Jurnal Elkolind Volume 12, Nomor 2, Juli 2025
DOI: http://dx.doi.org/10.33795/elkolind.v12i2/7631

# Implementation of a Smart Lock System with Real Time Visual Proof Transmission to Web

**Suci Fitriyah Darmayanti[1], Dewi Permata Sari[2], Renny Maulidda[3] , Aldi Satria Wansa[4]**
**e-mail: suciffdyy@gmail.com, dewi_permatasari@polsri.ac.id, rennymaulidda@polsri.ac.id, aldisatriawansa@gmail.com**
[1,2,3,4] Department of Electronics Engineering, Polytechnic of Sriwijaya, Palembang, 30139, Indonesia

**Informasi Artikel**

***Riwayat Artikel***

Diterima 25 Juni 2025
Direvisi 30 Juli 2025
Diterbitkan 31 Juli 2025

**Kata kunci:**

Kunci Pintu Pintar
Integrasi Multi-Sensor
Sistem Keamanan Berbasis IoT

**ABSTRAK**

Sistem keamanan akses yang andal tidak hanya membedakan antara pengguna sah dan tidak sah, tetapi juga mencatat setiap percobaan akses sebagai bukti digital. Penelitian ini mengembangkan sistem kunci pintu pintar berbasis multi-sensor yang mengintegrasikan RFID, sensor sidik jari, dan sensor getaran, serta dilengkapi dokumentasi visual otomatis menggunakan ESP32-CAM. Arduino Mega digunakan sebagai pengendali utama yang berkomunikasi secara serial dengan dua modul ESP32 untuk akuisisi data dan pengiriman gambar ke server Laravel. Sistem memberikan respons real-time terhadap autentikasi yang berhasil, percobaan akses yang gagal, dan deteksi gangguan fisik. Ketika kondisi tertentu terpenuhi, sistem secara otomatis mengambil gambar, mencatat waktu, dan mengirim data ke dashboard web. Hasil pengujian menunjukkan respons rata-rata 4–5 detik dari pemicu hingga unggahan gambar. Sistem ini menawarkan kombinasi keamanan fungsional dan visual, menjadikannya solusi praktis dan adaptif untuk aplikasi keamanan berbasis IoT.

***Keywords:***

Smart Door Lock
Multi-Sensor Integration
IoT-Based Security System

**ABSTRACT**

*A reliable access control system must distinguish between authorized and unauthorized users and record each access attempt as verifiable evidence. This study proposes an intelligent door lock system based on multi-sensor integration, combining RFID, fingerprint, and vibration sensors, enhanced with automatic visual documentation via ESP32-CAM. An Arduino Mega is the central controller, communicating serially with two ESP32 modules for data acquisition and image transmission to a Laravel-based server. The system provides real-time responses to successful authentications, failed access attempts, and physical disturbances. Upon such events, it automatically captures an image, timestamps the incident, and sends sensor data and visual evidence to a web-based dashboard. Testing results show a stable average response time of 4–5 seconds from trigger to image upload. This system offers functional and visual security, making it a practical and adaptive solution for IoT-based room protection.*

***Penulis Korespondensi:***
Dewi Permatasari,
Teknik Elektro,
Politeknik Negeri Sriwijaya,
Jl. Srijaya Negara, Sumatera Selatan, Palembang, Indonesia, 30139.
Email: dewi_permatasari@polsri.ac.id
Nomor HP/WA aktif: +62 895-4211-28594

## 1. INTRODUCTION

Physical access control remains critical in developing Internet of Things (IoT)-based systems, especially in security-sensitive environments such as homes, offices, and institutions. Numerous studies have explored the integration of RFID and fingerprint technologies to provide dual-factor authentication, which improves access accuracy and reliability. However, many systems still lack automated evidence capture and centralized monitoring.

In previous works, RFID and fingerprint-based access control systems have been implemented effectively using microcontrollers like Arduino and NodeMCU. While these systems identify authorized users well, most fail to address physical intrusion detection or incorporate visual monitoring to record access events [1], [2].

Several studies have attempted to enhance security by including vibration sensors that detect forced access or tampering. Despite this addition, they often rely on passive alerts like buzzers or SMS and rarely trigger visual documentation. Similarly, systems that utilize ESP32-CAM for image capture often function in isolation from sensor-based access logic, reducing their practicality for integrated security applications [3], [4], [5].

Administrators value web-based dashboards for monitoring access events. However, few existing implementations combine real-time photo logging, sensor status updates, and access verification in a unified Laravel-based backend. This gap is evident in research that either focuses only on logging RFID/fingerprint activity or only handles image capture without proper access validation [6], [7], [8], [9].

More advanced systems have integrated IoT modules with dashboard visualization, yet many limit their scope to two sensors or do not offer a response to failed authentication attempts. A small number of studies explore multi-sensor configurations or face recognition via ESP32-CAM but typically do not address broader scenarios like forced access or synchronization between multiple devices [10], [11], [12], [13], [14], [15].

Based on this literature review, the proposed system aims to fill these gaps by designing and implementing a smart door lock that integrates RFID, fingerprint, and vibration sensors with real-time photo capture using ESP32-CAM. All activity is logged and visualized through a Laravel web dashboard, making the system responsive to authorized access, failed attempts, and physical tampering.

## 2. RESEARCH METHODOLOGY

### 2.1 System Architecture Overview

This research adopts an experimental approach to design and implement an intelligent door lock system with multi-sensor authentication and real-time visual documentation. The methodology includes system design, microcontroller programming, communication protocol integration, and functional testing across multiple access scenarios.

The system integrates three types of sensors RFID, fingerprint, and vibration for access control and intrusion detection. An Arduino Mega is used as the central controller, connected via UART serial communication to two ESP32 modules: the ESP32 Doit V1, which handles data uploads to the Laravel-based server, and the ESP32-CAM, which captures and transmits photo evidence. All data is processed and stored by the Laravel backend, which provides a dashboard interface for monitoring.

