NATIONAL ENGITECH DIGEST

VOL. 3 ISSUE 1

e-ISSN: 3030-5969

SecureEye: An IoT-Based Smart Anti-Theft Security System with Real-Time Surveillance and Telegram Notification

Khairul Anuar Bin Mohd Radzali[1], Kajaneesvaran A/L Murugiah[2], Mark Magilzhnan Mathew[3], Muhammad Ikmal Zuhairi Bin Akhdiat Miharja[4]

[1234]Jabatan Teknologi Maklumat dan Komunikasi, Politeknik Ungku Omar, Jalan Raja Musa Mahadi, 31400 Ipoh, Perak

kanuar@puo.edu.my[1]

Abstract

The increasing number of residential and commercial security incidents has created a demand for intelligent surveillance systems capable of providing real-time monitoring and immediate user notification. Conventional anti-theft alarm systems mainly rely on audible alarms without providing visual evidence, making it difficult for users to verify security incidents and respond promptly. This project presents **SecureEye**, an Internet of Things (IoT)-based smart anti-theft security system that integrates an ESP32-CAM module, motion detection sensor, Arduino Uno, ESP8266 Wi-Fi module, and Telegram Bot to provide real-time monitoring, image capture, and remote notification. The system was developed using the Agile software development methodology, allowing continuous refinement through iterative design, implementation, and testing. When motion is detected, the system activates an alarm, captures images of the monitored area, and sends notifications to authorised users through Telegram. System functionality was evaluated through unit testing and integration testing to verify user authentication, Telegram communication, security functions, notification delivery, and system configuration. The testing results demonstrated that the majority of system functions operated successfully according to the specified requirements, indicating that SecureEye provides a feasible IoT-based approach for remote security monitoring. The proposed system offers an affordable and practical security solution for residential and small commercial applications.

Keywords; ESP-CAM module; motion detection sensor; image capture; Telegram notification; Agile methodology.

Introduction

The increasing adoption of Internet of Things (IoT) technology has enabled conventional security systems to evolve towards connected solutions that support remote monitoring, automated detection, and real-time notification. In particular, IoT-based surveillance systems can combine motion sensors, cameras, wireless communication, and mobile applications to provide users with faster awareness of potential security incidents. Recent studies have demonstrated the feasibility of using low-cost ESP32-CAM platforms for real-time monitoring and intrusion detection, making IoT-based surveillance a practical option for residential and small-scale commercial applications [1] - [5]. However, conventional anti-theft alarm systems commonly rely on audible alarms and provide limited visual information about detected events, making it difficult for users to verify security incidents and respond promptly. This limitation can delay user response because users may be unable to verify whether an alarm represents an actual intrusion. Previous studies have addressed this issue by integrating ESP32-CAM with motion sensors, audible alarms, and remote notification mechanisms for security monitoring in rooms and commercial premises [2][4]. Other recent work has further demonstrated the use of ESP32-CAM with real-time notification to improve remote awareness of security events [3][5].

Nevertheless, many existing solutions focus primarily on specific monitoring or notification functions, creating an opportunity to develop a more integrated and user-oriented security platform that combines surveillance, alarm activation, authentication, and remote system management. SecureEye addresses this gap by developing an IoT-based smart anti-theft security system that integrates an ESP32-CAM, motion detection, audible alarm, wireless communication, and Telegram Bot interaction. The system aims to provide real-time surveillance and photographic evidence during security events, to alert users through an alarm and remote Telegram notification, and to provide a simple interface for authentication and system management. The system was developed using an Agile methodology and evaluated through unit and integration testing. The proposed system contributes a practical and relatively low-cost security solution that integrates detection, visual evidence, alerting, and remote interaction within a single platform, with potential application in residential and small commercial environments.

