

MOBILE APPLICATION DEVELOPMENT BASED ON IOT TECHNOLOGY FOR SMART DOOR LOCK SYSTEM

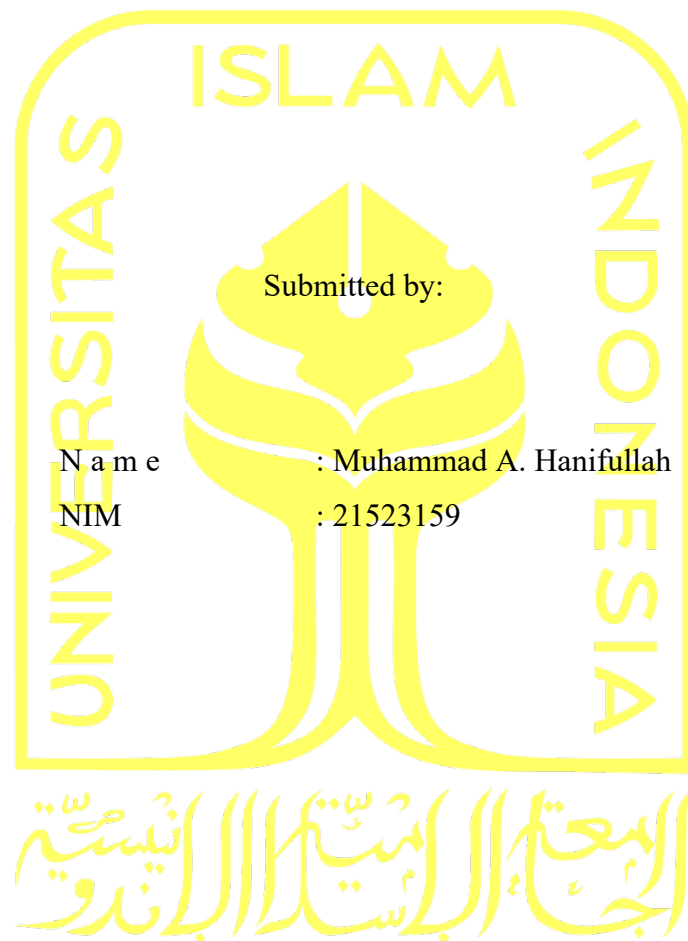


Submitted by:

N a m e : Muhammad A. Hanifullah
NIM : 21523159

**INFORMATICS STUDY PROGRAM - UNDERGRADUATE PROGRAM
FACULTY OF INDUSTRIAL TECHNOLOGY
UNIVERSITAS ISLAM INDONESIA
2025**

SUPERVISOR'S APPROVAL PAGE

**SMART DOOR LOCK SYSTEM: A MOBILE APPLICATION
DEVELOPMENT BASED ON IOT TECHNOLOGY****FINAL TASK**

Yogyakarta, 17 July 2025

Supervisor,

(Kurniawan Dwi Irianto, S.T., M.Sc.)

RATIFICATION PAGE OF THE EXAMINING LECTURER

**SMART DOOR LOCK SYSTEM: A MOBILE APPLICATION
DEVELOPMENT BASED ON IOT TECHNOLOGY****FINAL TASK**

Has been defended in front of the examiner as one of the requirements to
obtain a Bachelor of Informatics degree from the Undergraduate Program in Informatics
at the Faculty of Industrial Technology, Universitas Islam Indonesia

Yogyakarta, 7 August 2025

Board of Examiner

Kurniawan Dwi Irianto, S.T., M.Sc.

Examiner 1

Chandra Kusuma Dewa, S.Kom., M.Cs.,
Ph.D.

Examiner 2

Kholid Haryono, S.T., M.Kom.

Acknowledged by,

Head of the Undergraduate Informatics Study Program

Faculty of Industrial Technology

Universitas Islam Indonesia



(Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.)