The combination of ESP32 and Arduino microcontrollers has been widely adopted in similar IoT-based security systems due to their cost-efficiency and communication flexibility [1], [8]. The use of ESP32-CAM to capture and upload photos for event logging is also a proven practice in comparable systems [4].

The selection of Arduino Mega as the central controller is motivated by its multiple serial ports (UART), which simplify communication with multiple ESP32 modules simultaneously. This architecture ensures that sensor data and visual documentation processes can operate in parallel without data collision. Additionally, separating responsibilities where ESP32 Doit V1 focuses on data transmission, and ESP32-CAM handles image capture improves system modularity, fault tolerance, and maintenance flexibility.

The Laravel-based backend is a centralized data management, visualization, and system control platform. Through RESTful API endpoints, it receives JSON formatted sensor data and Base64-encoded images from ESP32 modules, storing them in a structured database. This backend not only enables real-time monitoring via web dashboard but also supports future scalability such as adding user authentication layers, alert notifications, or data analytics features.

## 2.2 Block Diagram of the System

Figure 1 illustrates the overall system architecture, showing its division into input, process, and output blocks.

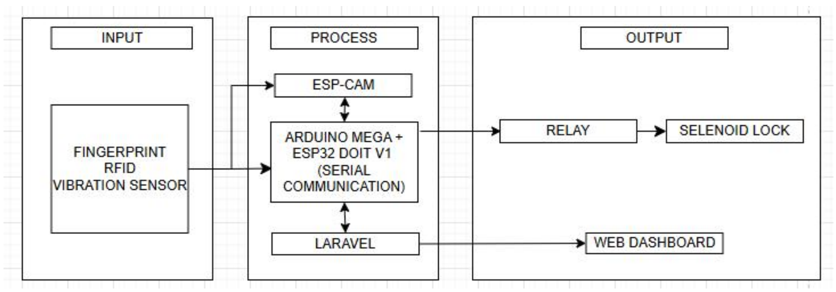


Figure 1 Block Diagram of the Smart Door Lock System

This diagram presents the system’s architecture in a visual block structure, clarifying the functional separation between input sensors, processing units, and output components. It supports the implementation by showing how each module interacts within the overall system.

## 2.2 Communication Flow

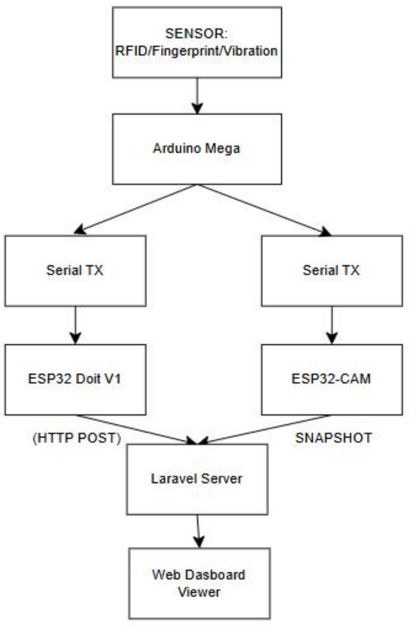


Figure 2 Communication Flow Between Arduino, ESP32 Modules, and Web Server

This section explains the communication flow between components. Input sensors transmit data to the Arduino Mega, which relays commands via serial to both ESP32 modules. The ESP32 Doit V1 sends access logs to the server using HTTP POST, while the ESP32-CAM receives "SNAPSHOT" commands and uploads images. The Laravel server stores the data in a database and displays it on a web-based dashboard.

This type of layered communication architecture has also been applied in Laravel-based smart monitoring systems [11].

## 2.3 System Flowchart

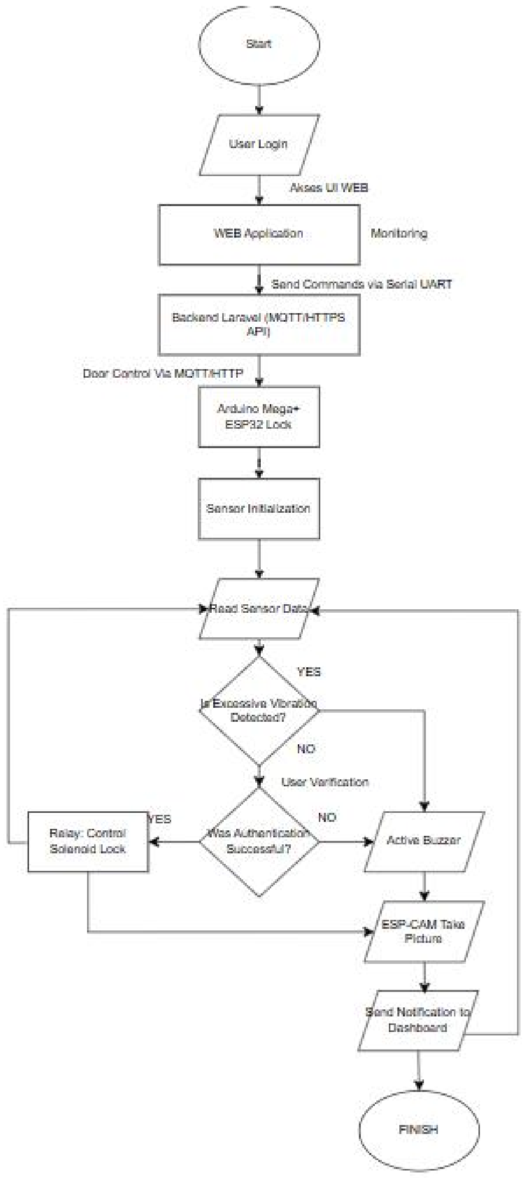
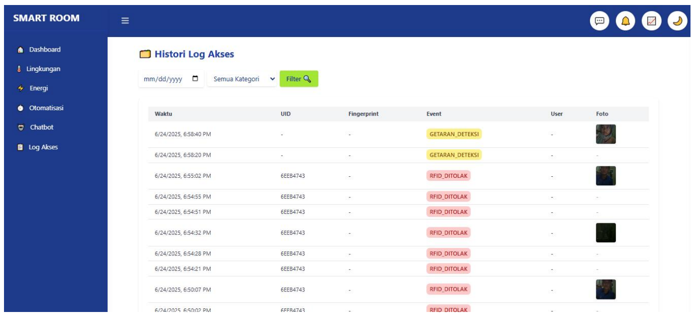