Methodology

SecureEye was developed using an iterative Agile approach consisting of requirement identification, system design, prototype development, testing, and refinement. This approach was selected to allow the hardware and software components to be progressively integrated and evaluated throughout the development process. Similar IoT security systems have demonstrated the feasibility of integrating ESP32-CAM and motion sensors for automated image capture and real-time notification [2][3].

a) System Design and Development

The development process began by identifying the functional requirements of SecureEye, which included real-time surveillance, motion detection, image capture, alarm activation, remote notification, user authentication, and system management. Based on these requirements, the system architecture was designed to connect the sensing, processing, surveillance, communication, and user-interaction components. The main hardware components selected were an Arduino Uno, ESP32-CAM, ESP8266 Wi-Fi module, motion sensor, buzzer, force sensor, power supply, and network equipment. The Arduino Uno was specified as the main controller for interfacing with sensors and peripherals, while the ESP32-CAM provided image capture and surveillance functionality. The ESP8266 was used to provide wireless connectivity, the motion sensor detected movement within the monitored area, and the buzzer provided a local audible alert. The force sensor enables access control functionality, such as detecting physical force applied to a door or entry point. These components were integrated according to the intended system requirements to support detection, alerting, image capture, and remote notification.

The software development was carried out using Arduino IDE, together with the required ESP8266, ESP32, and camera libraries. The communication architecture incorporated a Telegram Bot to provide remote interaction, user authentication, administrative commands, and security notifications. The system configuration involved establishing network connectivity and connecting the hardware and software components to support remote communication. The operating sequence of SecureEye begins when the motion sensor detects activity in the monitored area. The detected event triggers the security response, including alarm activation and image capture through the ESP32-CAM. The captured information can then be transmitted through the network for remote monitoring and notification. The Telegram Bot provides the user interface for authentication and selected system-management functions. This architecture follows the general approach demonstrated by recent ESP32-CAM security systems, where motion detection is used to trigger image capture and remote notification [2][3].

Motion Sensor → Arduino Uno / Controller → ESP32-CAM → ESP8266/Wi-Fi → Telegram Bot → User, with Buzzer connected to the controller as the local alarm.

Figure 1. Proposed SecureEye system architecture.

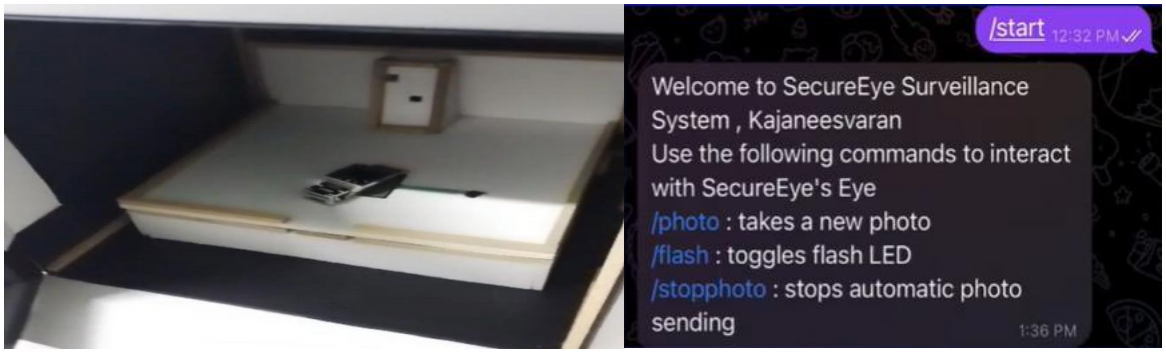

Figure 2. Physical design of SecureEye and Telegram notification.

b) System Testing and Evaluation

The testing was conducted according to predefined test cases, where each function was assessed by comparing its actual response with the expected result specified for that function. The evaluation focused on functional operation, authentication, communication, security handling, administrative management, and integration between the Telegram Bot and the SecureEye core system.

