IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, October 2026
Impact Factor: 8.2

Hardware-Assisted Security Using IoT Devices (Using RFID-Based Access Control)

Smt. Vaishali M. Khanapure, Mr. Omkar Chopade, Mr. Shreetej Rokade, Mr. Pranjal Shinde, Mr. Siddharth Shinde
Department of Information Technology
Government Polytechnic, Pune, Maharashtra, India

Abstract: *This paper presents the design and functional evaluation of a low-cost, hardware-assisted access-control prototype for Internet of Things (IoT) devices. The implementation combines an RC522 radio-frequency identification (RFID) reader, an ESP8266-based NodeMCU controller, RFID cards or tags, relay-controlled electronic locking, audible and visual alerts, local EEPROM storage, and a browser-based management interface. When a card is presented, the controller reads its identifier and compares it with stored records and the record's access-permission flag. An enabled registered card activates the relay for approximately two seconds, while an unknown card or a registered card with access disabled is denied. Unauthorized use triggers three buzzer beeps and three red-LED blinks; after three unauthorized attempts, the firmware suspends RFID processing for a 15-second lockout. The NodeMCU also hosts a local Wi-Fi access point and HTTP interface for card registration, deletion, permission changes, tap-count reset, and status display. The supplied capstone report documents 18 test cases with a Pass status across card detection, authorization, actuation, alerting, lockout, EEPROM persistence, web management, and initialization. The project report does not provide measured latency distributions, large-scale reliability data, or penetration-test results; consequently, this paper distinguishes functional verification from claims of security effectiveness. The design is suitable as an educational prototype and a basis for further work, while production deployment would require stronger credential protection, authenticated administration, encrypted communication, and more comprehensive evaluation.*

Keywords: Internet of Things, RFID, access control, NodeMCU ESP8266, embedded security, local web interface, electronic lock

I. INTRODUCTION

Connected devices increasingly interact with physical spaces and equipment. In homes, laboratories, offices, and educational facilities, a digital controller may operate a lock, instrument, cabinet, or other restricted resource. Such systems need a clear way to determine which users may perform an action and how the system should respond when a credential is not accepted. Traditional keys are tangible and can be lost or copied; shared passwords can be disclosed or reused. Electronic identification offers an alternative for automating access decisions, but it must be connected to a dependable control path and an understandable administration process.

Radio-frequency identification provides contactless identification through a reader and a tag or card. In the system studied here, an RC522 reader obtains a card identifier and passes it to a NodeMCU ESP8266. The controller compares the identifier with locally stored card records, checks whether access is enabled, and then operates a relay and electronic lock or produces a denial alert. The device also hosts a browser-accessible interface so an operator can register cards and change permissions without rebuilding the firmware for each enrolment change. RFID's practical advantages and security limits are well established in the literature [1]-[5].

The motivating use cases are restricted-access cabinets, teaching laboratories, small offices, and equipment enclosures where the operator needs straightforward local enrolment and visible feedback. For such settings, a low-cost prototype can demonstrate how an embedded controller converts a credential into a physical control signal. The relevant system boundary includes the tag, reader, firmware, locally stored card table, relay driver, lock, indicators, power supply, and administrator device. A failure at any point can affect the intended outcome: a reader fault may prevent valid entry, a stuck relay may undermine the lock interval, and an exposed management endpoint may allow stored permissions to be altered. The design is therefore considered as a complete socio-technical access path, not merely as an RFID-reading exercise.

A central engineering aim is to bind the digital decision to a physical result: an accepted card must produce a brief unlocking action, while an unaccepted credential must not energize the relay. The project extends this basic decision with local status information, a per-card tap counter, a persistent card list, and a temporary lockout after repeated invalid attempts. This combination makes the prototype more than a standalone card reader, while keeping the control path within a small embedded device and avoiding a mandatory cloud service.

The system is designed for local operation. At startup, the NodeMCU initializes the RFID reader and output pins, retrieves stored card records from EEPROM, creates a Wi-Fi access point named RFID_Access, and starts an HTTP server accessible from a connected device. The supplied implementation uses the local address 192.168.4.1 for the management page. Status is requested periodically by the browser, allowing the operator to observe the last tapped card, the registered-card list, tap counts, access flags, and relay state. Local operation simplifies deployment but also places much of the management trust boundary on the device's Wi-Fi configuration and web handlers.

A. Problem Statement

The problem addressed is the absence of an integrated, inexpensive mechanism that can identify a card, apply a locally changeable authorization rule, operate a physical lock, alert on rejection, and let an operator manage credentials. The prototype must also respond to repeated unauthorized attempts without requiring continuous human observation. A useful design therefore needs a connected sequence of sensing, decision-making, actuation, state retention, and administration rather than an RFID reader alone.

B. Objectives

- Read the identifier of an RFID card or tag using the RC522 reader.
- Grant access only when the card is registered and its access flag is enabled.
- Operate a relay-controlled electronic lock for a programmed interval of approximately two seconds.
- Give audible and visual feedback when access is denied and temporarily inhibit reads after three unauthorized attempts.
- Store card identifiers, assigned names, access permissions, and tap counts in local non-volatile memory.
- Provide a local web interface for enrolment, deletion, permission changes, resets, and status monitoring.

C. Contributions and Scope

This paper contributes a reproducible description of the implemented prototype, a mapping between its hardware and software functions, a record of the functional test outcomes reported in the capstone document, and a practical security analysis of the design. It does not claim a new RFID cryptographic protocol, a field deployment, or measured superiority over commercial products. The intended scope is a low-cost educational access-control demonstrator that illustrates hardware-software integration and provides a clear starting point for stronger authentication and operational testing. The rest of the paper reviews relevant work, specifies the system, describes its architecture and implementation, presents the documented tests, and discusses limitations and future improvements.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, October 2026
Impact Factor: 8.2

D. Assumptions and Evaluation Boundary

The paper assumes that the reader is mounted close enough for the selected card or tag to be detected, the controller and reader share a suitable power reference, and the relay module is electrically compatible with the controller and lock supply. These are integration assumptions, not outcomes of a separate electrical certification exercise. The management interface is assumed to be used on a trusted local network. No remote service, centralized identity provider, synchronized clock, or external access log is present in the documented design. These boundaries determine what the project can demonstrate and what must be added before a real deployment.

II. RELATED WORK AND RESEARCH GAP

RFID has long been used for identification and automated data capture. Want surveys the underlying technology and its application context [1], while Juels and Weis et al. review security and privacy risks associated with low-cost RFID systems [2], [3]. These studies are relevant because a tag identifier is useful for lookup, but identifier-based recognition alone does not guarantee that a credential cannot be copied, replayed, or misused. A practical access system must therefore distinguish the operational convenience of RFID from the stronger guarantees expected of cryptographic authentication.