FINAL PROJECT AUTHENTICITY STATEMENT PAGE

The undersigned:

Name : Muhammad A. Hanifullah

NIM : 21523159

Final project with title:

**MOBILE APPLICATION DEVELOPMENT BASED ON IOT
TECHNOLOGY FOR SMART DOOR LOCK SYSTEM**

Stating that all components and contents in this final project are my own work. If in the future it is proven that some parts of this work are not my own work, the final project submitted as my own work is ready to be withdrawn and ready to bear any risks and consequences.

Thus, this statement letter is made, hopefully it can be used properly.

Yogyakarta, 17 July 2025



(Muhammad Arief Hanifullah)

DEDICATION PAGE

This final project is dedicated to:

My beloved parents, Ahmad Turmudzi and Mayatun,

for your unconditional love, prayers, and unwavering support that have guided me through every step of my journey.

My uncle, Mr. Purwadi,

whose concern, generosity, and encouragement—both emotional and financial—have been instrumental in helping me reach this milestone.

All of my teachers, mentors, and educators,

especially those at Daarul Fath, Insan Kamil, and Universitas Islam Indonesia, for your knowledge, patience, and the inspiration you have instilled in me.

Everyone who has stood by me—in silence or in presence—
thank you for being part of this journey.

And to those who will come to know me in the future,

may this work serve as a small reflection of who I am and what I stood for.

MOTO PAGE

من جد وجد

“Whoever strives, finds.”

من صبر ظفر

“Whoever is patient, triumphs.”

من سار على الدرب وصل

"Whoever walks the path, reaches (their destination)."

FOREWORD

All praise and gratitude are due to God Almighty, by whose grace this final project report could be completed. This report is submitted as a requirement to fulfill the undergraduate degree in Informatics at Universitas Islam Indonesia.

The purpose of writing this report is to document the objectives, methods, and results of the final project research. Throughout the process, the author encountered several non-academic challenges, including time constraints, technical obstacles, and logistical issues. However, with perseverance and support from many parties, these challenges were gradually overcome.

The author would like to express sincere gratitude to:

1. Mr. Kurniawan Dwi Irianto, as the project supervisor, for his valuable guidance, constructive feedback, and encouragement during the preparation of this report.
2. Universitas Islam Indonesia, for awarding the full scholarship that enabled the author to pursue and complete undergraduate studies. This generous support has been a key foundation in the author's academic journey.
3. The author's beloved parents, Ahmad Turmudzi and Mayatun, for their endless prayers, unwavering support, and constant encouragement during the entire process.
4. Mr. Purwadi, the author's uncle, whose deep concern for the author's education—as well as his financial and emotional support—played a vital role in making this academic journey possible. His belief in the author's potential has been a constant source of motivation and strength.
5. The author's sister, Salma, who has been not only a sibling but also a trusted companion in the pursuit of dreams. Her continuous motivation and shared commitment were key in encouraging the author to complete this report on time.
6. Friends and colleagues, for their motivation, assistance, and companionship throughout the course of this project. Special appreciation is extended to the lecturers at the campus for their dedication and guidance, the author's fellow *asatidz* at Cahyaqu for their continuous support and understanding, and the teachers at Daarul Fath and Insan Kamil for their lasting inspiration and invaluable lessons that have shaped the author both academically and personally.
7. Peer discussion groups, online forums on Facebook, and late-night companions, whose presence—whether through shared ideas, casual conversations, or

laughter—helped sustain the author’s motivation and provided unexpected inspiration along the way.

8. All individuals who have contributed to the author’s life journey, both personally and academically, whose names may not be mentioned individually, but whose support, presence, and influence are deeply appreciated.
9. Content creators, educators, and contributors on the internet, who freely share their knowledge through tutorials, articles, and videos. Their generosity in making high-quality information freely accessible has played a significant role in helping the author overcome technical challenges and broaden understanding throughout this project.

The author acknowledges that this report may still have limitations and welcomes any constructive suggestions or feedback for improvement. It is the author’s hope that this final project may provide meaningful insights and be of benefit to future readers, especially those interested in the field of Internet of Things.