Start
User Login
Akses UI WEB
WEB Application
Monitoring
Send Commands via Serial UART
Backend Laravel (MQTT/HTTPS API)
Door Control Via MQTT/HTTP
Arduino Mega+ ESP32 Lock
Sensor Initialization
Read Sensor Data
YES
Is Excessive Vibration Detected?
NO
User Verification
Relay: Control Solenoid Lock
YES
Was Authentication Successful?
NO
Active Buzzer
ESP-CAM Take Picture
Send Notification to Dashboard
FINISH

Figure 3 System Flowchart of Smart Door Lock Control and Monitoring Logic

Figure 3 depicts the complete logic flow. After sensor initialization and data reading, the system checks for excessive vibration. If detected, the ESP32-CAM is triggered to capture and upload an image. If no vibration is detected, the system continues to RFID or fingerprint authentication. A valid input unlocks the door and triggers the camera, while failed attempts increment a counter. Upon three consecutive failures, the system activates the buzzer, captures an image, and sends an alert to the dashboard.

The use of a vibration sensor as a trigger mechanism to detect forced access or tampering enhances system security beyond traditional authentication-only systems [3]

## 2.3 Web Dashboard Interface



| Waktu | UID | Fingerprint | Event | User | Foto |
|---|---|---|---|---|---|
| 6/24/2025, 6:58:40 PM | - | - | GETARAN_DETEKSI | - | |
| 6/24/2025, 6:58:20 PM | - | - | GETARAN_DETEKSI | - | - |
| 6/24/2025, 6:55:02 PM | 6EEB4743 | - | RFID_DITOLAK | - | |
| 6/24/2025, 6:54:55 PM | 6EEB4743 | - | RFID_DITOLAK | - | - |
| 6/24/2025, 6:54:51 PM | 6EEB4743 | - | RFID_DITOLAK | - | - |
| 6/24/2025, 6:54:32 PM | 6EEB4743 | - | RFID_DITOLAK | - | |
| 6/24/2025, 6:54:28 PM | 6EEB4743 | - | RFID_DITOLAK | - | - |
| 6/24/2025, 6:54:21 PM | 6EEB4743 | - | RFID_DITOLAK | - | - |
| 6/24/2025, 6:50:07 PM | 6EEB4743 | - | RFID_DITOLAK | - | |
| 6/24/2025, 6:50:02 PM | 6EEB4743 | - | RFID_DITOLAK | - | - |

Figure 4 Web Dashboard Interface for Access Log History

The system includes a web-based dashboard built with Laravel, which allows users or administrators to monitor access activity in real-time. As shown in Figure 4, the dashboard displays detailed access logs, including timestamp, UID, authentication method (RFID or fingerprint), event status (e.g., access granted, denied, or vibration detected), user identification, and captured photo evidence. This structured visualization enables traceability of successful and failed access attempts, supporting a more robust and transparent security system.

## 3. RESULTS AND DISCUSSION

### 3.1 Component Testing

Initial testing was carried out on each component individually to ensure that all sensors and modules functioned correctly before being integrated into the whole system. This testing involved assembling the hardware and observing the output from each device, including the RFID sensor, fingerprint sensor, vibration sensor, relay module, ESP32-CAM, and ESP32 Doit V1.

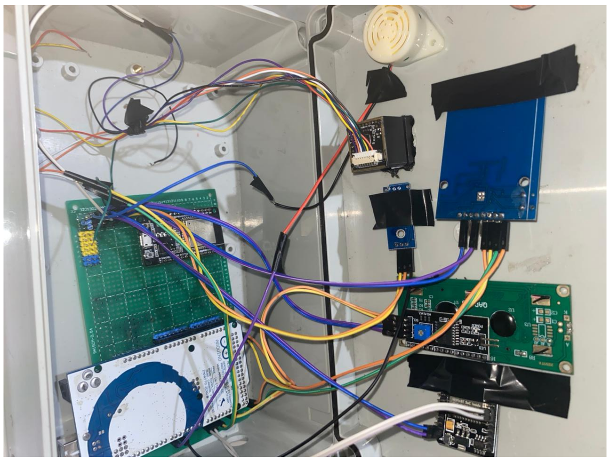


Figure 5 Internal Circuit of the Smart Door Lock System

This figure shows the internal configuration of the system, including the Arduino Mega, ESP32 Doit V1, ESP32-CAM, RFID module, fingerprint sensor, and vibration sensor mounted on a protoboard. The wiring follows a modular and testable design.

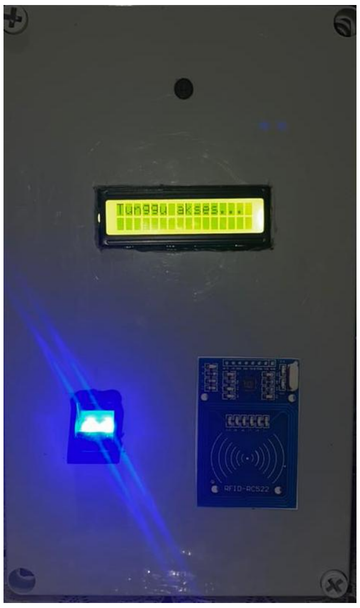


Figure 6 Outer Appearance of the Smart Door Lock Device

This figure displays the external appearance of the assembled system. The fingerprint sensor and RFID reader are embedded on the front panel, along with a 16x2 I2C LCD to display system messages. The device is housed in a protective enclosure for standalone operation.