IoT security literature similarly treats protection as a system-level concern. Al-Fuqaha et al. describe the technologies and communication mechanisms that support IoT deployments [7], while Granjal et al., Yang et al., Sicari et al., and Roman et al. discuss security, privacy, trust, and protocol challenges across connected devices [8]-[11]. In an embedded access-control setting, these concerns include who can reach the management interface, how credentials are stored, and whether a network-level compromise can be converted into a physical action. A compact prototype can demonstrate core control logic but should not be interpreted as a complete production security architecture.

Research specifically on RFID-based IoT access control includes schemes that add stronger security mechanisms, such as dedicated cryptographic components, to open-source hardware designs [4]. Related work also studies RFID attendance and access management for workforce settings [6]. Abdulghani et al. organize security and privacy guidance for RFID-based IoT applications and highlight the need to consider threats, mitigations, and design principles together [5]. Compared with these security-focused approaches, the present implementation emphasizes low-cost hardware, offline-local operation, mutable access permissions, audible and visual denial alerts, and a browser-based credential management workflow. Its design does not document cryptographic challenge-response authentication or secure remote administration.

TABLE I. QUALITATIVE COMPARISON OF ACCESS-CONTROL APPROACHES

Approach	Strength	Limitation Relevant to This Project
Physical key or manual check	Simple and familiar operation	Limited electronic logging and flexible revocation
Basic RFID reader and relay	Fast, contactless identification	Often lacks credential administration and repeated-attempt response
RFID access control with IoT interface	Remote or local software management	Network and web interfaces add security responsibilities
Cryptographic RFID/IoT design	Can strengthen authenticity and integrity	May require more capable tags, secure elements, or added complexity
Prototype in this paper	Local enrolment, access toggling, alerting, lockout, tap counts	UID-based decision and local web management need hardening

A. Research Gap

The project addresses a practical integration gap: demonstrations often show the reader-to-lock path, but a usable installation also needs a way to enrol, disable, remove, and review credentials without rewriting the firmware. It brings these functions together with a repeated-failure lockout. Its research value is therefore implementation-oriented. The key remaining gap is not basic functionality, which is covered by the documented tests, but evidence about resistance to cloning, web-interface abuse, power interruption, longer-term reliability, and operation at multiple access points.

B. Positioning of the Present Work

The contribution is not the assertion that RFID, relays, or embedded web servers are new. It is the documented combination of those components into a maintainable prototype with explicit authorization states and administration operations. Its intended deployment scale is one controller and one access point. The design is less elaborate than cloud-managed or cryptographically enhanced systems, but it illustrates the basic control model, including how permission changes and stored records influence the physical output. This positioning is important when interpreting the evaluation: the system is a functional reference implementation that should be strengthened incrementally, not a substitute for a certified commercial access controller.

III. SYSTEM REQUIREMENTS AND DESIGN GOALS

The prototype combines an embedded controller with identification, switching, indication, storage, and management components. Its design is deliberately local: the NodeMCU receives RFID events and makes the access decision without depending on an external application server. This reduces infrastructure requirements for a demonstration, but means card management, stored data, and event visibility are tied to the state and availability of a single controller.

TABLE II. FUNCTIONAL REQUIREMENTS AND IMPLEMENTED FEATURES

ID	Requirement	Implementation in the Prototype
FR1	Card identification	Read the RFID UID through the RC522 reader over SPI.
FR2	Authorization decision	Find a stored UID and check its access-enabled flag.
FR3	Physical access	Activate relay and electronic lock for about two seconds.
FR4	Denial indication	Three buzzer beeps and three red-LED blinks for an invalid or disabled card.
FR5	Repeated-attempt control	Enter a 15-second lock state after three unauthorized attempts.
FR6	Credential administration	Register, delete, enable/disable, and reset card records from the local web UI.
FR7	Persistent record storage	Store card details and tap counts in EEPROM.
FR8	System visibility	Display last tapped card, card list, tap count, access flag, and relay status.

A. Hardware and Software Resources

The principal hardware is the NodeMCU ESP8266, RC522 RFID reader, RFID cards or tags, relay module, 12 V solenoid/electronic lock, red LED, buzzer, supporting transistor where required, breadboard, jumper wires, USB connection, and battery-based power components. The software environment is Arduino IDE with ESP8266 board support. The firmware uses the MFRC522 library for reader communication, SPI for the reader bus, ESP8266WiFi and ESP8266WebServer for the local access point and HTTP routes, and EEPROM for persistence [18]-[22].

B. Non-Functional Goals

The design goals are affordability, local autonomy, understandable operation, modular replacement of components, and maintainability. A connected browser should be sufficient to manage the prototype on its local Wi-Fi network; no dedicated cloud service is required for the documented features. These goals are design intentions rather than independently measured non-functional results.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, October 2026
Impact Factor: 8.2

C. Operating Environment and Constraints

Because the prototype is locally hosted, the administrator's browser must be connected to the controller's access point, and the interface is unavailable when that local network or controller is offline. The system also relies on the lock's electrical and mechanical characteristics, which are outside the firmware's ability to verify. The report lists a 12 V solenoid/electronic lock and supporting power components, but does not provide a certified circuit diagram, current profile, thermal evaluation, or battery-runtime measurement. The physical assembly should therefore be treated as a prototype requiring appropriate power and relay-driver checks for each intended lock.

IV. SYSTEM ARCHITECTURE

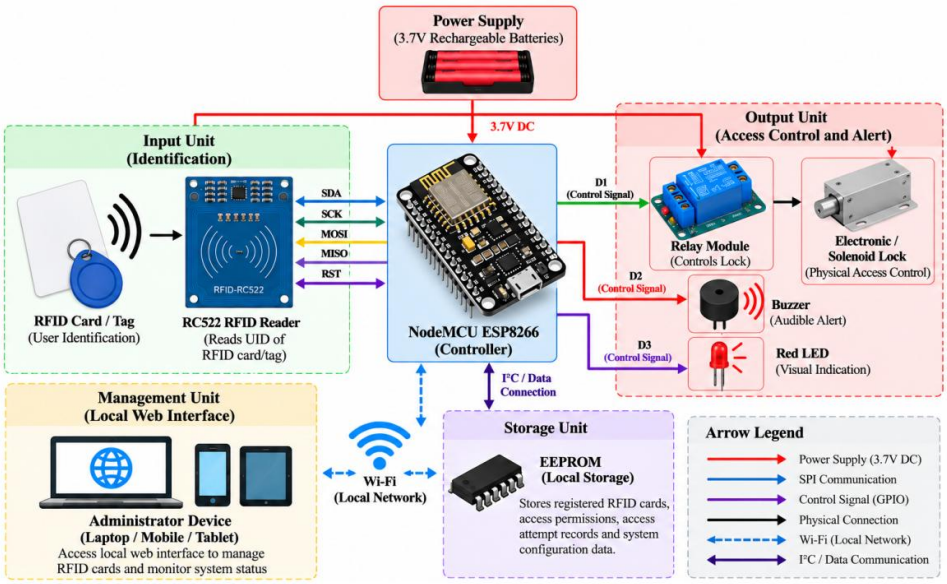

Fig. : Architecture Diagram

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, October 2026
Impact Factor: 8.2