For user unit testing, five test cases were conducted: Telegram Bot initialization, user authentication, token generation and validation, error handling, and security features. All five test cases were recorded as Pass. For administrator unit testing, six test cases were conducted covering bot initialization, administrator authentication, role management, user management, system configuration, and security/error handling. Five tests were recorded as Pass, while the user management test was recorded as Failure. Integration testing was then performed to determine whether the Telegram Bot communicated correctly with the SecureEye core system. Four integration functions were tested: Telegram Bot and core-system authentication, administrator management, user authentication and authorization, and system configuration and monitoring. All four integration tests were recorded as Pass. The testing criteria were therefore based on successful completion of the expected function for each test case. The evaluation did not include numerical measurements of detection

accuracy, notification latency, or energy efficiency because such measurements were not reported in the technical report. Instead, the testing focused on verifying whether the implemented functions operated according to their specified requirements.

Results and Discussion

a) Functional Testing Results

User-level unit testing covering Telegram Bot initialization, user authentication, token generation and validation, error handling, and security features and indicates that the basic user authentication and security-related functions operated according to their expected outcomes. Administrator-level unit testing covering Telegram Bot initialization, administrator authentication, role management, user management, system configuration, and security and error handling. The combined unit testing result was 5 successful tests out of 6. The 1 failed test represents the main functional issue identified during the evaluation. Integration testing was subsequently conducted to verify communication between the Telegram Bot and the SecureEye core system including authentication, administrator management, user authentication and authorization, and system configuration and monitoring.

Table 1. Summary of SecureEye testing results

Evaluation	Test Cases	Passed	Failed	Pass Rate
User Unit Testing	5	5	0	100%
Administrator Unit Testing	6	5	1	83.3%
Integration Testing	4	4	0	100%
Overall Testing	15	14	1	93.3%

The results demonstrate that SecureEye successfully fulfilled most of its tested functional requirements. The 100% integration testing result is particularly important because it indicates that the tested Telegram Bot functions were able to communicate with the SecureEye core system as intended. However, the failed user management test demonstrates that the prototype still requires refinement before the system can be considered fully reliable for all administrative operations.

b) Discussion of System Performance

The successful integration results demonstrate the feasibility of using IoT communication for remote security management. This finding is consistent with previous ESP32-CAM-based security systems. Humam and Triawan [2] demonstrated the use of ESP32-CAM and motion detection for room security with real-time notification, while Cakrayuda et al. [3] integrated ESP32-CAM, motion detection, and Telegram notification for intelligent security monitoring. Similarly, Ramadhan et al. [4] developed an IoT-based security monitoring system for a commercial store using ESP32 DevKit V1, ESP32-CAM AI Thinker, an AM312 motion sensor, and a buzzer. Hidayah et al. [5] also demonstrated the integration of ESP32-CAM with real-time notification for an IoT-based home security system. These studies support the suitability of combining camera-based monitoring, motion detection, and remote notification for IoT security applications.

Compared with these previous approaches, SecureEye focuses on integrating security monitoring with user authentication, administrative management, and Telegram-based remote interaction within a single system. The successful integration testing in the present project indicates that the communication between the Telegram interface and the SecureEye core system operated as intended for the tested functions. However, unlike some previous studies that report specific performance indicators such as detection or notification performance, the current evaluation focused primarily on functional and integration testing. Therefore, further quantitative testing would be required to directly compare SecureEye's detection accuracy, notification latency, and reliability with existing systems.

c) Limitation and Unexpected Outcome

The principal unexpected outcome was the failure of the administrator (user management) unit testing. Since the other administrator functions, including authentication, role management, system configuration, and security/error handling, were successfully tested, the failure appears to be isolated to the user-management functionality rather than the overall Telegram-based administrative interface. This limitation indicates that further debugging and refinement are required in the implementation of user-management operations. Possible areas for improvement include checking the command-handling logic, validation of user information, database or storage operations, and communication between the Telegram interface and the corresponding management function. These possible causes are presented as areas for further investigation rather than confirmed causes

because the technical report does not identify the exact technical reason for the failed test. Another limitation identified in the project is dependence on Internet connectivity. Since remote communication and notification require network access, an unstable connection may affect remote monitoring and communication performance.