To support the testing process, the following table lists the main components of the system along with their respective functions and test results:

TABEL I . INDIVIDUAL COMPONENT TESTING RESULTS

| No. | Component | Function | Test Status | Remarks |
|---|---|---|---|---|
| 1. | Arduino Mega | Main controller of system logic | Functional | Stable serial communication |
| 2. | ESP32 Doit V1 | Sends data to Laravel via WiF | Functional | Successfully posted to API |
| 3. | ESP32-CAM | Captures and sends images | Functional | "SNAPSHOT" trigger successful |
| 4. | RFID Sensor RC522 | Card-based authentication | Functional | UID detected accurately |
| 5. | Fingerprint Sensor | Biometric authentication | Functional | Fingerprint matched successfully |
| 6. | Vibration Sensor | Detects door vibrations or tampering | Functional | Activates when shaken |
| 7. | 1-Channel Relay | Door lock actuator (ON/OFF control) | Functional | Successfully activates the solenoid |
| 8. | Solenoid Lock | Electronic door locking mechanism | Functional | Unlocks when the relay is activated |

The results in Table 1 show that all components operated as intended. The Arduino Mega established stable communication with both ESP32 modules. ESP32 Doit V1 successfully transmitted sensor data to the Laravel server, while ESP32-CAM responded correctly to snapshot commands and uploaded images.

Authentication modules (RFID and fingerprint) performed accurately, and the vibration sensor reliably detected physical disturbances. The relay and solenoid lock functioned properly, completing the access control mechanism. Overall, all components are verified to be functional and ready for system integration.

## 3.2 System Functionality Testing

System-level testing was conducted to evaluate the performance of the integrated smart door lock in several real-world scenarios. This includes testing the full flow of authentication using RFID and fingerprint sensors, image capture using ESP32-CAM, and alert triggers from the vibration sensor. Each scenario was executed multiple times to ensure consistency. The system was expected to either unlock the door or deny access while also capturing and transmitting visual evidence to the web server when appropriate.

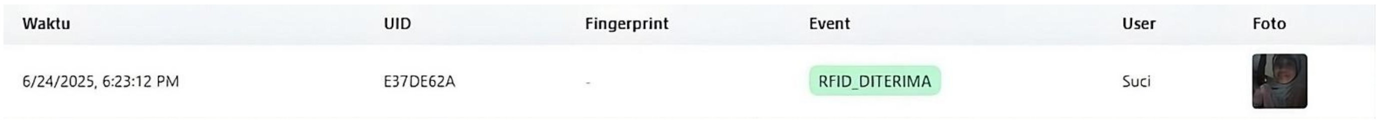
These tests validated the system's ability to respond correctly under varying inputs and conditions. The results demonstrate that the system can reliably support secure access control with real-time logging and visual documentation. All events were recorded correctly on the Laravel web dashboard, including the timestamp, type of sensor triggered, detected UID or fingerprint ID, user information (if any), and the corresponding visual evidence (photo).

TABEL 2 . FUNCTIONAL TESTING SCENARIOS AND RESULTS

| No. | Scenario | Active Components | System Output | Result |
|---|---|---|---|---|
| 1. | Valid RFID authentication | RFID, Relay, ESP32-CAM | Door unlocked, photo captured | Successfully |
| 2. | Invalid RFID | RFID | Access denied, photo will be captured if 3 time fail | Successfully |
| 3. | Valid fingerprint authentication | Fingerprint, Relay, ESP32-CAM | Door unlocked, photo captured | Successfully |
| 4. | Invalid fingerprint | Fingerprint | Access denied, photo will be captured if 3 time fail | Successfully |
| 5. | Door tampering (vibration detected) | Vibration sensor, ESP32-CAM | Photo captured, alert sent to server | Successfully |

The system responded correctly in all tested scenarios. Successful authentications triggered the door to unlock and the ESP32-CAM to transmit an image. Failed authentications initiated image capture after three consecutive rejections. When vibration was detected, the ESP32-CAM immediately captured and uploaded a photo, confirming that the intrusion detection mechanism was functioning as intended.

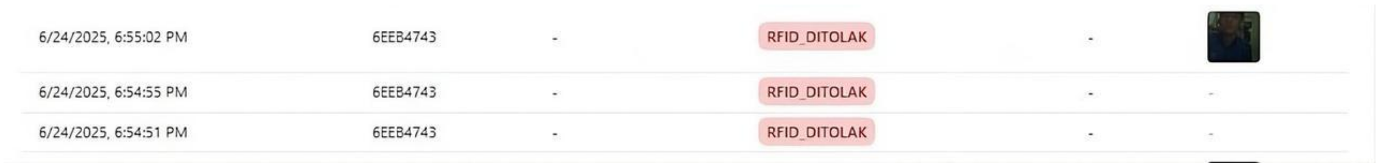
These figures illustrate how each event, whether successful or failed, was recorded and visually documented on the Laravel-based web interface.



| Waktu | UID | Fingerprint | Event | User | Foto |
|---|---|---|---|---|---|
| 6/24/2025, 6:23:12 PM | E37DE62A | - | RFID_DITERIMA | Suci | |

Figure 7 Log of Successful RFID Authentication

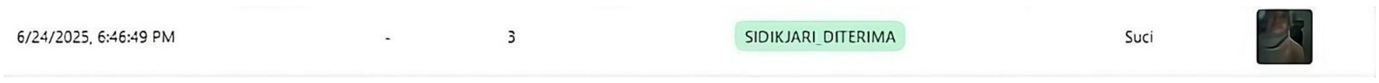
Presents an example of a successful RFID authentication, where the system correctly logs the UID, marks the event as "RFID_DITERIMA" (RFID Accepted), and uploads the user's photo to the server.



| | | | | | |
|---|---|---|---|---|---|
| 6/24/2025, 6:55:02 PM | 6EEB4743 | - | RFID_DITOLAK | - | |
| 6/24/2025, 6:54:55 PM | 6EEB4743 | - | RFID_DITOLAK | - | - |
| 6/24/2025, 6:54:51 PM | 6EEB4743 | - | RFID_DITOLAK | - | - |