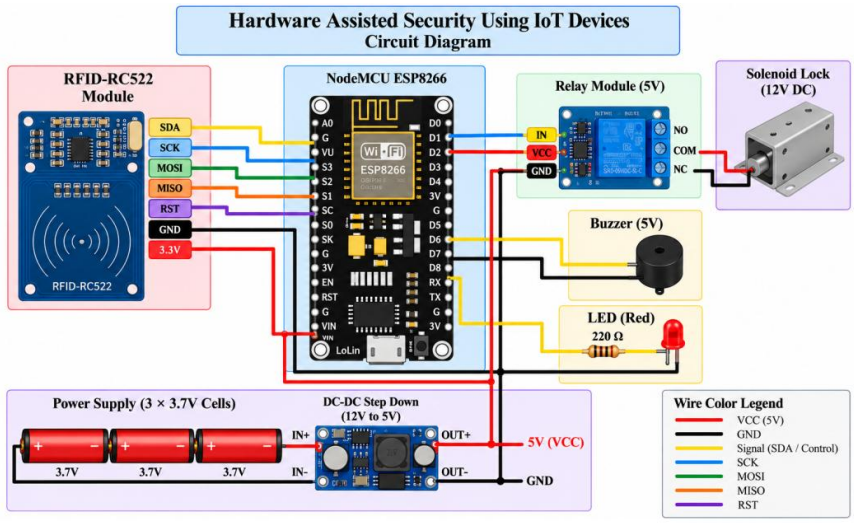

Fig. : Circuit Diagram

The architecture separates the system into five interacting parts: the RFID sensing path, the embedded decision controller, the physical output path, local record storage, and the management interface. The division makes it possible to reason about each responsibility independently while retaining a short end-to-end path from card presentation to the lock response.

D. Identification and Controller Layer

An RFID card or tag is presented to the RC522 reader. The reader communicates with the NodeMCU over SPI and provides the identifier required by the firmware. The ESP8266 acts as the decision point: it searches the locally stored card records, determines whether a matching record exists, and checks the access flag. The record includes a UID, assigned name, permission status, and tap count. A new record created through the interface is initially stored with access disabled, so enrolment and authorization remain separate administrative actions.

E. Actuation, Storage, and Administration Layers

For an enabled registered card, the firmware activates the relay that switches the electronic lock and then turns it off after the programmed interval. The red LED remains off and the buzzer gives a short confirmation beep. An unknown or disabled credential does not activate the relay; it triggers the denial pattern and updates the unauthorized-attempt handling. EEPROM retains the card list across restarts, while the locally hosted web application exposes the functions needed to maintain that list. The architecture does not depend on an external database or cloud link for these operations. This local design creates a simple deployment model: the reader and outputs are wired to the controller, the lock has its switching and power path, and an administrator connects to the controller's Wi-Fi access point to manage records. It also concentrates responsibility in one device. The controller's firmware, configuration, local memory, and administrative routes therefore become critical security boundaries that must be protected in a production version.

F. Operational Dependencies

The architecture has a deliberately short critical path, but it still depends on correct wiring and stable power for both the controller and the lock. The relay module acts as a switching boundary between the low-voltage controller and the lock circuit, so the lock's current must not be drawn directly from a controller GPIO. A suitable driver, supply arrangement, and protection against inductive transients are needed in the physical installation. The capstone describes the required relay, transistor where necessary, and battery/USB components; electrical safety and the lock's default state must be assessed in the context of the actual hardware used.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, October 2026
Impact Factor: 8.2

V. AUTHENTICATION WORKFLOW AND DECISION LOGIC

The event-handling process consists of detection, record lookup, permission checking, output actuation, and state update. These operations are sequenced so a record being present is not itself sufficient to unlock the resource: the access flag must also be enabled. This distinction allows an administrator to retain a card record while temporarily or permanently withdrawing the card's permission.

The diagram's login and access-log steps are conceptual: the report documents no separate administrator authentication or timestamped event log.

A. Access Decision Sequence

- Read the card UID when the RC522 detects a tag and compare it with locally registered records.
- If a matching record is present and access is enabled, activate the relay and release the lock for approximately two seconds; provide one short buzzer beep and increment the card tap count.
- If the UID is unknown or access is disabled, leave the relay inactive, produce three buzzer beeps and three LED blinks, and count the unauthorized attempt.
- After the third unauthorized attempt, suspend RFID processing for 15 seconds and keep the red LED on during the lockout; return to normal operation when the interval expires.

The relay interval and the lockout interval are firmware-configured durations. They are not reported as measured timing distributions. The firmware uses a time-based relay-off mechanism based on millis(), which avoids waiting synchronously for the complete relay interval. The alert behaviour is also handled as a programmed sequence. This is important because a blocking implementation could make the local server less responsive while an alert is being generated.

