, a critical limitation was identified during characterization on the UTHM campus Wi-Fi, where strict institutional firewalls blocked outbound HTTP requests. This network constraint caused a failure in data transmission to the remote server, resulting in an HTTP error code of -1 in the serial monitor as seen in Fig. 4. Despite this, full functionality was achieved using unrestricted private hotspots, confirming that the system's cloud-dependent features, such as real-time logging, require an open network environment to operate at maximum capacity.

The effectiveness of the detection logic is empirically supported by the Live Monitoring Dashboard, which logs real-time data into a Google Sheets database. As illustrated in Fig. 8, the system successfully captures and organizes critical parameters, including access to timestamps, device IP addresses, specific classification reasons (e.g., "Heuristic-ML"), and the Received Signal Strength Indicator (RSSI). For example, a scanned URL such as "@s3ecure" was identified as a threat and logged with an RSSI of -23 dBm, proving the system's ability to track network health alongside security metrics. This data analytic strategy, combined with a push-button interface for user feedback, creates an iterative refinement loop that allows the machine learning model to adapt and improve its accuracy over time.



WiFi PRO v1.1 | Live Monitoring

Last updated: 2:58:17 AM

TIME	DEVICE IP	URL SCANNED	STATUS	REASON	SIGNAL (RSSI)
1/4/2026, 2:57:30 AM	10.150.221.146	@s3ecure	THREAT	Heuristic-ML	-23 dBm
1/4/2026, 2:46:28 AM	10.150.221.146	arduino.cc	SAFE	Whitelisted	-57 dBm
1/4/2026, 2:43:01 AM	10.150.221.146	youtube.com	SAFE	Whitelisted	-56 dBm
1/3/2026, 6:12:47 AM	10.150.221.146	google.com	SAFE	Whitelisted	-40 dBm

Fig. 8 *Dashboard Database*

4. Discussion

The experimental findings confirm that the integration of machine learning with a microcontroller platform offers a viable, low-cost solution for real-time phishing mitigation at the network edge. A key strength of the developed system is its hybrid detection architecture, which utilizes local heuristic analysis for immediate response while leveraging a cloud-based Flask API for deeper classification. This approach addresses the inherent memory and processing constraints of the ESP32, allowing the device to perform complex URL evaluations that would otherwise be impossible on a standalone microcontroller. The hardware responsiveness, as evidenced by the synchronized activation of the OLED, LEDs, and buzzer, demonstrates that embedded systems can provide user-centric security feedback comparable to traditional software-based solutions.

The performance of the Random Forest model and heuristic logic highlights the effectiveness of lexical feature extraction specifically focusing on domain age, SSL status, and suspicious keywords. While deep learning models often achieve higher accuracy, the results of this study align with previous research suggesting that lightweight, feature-engineered models are more suitable for real-time execution in IoT environments due to their low computational overhead. Furthermore, the implementation of a push-button feedback loop introduces a human-in-the-loop element, which is critical for reducing false positives and adapting to the rapidly evolving tactics used in modern phishing campaigns.

However, the evaluation in restricted network environments, such as the UTHM campus Wi-Fi, exposed a significant dependency on outbound connectivity. The occurrence of the HTTP -1 error code indicates that strict institutional firewalls can compromise the system's ability to log data and access cloud-based intelligence. This finding suggests that for true autonomy in highly secured or isolated environments, future iterations must prioritize "Edge-AI" by further optimizing on-device models to run entirely offline. Despite these connectivity challenges, the system's ability to maintain basic heuristic protection ensures a baseline level of security, proving its robustness as a portable cybersecurity tool.

5. Conclusion

In conclusion, this project successfully developed and implemented a real-time phishing detection system that bridges the gap between machine learning and embedded systems. By integrating the ESP32 microcontroller with a Flask-based server, the project demonstrates an efficient pipeline where URL inputs are processed and classified with high accuracy using a trained machine learning model. The hardware architecture, featuring OLED displays, LEDs, and buzzers, provides a robust and responsive alert mechanism that validates the system's practical utility in real-world scenarios. Furthermore, the methodology highlights the critical role of advanced simulation tools like Wokwi in overcoming the technical constraints of traditional platforms for Wi-Fi-enabled testing. Ultimately, this study proves that lightweight machine learning models can be effectively deployed on low-power, resource-constrained platforms without compromising security performance. This work offers a significant contribution to the field of cybersecurity by providing a portable, low-cost, and autonomous solution tailored for the growing demands of IoT and edge-computing environments.

Acknowledgement

This project received support from Universiti Tun Hussein Onn Malaysia (UTHM). The author expresses gratitude to the Faculty of Engineering Technology at UTHM for providing the essential facilities, technical resources and a conducive environment that made this research possible. Additionally, appreciation is extended to the supervisor for his unwavering commitment to assist and guide throughout the project, generously sharing his knowledge and expertise.

References

[1] R. Zieni, L. Massari, and M. C. Calzarossa, “Phishing or Not Phishing? A Survey on the Detection of Phishing Websites,” *IEEE Access*, vol. 11, pp. 18499–18519, 2023, doi: https://doi.org/10.1109/access.2023.3247135.

[2] *Zscaler ThreatLabz 2023 Phishing Report*. (n.d.). Retrieved January 4, 2026, from https://www.egis-security.com/resources/Zscaler%20ThreatLabz%202023%20Phishing%20Report.pdf

[3] A. Correa Bahnsen, I. Torroledo, L. Camacho, and S. Villegas, “DeepPhish: Simulating Malicious AI.” Available:https://albahnsen.wordpress.com/wp-content/uploads/2018/05/deepphish-simulating-malicious-ai_submitted.pdf

[4] APWG, "APWG | Phishing Activity Trends Reports," *Apwg.org*, 2024. https://apwg.org/trendsreports/

[5] Ahmad, S., Zaman, M., AL-Shamayleh, A. S., Tanzila Kehkashan, Ahmad, R., Safi', Ergen, I., & Adnan Akhunzada. (2024). Across the Spectrum In-Depth Review AI-Based Models for Phishing Detection. *IEEE Open Journal of the Communications Society*, 1–1. https://doi.org/10.1109/ojcoms.2024.3462503

[6] Wainbuch, R., & Samuel, A. J. (2024). *TinyML: Deploying Machine Learning on Microcontrollers for IoT Applications*. *Journal of Science, Technology and Engineering Research*, 2(2), 44–57. https://doi.org/10.64206/d8sh8k34