Figure 8 Log of Rejected Fingerprint Authentication

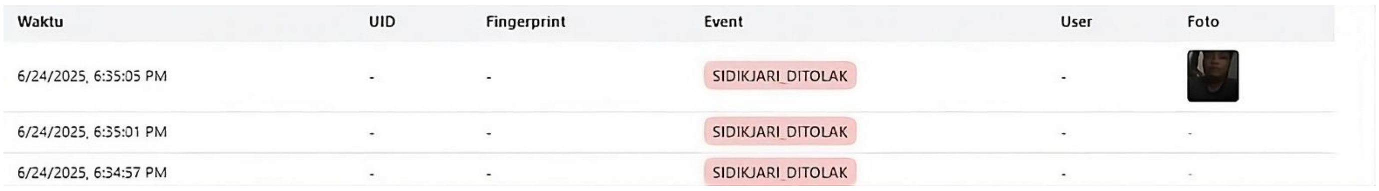
Figure 8 shows an example of unauthorized RFID access attempts. The UID was not recognized, triggering the event "RFID_DITOLAK" (RFID Rejected), while the system still captured and stored a visual log of the individual involved.



| | | | | | |
|---|---|---|---|---|---|
| 6/24/2025, 6:46:49 PM | - | 3 | SIDIKJARI_DITERIMA | Suci | |

Figure 9 Log of Successful Fingerprint Authentication

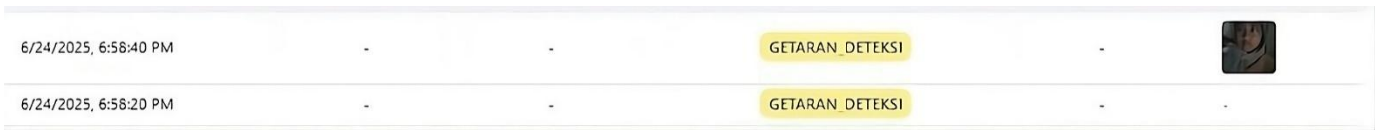
In Figure 9 the system displays a successful fingerprint authentication, logs the fingerprint ID, marks the event as "SIDIKJARI_DITERIMA," and saves the corresponding user photo.



| Waktu | UID | Fingerprint | Event | User | Foto |
|---|---|---|---|---|---|
| 6/24/2025, 6:35:05 PM | - | - | SIDIKJARI_DITOLAK | - | |
| 6/24/2025, 6:35:01 PM | - | - | SIDIKJARI_DITOLAK | - | - |
| 6/24/2025, 6:34:57 PM | - | - | SIDIKJARI_DITOLAK | - | - |

Figure 10 Log of Rejected RFID Attempt

Figure 10 illustrates a case in which fingerprint authentication was rejected. Despite the failure, the system still captured a photo and registered the event as "SIDIKJARI_DITOLAK" (Fingerprint Rejected), ensuring all unauthorized attempts were visually documented.



| | | | | | |
|---|---|---|---|---|---|
| 6/24/2025, 6:58:40 PM | - | - | GETARAN_DETEKSI | - | |
| 6/24/2025, 6:58:20 PM | - | - | GETARAN_DETEKSI | - | - |

Figure 11 Log of Vibration Detection Event

Figure 10 demonstrates the system's response to physical disturbance detected by the vibration sensor. The event is marked as "GETARAN_DETEKSI" (Vibration Detected), and a photo was automatically captured and stored.

## 3.3 Response Time Testing

Response time testing was conducted to evaluate the system’s performance in terms of speed and responsiveness. This test measured the delay between the occurrence of a trigger event, such as successful authentication or vibration detection, and the moment when the ESP32-CAM completes the process of capturing and uploading the image to the Laravel-based web server. The observed delay encompasses the duration required for serial communication between the Arduino Mega and the ESP32-CAM, the time taken to process and capture the image, the HTTP POST request and image upload to the server, and the reception of a confirmation response from the server.

The response time Tdelay is calculated by subtracting the trigger time from the image upload time, as shown in Equation (1):

$$T_{delay} = T_{upload} - T_{trigger} \quad (1)$$

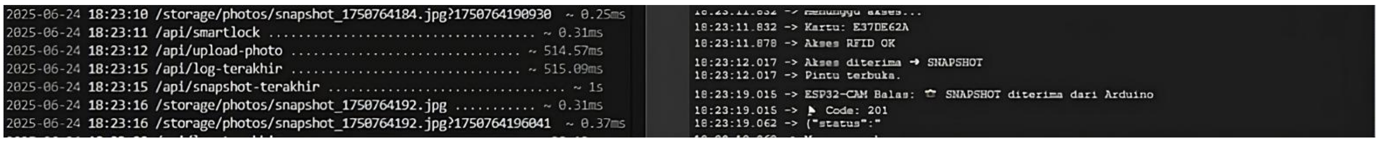
Data for this measurement was obtained through timestamps recorded on the ESP32-CAM serial monitor and verified against the backend server logs. Table 3 summarizes the average delay values obtained from five distinct events.

TABEL 3 . DELAY MEASUREMENT FROM TRIGGER TO PHOTO UPLOAD

| No. | Event Type | Trigger Time | Upload Time | Delay (s) |
|---|---|---|---|---|
| 1. | RFID Authentication | 18:23:12 | 18:23:16 | 4.0 |
| 2. | Fingerprint Rejected | 18:35:04 | 18:35:09 | 5.0 |
| 3. | Fingerprint Accepted | 18:46:49 | 18:46:54 | 5.0 |
| 4. | RFID Rejected | 18:55:03 | 18:55:08 | 5.0 |
| 5. | Vibration Detection | 18:58:40 | 18:58:45 | 5.0 |

Based on the results, the system consistently completed the image capture and upload process within 4 to 5 seconds from the time the trigger command was issued. This delay is acceptable for security monitoring purposes, especially in scenarios where real-time feedback is not strictly required, but timely documentation is essential.

Figure 12 to Figure 16 show excerpts from the backend logs used to extract timing data for each event scenario.