B. State Transitions and Edge Cases

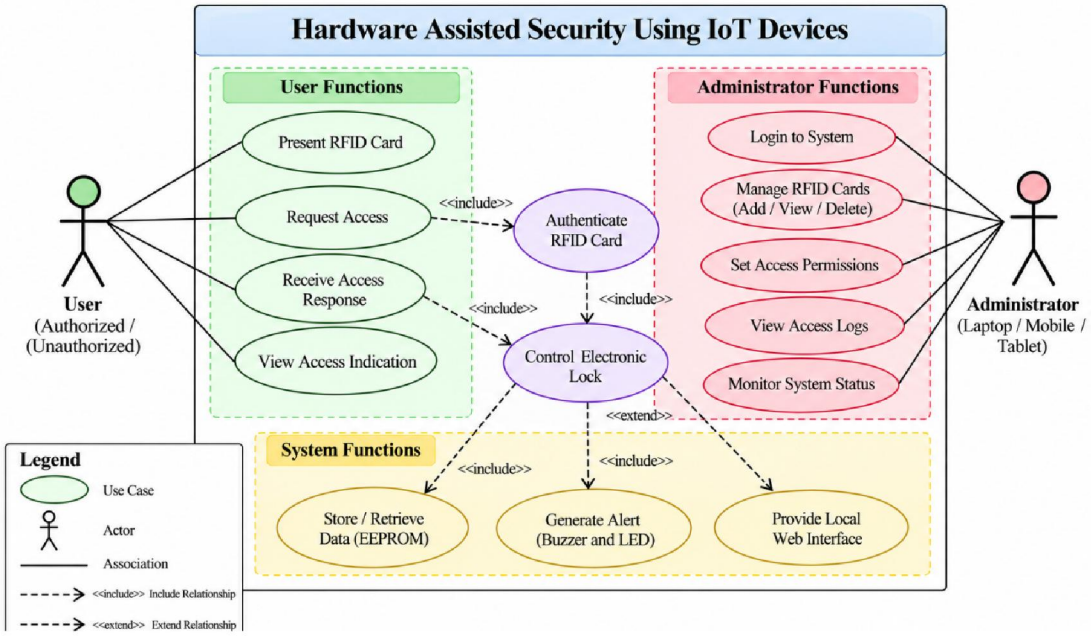

Fig. : Use Case Diagram

The core transition is between idle, authorized actuation, unauthorized alert, and temporary lockout. An implementation review should also consider edge cases: a card may be removed before reading is complete; the same card may be

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, October 2026
Impact Factor: 8.2

detected repeatedly if the reader is not debounced; an administrator may change a permission while a card event is pending; or a reset request may coincide with a write to EEPROM. The supplied report does not describe dedicated tests for all such race conditions. Future tests should define how the controller handles simultaneous browser and reader activity and confirm that the relay returns to its safe state when the main loop encounters an error or a power interruption.

VI. LOCAL WEB INTERFACE AND CREDENTIAL MANAGEMENT

The ESP8266 hosts a browser interface on its own Wi-Fi access point. An operator connects a phone or computer to the local network and opens the controller's local address. The interface is designed for administration of the stored RFID list rather than for remote cloud management. The supplied web screenshots show a compact dashboard for presenting the last-tapped card, entering a card name, changing access state, and using reset or deletion actions.

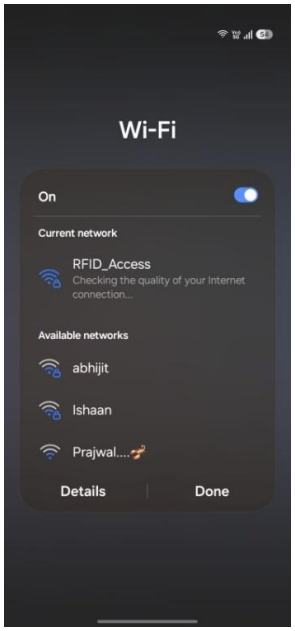

Fig. : Wi-Fi Access-Point Connection

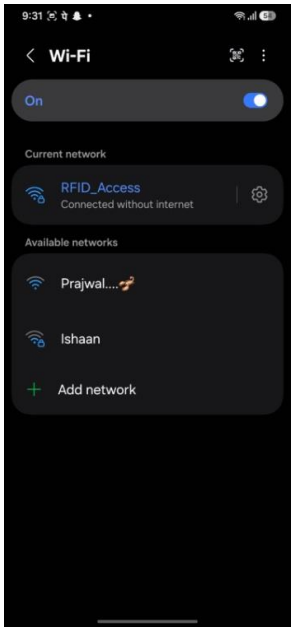

Fig. : RFID Dashboard: Initial View

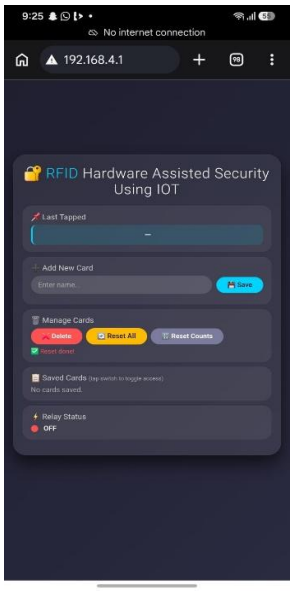

Fig. : RFID Dashboard: Registered Cards

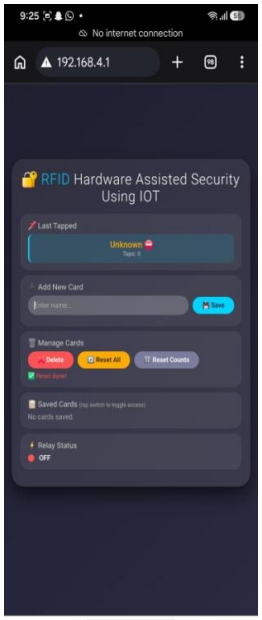

Fig. : RFID Dashboard: Access Status

TABLE III. LOCAL WEB-MANAGEMENT FUNCTIONS

Interface Function	Effect on System State
Status display	Returns the card list, last tapped card, access flag, tap count, and relay state.
Register card	Uses the most recently detected card and an entered name to add or update a record.
Delete card	Removes the registered record corresponding to the last tapped card.
Toggle access	Changes the permission flag for a selected record without deleting it.
Reset all cards	Removes all stored card records.
Reset tap counts	Clears per-card counters while retaining the card list.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, October 2026
Impact Factor: 8.2

The browser polls the status endpoint at approximately one-second intervals so the displayed information can refresh without a manual page reload. The implementation exposes HTTP handlers for status, save, delete, access toggle, card reset, and tap-count reset. The web application uses JavaScript to send these requests and update the page. The project description does not document a separate web login or encrypted transport; this is a material limitation for any network where devices other than the intended administrator may connect.

The interface's convenience comes from allowing an operator to change records at runtime rather than recompile the controller for every enrolment update. That convenience also gives state-changing routes high impact. In particular, enabling a card changes the authorization outcome of a later physical interaction, while reset-all can remove every enrolled card. An improved interface should make the selected record explicit, display the UID before destructive actions, require authenticated administration, and show a confirmation outcome after the EEPROM write has completed. It should also distinguish "registered" from "authorized" in the dashboard, since those states are not equivalent in the firmware.

VII. DATA FLOW AND STORED INFORMATION

The data-flow diagram describes the controller as the core process between the user's RFID card, the access-management operator, the actuation path, and local data storage. Input data includes the scanned UID and administrative commands. The controller produces an access decision, a physical lock or alert response, and changes to the stored card state. The web interface reads current state from the controller; it is not a separate authoritative database.