d) Overall Effectiveness and Future Improvement

Overall, 14 of the 15 documented test cases were successfully completed, including all four integration tests, resulting in an overall test-case pass rate of 93.3%. The results indicate that the majority of SecureEye's tested functions operated according to their intended requirements, while the failed user management test represents an area requiring further refinement. The results support the project's objective of developing an IoT-based security system that combines surveillance-related functions with remote system interaction. However, the results should not be interpreted as a 93.3% or 100% overall system accuracy because the evaluation was based on functional test cases rather than repeated measurements of detection accuracy, response time, or notification latency. Therefore, future evaluation should include repeated motion-detection trials, notification delivery success rate, end-to-end response time, image-capture success rate, false-positive rate, and operation under different network and lighting conditions.

Future improvements should also address the failed user-management function, improve network resilience, reduce false alarms, strengthen data protection, and consider backup power or alternative communication methods. Such improvements would increase the reliability and practical applicability of SecureEye in residential and small commercial environments.

Conclusion

SecureEye was successfully developed as an IoT-based smart anti-theft security system integrating motion detection, image capture, audible alarm, wireless communication, and Telegram-based remote interaction. The project objectives were achieved by developing the required security functions and demonstrating communication between the Telegram interface and the SecureEye core system through functional and integration testing. The results confirm that the majority of the implemented functions operated according to their intended specifications, demonstrating the feasibility of SecureEye as a practical remote security monitoring solution for residential and small commercial environments.

The main limitations identified were the unsuccessful user management function during unit testing and the system's dependence on Internet connectivity for remote communication. Future improvements should focus on resolving the user management issue, improving network reliability through alternative connectivity, enhancing motion detection to reduce false alarms, strengthening data security, and introducing backup power capabilities. Further testing under different environmental, lighting, and network conditions would also provide a more comprehensive evaluation of the system's reliability and practical performance.

References

[1] R. B. Salikhov, V. Kh. Abdrakhmanov, and I. N. Safargalin, "Internet of Things (IoT) Security Alarms on ESP32-CAM," *Journal of Physics: Conference Series*, vol. 2096, no. 1, p. 012109, 2021, doi: 10.1088/1742-6596/2096/1/012109.

[2] F. Humam and M. A. Triawan, "Sistem Keamanan Ruangan Menggunakan ESP32CAM dan Sensor Gerak Berbasis IoT," *Infotek: Jurnal Informatika dan Teknologi*, vol. 7, no. 2, pp. 575–584, 2024, doi: 10.29408/jit.v7i2.26109.

[3] L. Cakrayuda, M. R. Arhieadhie, and A. S. Putra, "SiCEMOT: Sistem Keamanan Cerdas Berbasis ESP32-CAM, Sensor Gerak, dan Notifikasi Telegram," *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 13, no. 2, pp. 1176–1185, 2025, doi: 10.23960/jitet.v13i2.6420.

[4] D. Ramadhan, A. Hakim, and D. R. Irawati, "Sistem Pemantauan dan Keamanan pada Toko Berbasis Mikrokontroler ESP32 DevKit V1, ESP32 CAM AI Thinker, Sensor AM312 dan Buzzer Berbasis IoT," *Journal of Information System, Informatics and Computing*, vol. 9, no. 1, pp. 191–199, 2025, doi: 10.52362/jisicom.v9i1.1926.

[5] Q. H. Hidayah, R. P. Laksana, A. Prabowo, and D. S. Simatupang, "IoT-Based Home Security System: ESP32-CAM Integration and Real-Time Notification," *International Transactions on Education Technology (ITEE)*, vol. 4, no. 2, pp. 149–161, 2026, doi: 10.33050/itee.v4i2.956.