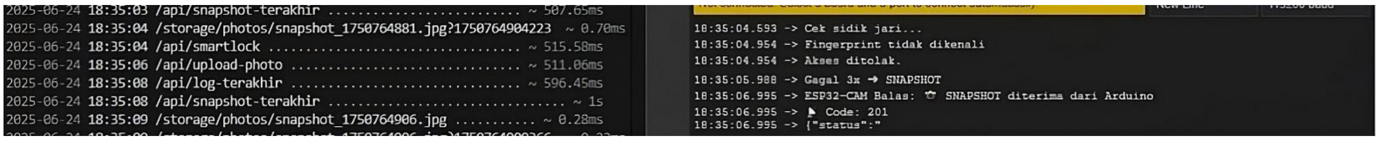


```
2025-06-24 18:23:10 /storage/photos/snapshot_1750764184.jpg?1750764190930  ~ 0.25ms
2025-06-24 18:23:11 /api/smartlock ................................. ~ 0.31ms
2025-06-24 18:23:12 /api/upload-photo .............................. ~ 514.57ms
2025-06-24 18:23:15 /api/log-terakhir .............................. ~ 515.09ms
2025-06-24 18:23:15 /api/snapshot-terakhir ............................ ~ 1s
2025-06-24 18:23:16 /storage/photos/snapshot_1750764192.jpg .......... ~ 0.31ms
2025-06-24 18:23:16 /storage/photos/snapshot_1750764192.jpg?1750764196041  ~ 0.37ms
```

```
18:23:11.832 -> Kartu: E37DE62A
18:23:11.878 -> Akses RFID OK
18:23:12.017 -> Akses diterima ➜ SNAPSHOT
18:23:12.017 -> Pintu terbuka.
18:23:19.015 -> ESP32-CAM Balas: SNAPSHOT diterima dari Arduino
18:23:19.015 -> Code: 201
18:23:19.062 -> ("status":"
```

Figure 12 RFID Authentication RFID Accepted Event Timing Log

This log shows a successful RFID authentication. At timestamp 18:23:12, the door was unlocked, and a SNAPSHOT command was sent. The image was captured and uploaded by ESP32-CAM at 18:23:16, resulting in a delay of 4 seconds.

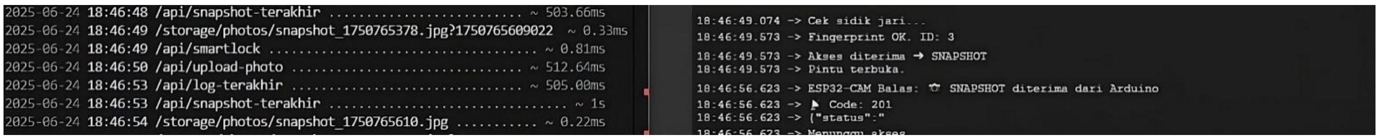


```
2025-06-24 18:35:03 /api/snapshot-terakhir ......................... ~ 507.65ms
2025-06-24 18:35:04 /storage/photos/snapshot_1750764881.jpg?1750764904223  ~ 0.70ms
2025-06-24 18:35:04 /api/smartlock ................................ ~ 515.58ms
2025-06-24 18:35:06 /api/upload-photo ............................. ~ 511.06ms
2025-06-24 18:35:08 /api/log-terakhir ............................. ~ 596.45ms
2025-06-24 18:35:08 /api/snapshot-terakhir ............................. ~ 1s
2025-06-24 18:35:09 /storage/photos/snapshot_1750764906.jpg .......... ~ 0.28ms
```

```
18:35:04.593 -> Cek sidik jari...
18:35:04.954 -> Fingerprint tidak dikenali
18:35:04.954 -> Akses ditolak.
18:35:05.988 -> Gagal 3x ➜ SNAPSHOT
18:35:06.995 -> ESP32-CAM Balas: 📸 SNAPSHOT diterima dari Arduino
18:35:06.995 -> 📡 Code: 201
18:35:06.995 -> {"status":"
```

Figure 13 Fingerprint Rejected Event Timing Log

This example demonstrates a failed fingerprint attempt. After three unsuccessful tries (18:35:04), a snapshot was triggered and the image was uploaded by 18:35:09, with a consistent 5-second delay.

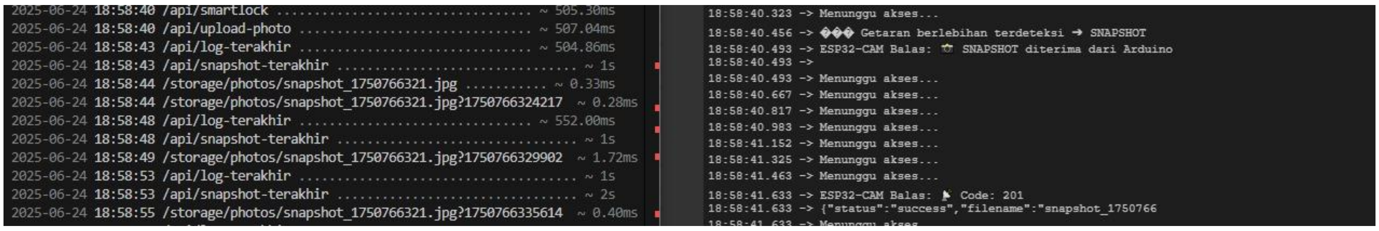


```
2025-06-24 18:46:48 /api/snapshot-terakhir ......................... ~ 503.66ms
2025-06-24 18:46:49 /storage/photos/snapshot_1750765378.jpg?1750765609022  ~ 0.33ms
2025-06-24 18:46:49 /api/smartlock .................................. ~ 0.81ms
2025-06-24 18:46:50 /api/upload-photo ............................. ~ 512.64ms
2025-06-24 18:46:53 /api/log-terakhir ............................. ~ 505.00ms
2025-06-24 18:46:53 /api/snapshot-terakhir ............................. ~ 1s
2025-06-24 18:46:54 /storage/photos/snapshot_1750765610.jpg .......... ~ 0.22ms
```

```
18:46:49.074 -> Cek sidik jari...
18:46:49.573 -> Fingerprint OK. ID: 3
18:46:49.573 -> Akses diterima ➜ SNAPSHOT
18:46:49.573 -> Pintu terbuka.
18:46:56.623 -> ESP32-CAM Balas: 📸 SNAPSHOT diterima dari Arduino
18:46:56.623 -> 📡 Code: 201
18:46:56.623 -> {"status":"
```

Figure 14 Fingerprint Rejected Event Timing Log

This example demonstrates a failed fingerprint attempt. After three unsuccessful tries (18:35:04), a snapshot was triggered and the image was uploaded by 18:35:09, with a consistent 5-second delay.