A. Card Record Model

Each stored record has four principal fields: a four-byte UID as used in the supplied implementation, a text name, an access-enabled Boolean, and a tap counter. The firmware stores and loads records through the EEPROM library. The name field is constrained by the code to a length of at most 19 characters, and the tap count is represented using two bytes. The status response serializes these fields for browser display, together with details of the last tapped card and relay state.

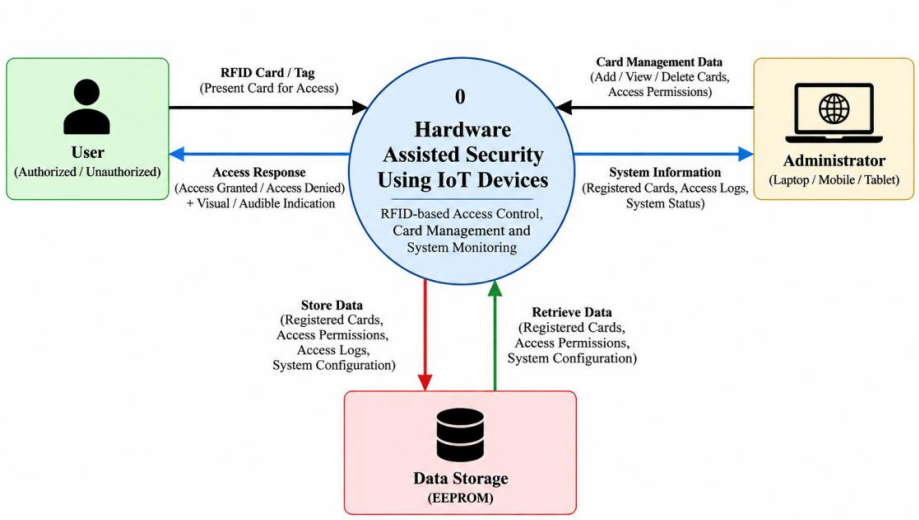

Fig. : Level 0 Data Flow Diagram

B. State and Persistence Boundaries

Persistent card data survives a controller restart when it has been written correctly, as reflected in the reported EEPROM test. In contrast, live conditions such as relay activation and the lockout timer are runtime control states. The distinction matters in recovery: a restart can reload card data, but the paper does not claim that an event history with date and time is retained. The source implementation tracks tap counts and last-tapped state, not a complete forensic log of every granted and denied event. A production system should explicitly define which states must survive resets and how interrupted writes are detected.

C. Data Quality and Administrative Integrity

A card record is only useful if its fields remain internally consistent. The current data model is compact and appropriate for a small embedded list, but a robust extension should validate UID length and uniqueness, enforce safe character handling for names, and verify record counts and bounds before reading or writing memory. Destructive operations should be auditable, and updates should be designed to recover from interrupted writes. Checksums or versioned records may help detect corrupted content, though integrity checks alone do not prevent a physical attacker from replacing the firmware or manipulating memory. These are implementation recommendations rather than features demonstrated by the current code.

VIII. USE-CASE ANALYSIS AND OPERATING ROLES

The use-case model separates the card holder's access attempt from the administrator's credential-management tasks. A card holder presents a credential and observes the physical result: the lock opens briefly if access is allowed or an audible/visual alert indicates rejection. The administrator uses the local interface to maintain the list of records, update permissions, reset counters, and view the system's current state. This role distinction is functional; the supplied project does not document an independent authentication mechanism for establishing that a browser user is an authorized administrator.

A. User-Facing Operation

The card holder does not need to enter a password at the reader. The physical workflow begins by presenting an RFID card or tag. Feedback is given locally, without requiring the user to open a browser. This keeps the normal access path short and makes the outcome visible or audible even when the management interface is not being watched. The reader-to-controller-to-relay chain should nevertheless be installed with appropriate electrical isolation and a lock mechanism suited to the door or equipment being protected.

B. Administrative Operation

Enrolment is performed by tapping a card, entering a name, and saving the record. Access permission can then be explicitly enabled. A card can be removed, the complete card list can be reset, and per-card tap counters can be cleared. Since the delete action applies to the last tapped registered card, an operator must verify which card is currently selected before deletion. The reset-all action is destructive and should require a confirmation step in a hardened interface. For usability and accountability, future versions should provide clearer confirmation, explicit selection, role-based administration, and a dated record of each configuration change.

C. Operational Scenarios

For a small laboratory, an operator could enrol cards for staff or students, explicitly enable access for approved users, and disable a card that has been lost without deleting its administrative record. For temporary equipment access, the same interface could remove a card when the permitted activity ends. The tap count provides a lightweight indication of use, although it is not a substitute for a timestamped access log. A deployment should also establish an operating procedure for lost cards, temporary lockout, power recovery, and reset-all actions, including who may perform them and how the change is recorded.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, October 2026
Impact Factor: 8.2

IX. SOFTWARE IMPLEMENTATION

The firmware is developed in Arduino IDE for the ESP8266-based NodeMCU. The MFRC522 library handles RFID reader operations; SPI provides the reader communication bus; ESP8266WiFi creates the local wireless network; ESP8266WebServer routes HTTP requests; and EEPROM stores the registered-card records [18]-[22]. The implementation brings these libraries into a single program so hardware access, web management, and local persistence can operate on the same controller.

A. Initialization and Main Loop

During startup, the program initializes serial output, starts SPI, initializes the RC522 reader, loads saved card records, configures the relay, red LED, and buzzer pins, creates the local Wi-Fi access point, and starts the web server. In the main operating loop, the controller handles web-server requests and checks for RFID cards. When a card is detected, the UID is converted to a comparable representation and searched in the card list. The result determines whether the system enters the authorized or unauthorized branch. Runtime state variables maintain relay status, last-tapped-card details, unauthorized-attempt handling, and the active alert pattern.

B. HTTP Interface and Updates

The server exposes a status route that returns a JSON representation of the card array, last-tapped information, and relay state. The save route validates the presence of a name and a recently tapped card, updates an existing record or appends a new one, and saves the result to EEPROM. The delete route removes the matching stored record. The access-toggle route switches the selected card's permission flag. Separate reset routes clear all records or clear card tap counts. The front end periodically fetches status and refreshes the visible card list and dashboard indicators.

C. Relay and Alert Timing

The relay is active only for the programmed unlock interval and is then automatically deactivated. The implementation uses millis()-based timing rather than relying solely on a blocking delay for relay duration. The alert state machine generates one short beep for accepted access and three buzzer beeps together with three red-LED blinks for a rejected attempt. When the unauthorized counter reaches three, the controller enters a temporary lock state for 15 seconds, ignoring RFID processing until it expires. These controls provide a clear behavioural contract for testing, although they are not a substitute for cryptographic authentication or a validated threat model.