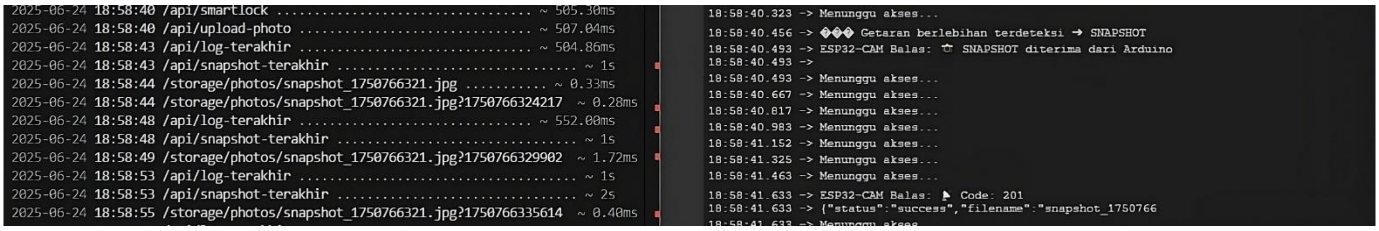


```
2025-06-24 18:58:40 /api/smartlock ................................ ~ 505.30ms
2025-06-24 18:58:40 /api/upload-photo ............................. ~ 507.04ms
2025-06-24 18:58:43 /api/log-terakhir ............................. ~ 504.86ms
2025-06-24 18:58:43 /api/snapshot-terakhir ............................. ~ 1s
2025-06-24 18:58:44 /storage/photos/snapshot_1750766321.jpg .......... ~ 0.33ms
2025-06-24 18:58:44 /storage/photos/snapshot_1750766321.jpg?1750766324217  ~ 0.28ms
2025-06-24 18:58:48 /api/log-terakhir ............................. ~ 552.00ms
2025-06-24 18:58:48 /api/snapshot-terakhir ............................. ~ 1s
2025-06-24 18:58:49 /storage/photos/snapshot_1750766321.jpg?1750766329902  ~ 1.72ms
2025-06-24 18:58:53 /api/log-terakhir .................................. ~ 1s
2025-06-24 18:58:53 /api/snapshot-terakhir ............................. ~ 2s
2025-06-24 18:58:55 /storage/photos/snapshot_1750766321.jpg?1750766335614  ~ 0.40ms
```

```
18:58:40.323 -> Menunggu akses...
18:58:40.456 -> ��� Getaran berlebihan terdeteksi ➜ SNAPSHOT
18:58:40.493 -> ESP32-CAM Balas: 📸 SNAPSHOT diterima dari Arduino
18:58:40.493 ->
18:58:40.493 -> Menunggu akses...
18:58:40.667 -> Menunggu akses...
18:58:40.817 -> Menunggu akses...
18:58:40.983 -> Menunggu akses...
18:58:41.152 -> Menunggu akses...
18:58:41.325 -> Menunggu akses...
18:58:41.463 -> Menunggu akses...
18:58:41.633 -> ESP32-CAM Balas: 📡 Code: 201
18:58:41.633 -> {"status":"success","filename":"snapshot_1750766
```

Figure 3.15 Fingerprint Rejected Event Timing Log

This example demonstrates a failed fingerprint attempt. After three unsuccessful tries (18:35:04), a snapshot was triggered and the image was uploaded by 18:35:09, with a consistent 5-second delay.



```
2025-06-24 18:58:40 /api/smartlock ................................ ~ 505.30ms
2025-06-24 18:58:40 /api/upload-photo ............................. ~ 507.04ms
2025-06-24 18:58:43 /api/log-terakhir ............................. ~ 504.86ms
2025-06-24 18:58:43 /api/snapshot-terakhir ............................. ~ 1s
2025-06-24 18:58:44 /storage/photos/snapshot_1750766321.jpg .......... ~ 0.33ms
2025-06-24 18:58:44 /storage/photos/snapshot_1750766321.jpg?1750766324217  ~ 0.28ms
2025-06-24 18:58:48 /api/log-terakhir ............................. ~ 552.00ms
2025-06-24 18:58:48 /api/snapshot-terakhir ............................. ~ 1s
2025-06-24 18:58:49 /storage/photos/snapshot_1750766321.jpg?1750766329902  ~ 1.72ms
2025-06-24 18:58:53 /api/log-terakhir .................................. ~ 1s
2025-06-24 18:58:53 /api/snapshot-terakhir ............................. ~ 2s
2025-06-24 18:58:55 /storage/photos/snapshot_1750766321.jpg?1750766335614  ~ 0.40ms
```

```
18:58:40.323 -> Menunggu akses...
18:58:40.456 -> ��� Getaran berlebihan terdeteksi ➜ SNAPSHOT
18:58:40.493 -> ESP32-CAM Balas: 📸 SNAPSHOT diterima dari Arduino
18:58:40.493 ->
18:58:40.493 -> Menunggu akses...
18:58:40.667 -> Menunggu akses...
18:58:40.817 -> Menunggu akses...
18:58:40.983 -> Menunggu akses...
18:58:41.152 -> Menunggu akses...
18:58:41.325 -> Menunggu akses...
18:58:41.463 -> Menunggu akses...
18:58:41.633 -> ESP32-CAM Balas: 📡 Code: 201
18:58:41.633 -> {"status":"success","filename":"snapshot_1750766
```

Figure 3.16 Vibration Detection Event Timing Log

In this case, a vibration was detected at 18:58:40. The ESP32-CAM reacted immediately to capture and upload the image, which was completed at 18:58:45, again reflecting a 5-second response time.