TABLE IV. PRINCIPAL SOFTWARE MODULES

Module / Library	Role
MFRC522 + SPI	Read card UID and communicate with the RC522 reader.
ESP8266WiFi	Provide the controller's local wireless access point.
ESP8266WebServer	Handle status and credential-management HTTP requests.
EEPROM	Persist card identifiers, names, permission flags, and tap counts.
millis()-based timers	Turn the relay off and sequence the lockout/alert intervals.

D. Separation of Control and Presentation

The browser interface provides a view and set of commands, but the controller remains the source of truth for the current card list and relay state. The interface obtains the latest state through the status endpoint instead of keeping its own permanent copy. This reduces disagreement between browser display and controller state, provided that status requests succeed and returned data are parsed correctly. A future version should handle network timeouts visibly, avoid presenting stale information as current, and return structured error messages for invalid requests. The same API boundary could later support a mobile application, but authentication and transport security must be designed before expanding access beyond a physically controlled local network.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, October 2026
Impact Factor: 8.2

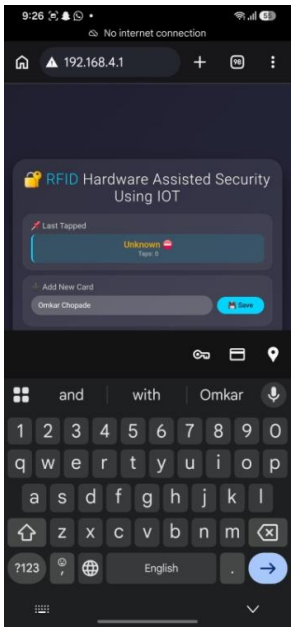

Fig. : Card Registration Interface

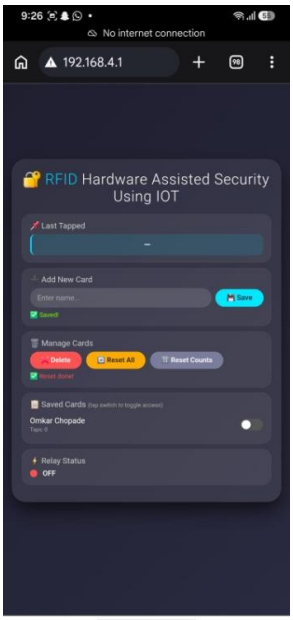

Fig. : Card Permission Controls

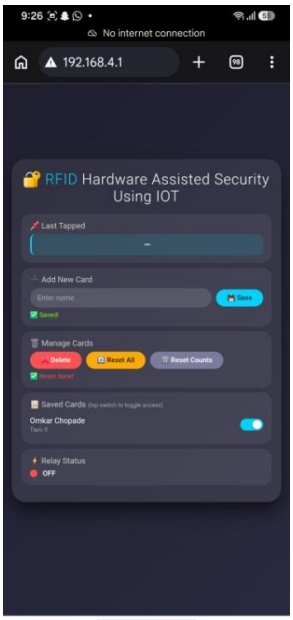

Fig. : Access-Event and Tap-Count View

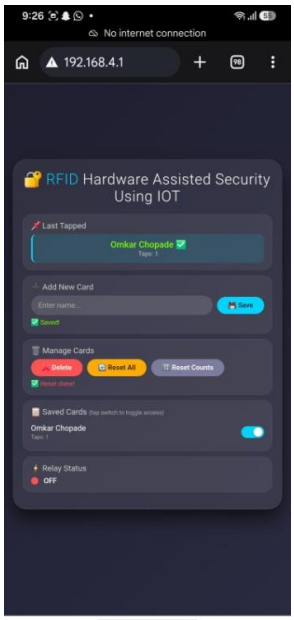

Fig. : Local Access-Control Management Screen

X. TESTING METHODOLOGY AND FUNCTIONAL VERIFICATION

The capstone report describes functional, integration, authentication, security-response, EEPROM, and interface testing. Since software commands operate physical hardware, verification includes both the logical decision and the resulting relay, buzzer, LED, memory, and web-interface behaviour. The table below consolidates the individual test cases that are present in the supplied report. The source test table includes TC-01 through TC-13 and TC-16 through TC-20; it contains no rows labelled TC-14 or TC-15, so the publication does not infer outcomes for those identifiers.

Test	Feature	Input / Action	Reported Expected Behaviour	Status
T01	RFID detection	Present a card near the RC522.	Card is detected and UID read.	Pass
T02	Authorization	Tap a registered card with access enabled.	Card is authenticated and access granted.	Pass
T03	Relay operation	Tap an authorized card.	Relay and lock activate for about 2 s.	Pass
T04	Authorized feedback	Tap an authorized card.	One short beep; red LED remains off.	Pass
T05	Tap counter	Tap an authorized registered card.	The card tap count increases by one.	Pass
T06	Disabled card	Tap a registered card with access disabled.	Access denied; relay stays inactive.	Pass
T07	Unknown card	Tap an unregistered card.	Card is identified as unknown; access denied.	Pass
T08	Denial alert	Make an unauthorized card tap.	Three buzzer beeps and three LED blinks.	Pass

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, October 2026
Impact Factor: 8.2

Test	Feature	Input / Action	Reported Expected Behaviour	Status
T09	Repeated failures	Make three unauthorized taps.	System enters a 15-second lock state.	Pass
T10	Lock-state response	Tap a card during lockout.	RFID processing is ignored during the lockout.	Pass
T11	Automatic recovery	Wait for the 15-second interval to end.	Normal operation resumes; LED turns off.	Pass
T12	Card registration	Tap a card and enter its name in the UI.	UID and name are stored successfully.	Pass
T13	Card deletion	Delete the last-tapped registered card.	Selected record is removed from the list.	Pass
T14	Reset card list	Choose Reset All in the UI.	All registered-card records are removed.	Pass
T15	EEPROM persistence	Save card information and restart.	Previously stored information is loaded.	Pass
T16	Status display	Open the local management page.	Card list, last card, and relay state display.	Pass
T17	Relay auto-off	Observe relay after an authorized access.	Relay automatically turns off after about 2 s.	Pass
T18	Startup	Power on the NodeMCU system.	Reader, GPIO pins, AP, and web server initialize.	Pass

All 18 test cases present in the source table are marked Pass. The tests establish the reported functional behaviour of the prototype under the stated actions. They do not, by themselves, establish resistance to credential cloning, network attacks, power faults, environmental variation, or high-volume concurrent use. No independent penetration test, statistical repetition count, or benchmark against a commercial access-control product is reported.

XI. RESULTS AND DISCUSSION

The documented results indicate that the primary control chain works as intended in the project's functional test environment. The RC522 detects a card and provides its UID to the controller; the controller differentiates registered enabled records from disabled or unknown cards; the relay is switched only on the authorized path; and the system returns to normal after the programmed lockout interval. The administrator workflow is also represented in the test cases: registration, deletion, reset, stored-record recovery, and status display are reported as passing checks.

TABLE VI. SUMMARY OF EVIDENCE AND EVALUATION SCOPE

Evaluation Area	Observed Behaviour Reported	Evidence Type
Identification and authorization	Registered/enabled cards accepted; disabled and unknown cards rejected.	Functional test cases
Actuation	Relay controls the lock for approximately two seconds and auto-turns off.	Functional test cases
Alerting and lockout	Denied access triggers three beeps and blinks; three failures cause a 15-second lockout.	Functional test cases
Credential management	Card enrolment, deletion, access toggling, and resets are supported	Implementation

Evaluation Area	Observed Behaviour Reported	Evidence Type
	by the local UI.	description and interface checks
Persistence and monitoring	Stored records are loaded from EEPROM; last card and relay state are displayed.	Functional test cases
Performance and security assurance	No quantified benchmark or penetration-test outcome is given.	Not evaluated in the supplied report

A. Interpretation

A principal result is integration: sensing, authorization, physical actuation, alerts, local persistence, and a management UI are implemented in one low-cost controller. The lockout behaviour demonstrates a simple response to repeated invalid use, and the permission flag permits a registered card to be retained without granting access. EEPROM avoids requiring the card list to be recreated after every ordinary restart. These features are useful for a small laboratory or classroom demonstration where local operation and easy configuration are more important than remote fleet management.

B. What the Results Do Not Establish

The evidence should not be over-interpreted. A Pass outcome shows that a test action produced the expected functional response in the reported setup; it is not a numerical estimate of false-acceptance rate, false-rejection rate, reliability, response latency, or security strength. The paper cannot report a success percentage for thousands of independent trials because the source provides no repetition counts. It also cannot assert that the local interface is safe against an attacker who obtains network access, that card identifiers cannot be cloned, or that the lock will remain secure under electrical fault or physical bypass.

C. Practical Value

For prototype development, the selected division of responsibilities is straightforward and reusable. The controller and reader can be tested independently, the relay/lock path can be checked without the web UI, and administrative endpoints can be verified against EEPROM state. This modularity supports future strengthening: a more secure card-authentication method can be integrated while retaining the same high-level permission and actuation model. A more complete evaluation should add repeated trials, event logs with timestamps, network-security tests, power-cycle tests, and measurements taken under explicitly documented conditions.

D. Suggested Repeatable Evaluation Protocol

A follow-up evaluation should define a fixed set of cards, readers, and power conditions, then repeat each authorized and unauthorized scenario enough times to estimate variability. Record the time from card presentation to relay activation, the time until relay deactivation, the frequency of missed reads, and the number of unintended unlocks observed during negative tests. Network testing should independently verify that invalid or unauthenticated requests cannot modify records. Storage testing should interrupt power at controlled points during a write and inspect the recovered records after restart. Results should report sample sizes and test conditions, and any security assessment should be performed only on an isolated test fixture rather than an operational door.

XII. SECURITY ANALYSIS, LIMITATIONS, AND FUTURE WORK

A. Security Analysis

The prototype has a local decision path, which limits its dependence on a remote service, but several security considerations remain. First, the implementation relies on the card UID for lookup. RFID literature warns that identifiers and low-cost tags must not automatically be treated as proof of a cardholder's identity; copying or emulating a UID may be possible depending on the tag and attack capabilities [2]-[5]. The supplied project does not describe a

cryptographic challenge-response exchange with the card. A registered UID should therefore be understood as a convenient identifier, not a complete high-assurance authentication factor.

Second, the controller creates a Wi-Fi access point and exposes HTTP handlers for operations that change stored state. A fixed access-point credential is present in the supplied firmware, and the project description does not document separate administrator login, encrypted transport, session management, or role-based authorization. Anyone who obtains network access may be able to reach the management interface unless additional controls are deployed. The local-only design reduces remote exposure but does not remove the need to secure Wi-Fi credentials, administrative routes, and physical access to the controller.

Third, EEPROM provides persistence but should not be treated as protected secret storage. Card records and permission flags are local to the device, and no encrypted-at-rest format or integrity-checking scheme is described. A physical attacker with access to the controller may be able to inspect or alter firmware or stored data. In addition, the prototype maintains a tap count and current/last-tapped state rather than a timestamped, tamper-evident audit log, which limits investigation after an incident. These concerns align with broader IoT security and trust literature [7]-[11], [17].

B. Limitations of the Current Evaluation

The supplied report evaluates expected functional behaviour, but does not record independent penetration testing, UID-cloning trials, stress tests, endurance cycles, environmental testing, or field deployments. The report also does not provide a component-by-component power budget, measured unlock-time variance, measured alert latency, maximum supported card count, recovery behaviour after power loss during an EEPROM write, or a fail-safe analysis of the lock in a wiring fault. The conclusions are therefore limited to implementation behaviour demonstrated by the documented tests.

C. Recommended Improvements

- Use cards and readers supporting cryptographic authentication, and avoid relying on a UID comparison as the sole proof of authorization.
- Replace fixed network credentials with configurable credentials, disable unnecessary services, and add administrator authentication and authorization to every state-changing route.
- Protect communication and session state where supported; if the device cannot provide suitable TLS, restrict management to a trusted isolated network and document that boundary.
- Add a timestamped event log for grants, denials, lockouts, permission changes, and destructive resets, with integrity protection and controlled export.
- Test cloned or replayed credentials, invalid HTTP requests, power interruptions, memory corruption, repeated rapid taps, and relay/lock failure modes.
- Consider a secure element, multi-factor authentication, backup power, a tamper switch, and centralized monitoring only after the local security boundary is hardened.

These improvements are proposed future work; they are not implemented or evaluated in the current prototype. Deployment in an institution, office, or laboratory should include an engineering review of the lock's fail-safe behaviour and local electrical requirements before the system is used to protect valuable or safety-critical assets.

D. Security Standards and Product Baselines

Standards and guidance can make the next design iteration more systematic. RFID air-interface specifications provide a basis for describing reader/tag interoperability [26], while RFID security-service specifications address security mechanisms at the air interface [27]. IoT product cybersecurity guidance also emphasizes capabilities and manufacturer practices that should be considered over a device's lifecycle [24], [25]. For networked control, end-to-end transport protection and the security of the wireless network are relevant even when operation is local [23], [28]. These references provide direction for a future hardened version; they do not imply that the present prototype implements those standards.