## 3.4 System Discussion

Testing confirmed that the intelligent door lock system performed reliably across both component-level and full-system scenarios. All modules including RFID, fingerprint, vibration sensor, ESP32-CAM, and ESP32 Doit V1 operated as intended, accurately triggering door control and image capture under various authentication outcomes.

Valid access led to immediate unlocking and photo upload, while failed attempts prompted image capture after three retries. The vibration sensor also responded effectively, triggering the camera with minimal delay. The average response time of 4-5 seconds was acceptable for its monitoring purpose.

The system outperforms several related works that lacked image documentation or web connectivity by integrating ESP32-CAM for automatic visual logging and a Laravel-based dashboard for real-time monitoringCombining three authentication layers further enhances its security profile.

Despite its strengths, the system depends on Wi-Fi stability for image uploads and lacks user management on the dashboard. These aspects can be improved in future iterations. Overall, the system meets its objective as a practical, low-cost smart security solution with robust multi-factor access control.

## 4. Conclusion and Suggestions

This study successfully developed a smart door lock system that integrates RFID, fingerprint, and vibration sensors with an ESP32-CAM for automated visual evidence, all connected to a Laravel-based web server. The system performed reliably in multiple test scenarios, consistently responding to valid and invalid access attempts by unlocking the door or denying entry while capturing and uploading images in real time.

Compared to previous research, the proposed system offers enhanced security through multi-factor authentication and real-time logging, including photo documentation. Although the average image upload delay ranged from 4–5 seconds, it remained acceptable for monitoring use cases. Future improvements could include user authentication on the web dashboard, local backup mechanisms, and extended field deployment to evaluate system robustness.

## REFERENCES

[1] W. Gamma *et al.*, “IOT SMART DOOR LOCK SYSTEM MENGGUNAKAN DOUBLE SENSOR,” vol. 8, no. 1, pp. 74–82, 2025.

[2] R. W. Tambunan, A. A. Ar-Rafif, and M. Galina, “Multi-Security System Based on RFID Fingerprint and Keypad to Access the Door,” *Elkha*, vol. 14, no. 2, p. 125, 2022, doi: 10.26418/elkha.v14i2.57735.

[3] D. Hermawan, J. Jufrizel, A. Ullah, and A. Faizal, “Rancang Bangun Keamanan Kotak Amal dengan Akses Fingerprint Menggunakan ESP32-Cam dan Telegram Berbasis IOT,” *J. Media Inform. Budidarma*, vol. 7, no. 3, p. 1013, 2023, doi: 10.30865/mib.v7i3.6252.

[4] A. Ziad and E. Darnila, “Development and Implementation of an ESP32 Microcontroller and Monitoring System for Smart Door Lock Using RFID Sensor for E-KTP ID and Fingerprint Based on the Internet of Things,” vol. 00023, pp. 1–11, 2024.

[5] R. R. Hiwrale, R. P. Bhole, P. G. Fandale, R. V Dobe, and S. B. Patil, “RESEARCH ON CLOUD-ENABLED BIOMETRIC DOOR ACCESS SYSTEM,” no. 03, pp. 11590–11597, 2025.

[6] O. Thombre, P. Thakur, S. Tikone, and P. S. Chavan, “IoT-Based Smart Logistics and Transportation Management System,” pp. 48–54, 2025.

[7] D. Prathapagiri and K. Eethamakula, “Wi-Fi Door Lock System Using ESP32 CAM Based on IoT,” *Dep. ECM, J. B. Inst. Eng. Technol.*, vol. XIII, no. VII, pp. 2001–2003, 2021.

[8] E. S. P. Cam and A. Winarno, “Home Door Security System with Face Recognition Using,” vol. 06, no. 02, pp. 9–14, 2024.

[9] K. Ananta, K. Permana, and I. N. Piarsa, “IoT-Based Smart Door Lock System with Fingerprint and Keypad Access,” vol. 6, no. 3, pp. 2086–2098, 2024, doi: 10.51519/journalisi.v6i3.844.

[10] S. Rajarajeswari and N. Hema, “Smart Door Locking System,” vol. 14, no. 3, pp. 101–126, 2022, doi: 10.1007/978-981-19-2184-1_5.

[11] K. G. Prakoso and B. Sisephaputra, “PERANCANGAN SISTEM MONITORING RUANGAN KULIAH BERBASIS NFC DAN IOT (Studi Kasus: Universitas Negeri Surabaya),” *J. Emerg. Inf. Syst. Bus. Intell.*, vol. 5, no. 2, pp. 189–197, 2024.

[12] A. Anggit Pratama and R. R. Santika, “Penerapan Presensi Menggunakan Rfid Dan Esp32 Cam Berbasis Website Pada Pt. Yono Express Services,” *Semin. Nas. Mhs. Fak. Teknol. Inf. Jakarta-Indonesia*, no. September, pp. 1001–1010, 2022.

[13] N. Raju, A. Navya, N. Koteswaramma, B. Mounika, and T. Rajeshwari, “IoT based Door Access Control System using ESP32cam,” *Int. J. Eng. Invent.*, vol. 11, no. 12, pp. 9–15, 2022, [Online]. Available: www.ijeijournal.com.

[14] M. G. PRAKASA and N. SYAFITRI, “Sistem Presensi dengan Fitur RFID dan Capture Citra menggunakan NodeMCU dan ESP32-Cam,” *ELKOMIKA J. Tek. Energi Elektr. Tek. Telekomun. Tek. Elektron.*, vol. 11, no. 2, p. 310, 2023, doi: 10.26760/elkomika.v11i2.310.

[15] D. A. Pratama *et al.*, “Sistem Pengaman Pintu Elektronik Otomatis Dengan Memanfaatkan E-KTP Sebagai RFID Card Ruang Dosen Teknik Elektronika Politeknik Negeri Sriwijaya,” *Jurnal TIPS: Jurnal Teknologi Informasi Dan Komputer Politeknik Sekayu*, vol. 9, no. 2, pp. 50–53, 2018.