XIII. CONCLUSION

This paper has described a hardware-assisted IoT access-control prototype that combines RFID identification, an ESP8266 NodeMCU controller, relay-based electronic locking, audible and visual denial alerts, EEPROM storage, and a local web interface. The implementation separates card registration from permission activation, allows stored credentials to be changed without reflashing firmware, and adds a 15-second lockout after three unauthorized attempts. The supplied capstone test table reports passing results for the 18 listed functional cases, including identification, actuation, alerting, lockout, card management, persistence, interface display, and startup.

The main achievement is the integration of these functions into a low-cost and locally operable prototype. The reported results support the functional design, but they do not establish cryptographic security or production readiness. UID-based lookup, a fixed Wi-Fi credential, the lack of documented web authentication or encrypted transport, and the absence of a timestamped audit trail are important limitations. Future development should strengthen authentication, protect administrative actions and stored records, define safe behaviour during faults, and evaluate the system using repeated trials and adversarial tests. With those changes and a more complete verification protocol, the prototype can serve as a foundation for more robust access control in educational and small-scale IoT environments.

ACKNOWLEDGMENT

The authors thank **Mrs. Vaishali M. Khanapure,** Lecturer, Department of Information Technology, Government Polytechnic, Pune, for guidance and constructive feedback. The authors also acknowledge the support and facilities provided by the Department of Information Technology, Government Polytechnic, Pune.

REFERENCES

[1] R. Want, “An Introduction to RFID Technology,” IEEE Pervasive Computing, vol. 5, no. 1, pp. 25–33, Jan.–Mar. 2006, doi: 10.1109/MPRV.2006.2.

[2] A. Juels, “RFID security and privacy: A research survey,” IEEE Journal on Selected Areas in Communications, vol. 24, no. 2, pp. 381–394, Feb. 2006, doi: 10.1109/JSAC.2005.861395.

[3] S. A. Weis, S. E. Sarma, R. L. Rivest, and D. W. Engels, “Security and privacy aspects of low-cost radio frequency identification systems,” in Security in Pervasive Computing. Berlin, Germany: Springer, 2004, pp. 201–212, doi: 10.1007/978-3-540-39881-3_18.

[4] G. I. Alamillo-Montes, A. Martínez-Cruz, and C. Feregrino, “Security Scheme for an RFID Access Control based on Internet of Things,” in Proc. 2022 IEEE Mexican International Conference on Computer Science (ENC), Xalapa, Mexico, 2022, doi: 10.1109/ENC56672.2022.9882943.

[5] H. A. Abdulghani, N. A. Nijdam, and D. Konstantas, “Analysis on Security and Privacy Guidelines: RFID-Based IoT Applications,” IEEE Access, vol. 10, pp. 131528–131554, Dec. 2022, doi: 10.1109/ACCESS.2022.3227449.

[6] A. T. Hussain, T. A. Taha, S. R. Ahmed, and S. A. Ahmed, “Automated RFID-Based Attendance and Access Control System for Efficient Workforce Management,” in Proc. 2023 7th International Symposium on Innovative Approaches in Smart Technologies (ISAS), Istanbul, Türkiye, 2023, doi: 10.1109/ISAS60782.2023.10391615.

[7] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of Things: A survey on enabling technologies, protocols, and applications,” IEEE Communications Surveys & Tutorials, vol. 17, no. 4, pp. 2347–2376, 2015, doi: 10.1109/COMST.2015.2444095.

[8] J. Granjal, E. Monteiro, and J. S. Silva, “Security for the Internet of Things: A survey of existing protocols and open research issues,” IEEE Communications Surveys & Tutorials, vol. 17, no. 3, pp. 1294–1312, 2015, doi: 10.1109/COMST.2015.2388550.

[9] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, “A survey on security and privacy issues in Internet-of-Things,” IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1250–1258, Oct. 2017, doi: 10.1109/JIOT.2017.2694844.

[10] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, “Security, privacy and trust in Internet of Things: The road ahead,” Computer Networks, vol. 76, pp. 146–164, Jan. 2015, doi: 10.1016/j.comnet.2014.11.008.

[11] R. Roman, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed Internet of Things," Computer Networks, vol. 57, no. 10, pp. 2266–2279, Jul. 2013, doi: 10.1016/j.comnet.2012.12.018.
[12] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," Computer Networks, vol. 54, no. 15, pp. 2787–2805, Oct. 2010, doi: 10.1016/j.comnet.2010.05.010.
[13] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," Future Generation Computer Systems, vol. 29, no. 7, pp. 1645–1660, Sep. 2013, doi: 10.1016/j.future.2013.01.010.
[14] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," Ad Hoc Networks, vol. 10, no. 7, pp. 1497–1516, Sep. 2012, doi: 10.1016/j.adhoc.2012.02.016.
[15] T. Kothmayr, C. Schmitt, W. Hu, M. Brünig, and G. Carle, "DTLS based security and two-way authentication for the Internet of Things," Ad Hoc Networks, vol. 11, no. 8, pp. 2710–2723, Nov. 2013, doi: 10.1016/j.adhoc.2013.05.003.
[16] S. Raza, H. Shafagh, K. Hewage, R. Hummen, and T. Voigt, "Lithe: Lightweight Secure CoAP for the Internet of Things," IEEE Sensors Journal, vol. 13, no. 10, pp. 3711–3720, Oct. 2013, doi: 10.1109/JSEN.2013.2277656.
[17] J. L. Hernández-Ramos, M. P. Pawlowski, A. J. Jara, A. F. Skarmeta, and L. Ladid, "Toward a lightweight authentication and authorization framework for smart objects," IEEE Journal on Selected Areas in Communications, vol. 33, no. 4, pp. 690–702, Apr. 2015, doi: 10.1109/JSAC.2015.2393436.
[18] NXP Semiconductors, "MFRC522 Standard Performance MIFARE and NTAG Frontend," product data sheet / technical documentation.
[19] Arduino, "Arduino Documentation," online technical documentation, https://docs.arduino.cc/.
[20] Espressif Systems, "ESP8266 Arduino Core Documentation," online technical documentation, https://arduino-esp8266.readthedocs.io/.
[21] Arduino, "SPI Library," online technical documentation, https://docs.arduino.cc/language-reference/en/functions/communication/SPI/.
[22] Arduino, "EEPROM Library," and ESP8266 Community, "ESP8266WebServer Library," online technical documentation.