Yogyakarta, 17 July 2025

Muhammad Arief Hanifullah

ABSTRACT

This report presents the development of a miniature IoT-based smart door lock system integrated with a custom Android application named Jenaela. The system addresses the security limitations of conventional pin tumbler locks, which are vulnerable to lock-picking and offer limited access control. The hardware system consists of an ESP32 DOIT V1 microcontroller, PN532 NFC module, relay with optocoupler integration, a DC-to-DC step-up converter, and a 12V solenoid lock. The software component includes Jenaela, built using TypeScript and the React Native framework, deployed with the Expo toolkit and EAS Build Preview. Unlike prior works that utilized third-party platforms such as Blynk, Jenaela offers full customization and enhanced control. Key features include user authentication, RFID tag registration and deletion, LAN-based device discovery, and unlocking via both RFID and remote switch. Functional testing using the Black Box method confirmed that all features operate correctly. The RFID unlocking method exhibited faster response times than WAN-based unlocking due to fewer dependencies. The results demonstrate the feasibility and effectiveness of integrating IoT hardware with a custom mobile application to improve security through dual access and real-time database interaction.

Keywords: Internet of Things (IoT); Smart Door Lock; Android Application; ESP32 RFID; Firebase Real-time Database.

GLOSARY

Arduino IDE	An open-source integrated development environment used for writing and uploading code to Arduino-compatible microcontrollers, including the ESP32.
EAS	A set of cloud-based tools from Expo for building, updating, and deploying React Native apps without native configuration.
ESP32	A high-performance microcontroller developed by Espressif Systems, featuring integrated Wi-Fi and Bluetooth, commonly used in IoT applications.
Expo	A toolchain and platform built around React Native that simplifies development, testing, and deployment of mobile applications.
Firebase	A Backend-as-a-Service (BaaS) platform by Google offering services such as real-time databases, authentication, hosting, and cloud functions.
IoT	A technological paradigm that connects physical devices to the internet, allowing them to collect, exchange, and act on data in real time.
PN532	An NFC (Near Field Communication) transceiver module developed by NXP Semiconductors, used for RFID communication at 13.56 MHz
Prototype	An early working version of a system or product, built to test and refine features before final implementation.
React Native	An open-source framework developed by Meta that enables cross-platform mobile app development using JavaScript and React.
UID	A broad network that connects devices over large geographic distances, typically via the internet.

TABLE OF CONTENTS

TITLE PAGE.....	i
SUPERVISOR’S APPROVAL PAGE	ii
RATIFICATION PAGE OF THE EXAMINING LECTURER.....	iii
FINAL PROJECT AUTHENTICITY STATEMENT PAGE.....	iv
DEDICATION PAGE	v
MOTO PAGE.....	vi
FOREWORD.....	vii
ABSTRACT	ix
GLOSARY	x
TABLE OF CONTENTS	xi
LIST OF TABLES	xiii
LIST OF FIGURES.....	xiv
CHAPTER I INTRODUCTION.....	1
1.1 Background	1
1.2 Problem statement.....	2
1.3 Research objective	2
1.4 Problem scopes	3
1.5 Research contribution	4
1.6 Research methodology	4
1.7 Structure of the Thesis	5
CHAPTER II LITERATURE.....	7
2.1 Related Works.....	7
2.2 Prototype Methodology.....	10
2.3 Internet of Things.....	13
2.4 Smart Door Lock.....	14
2.5 ESP32.....	16
2.6 ESP32 Surface-mount Printed Circuit Module.....	18
2.7 React Native	19
2.8 Expo	20
2.9 Firebase	21
2.10 PN532	23
2.11 Solenoid Lock.....	24
2.12 Relay.....	26
CHAPTER III METHODOLOGY	29
3.1 Requirements gathering	29
3.1.1 Interviews.....	30
3.1.2 Observations.....	32
3.1.3 Literature study	34
3.2 Quick design	34
3.3 Prototype development	35
3.3.1 Implementation of the ESP32	35
3.3.2 Integration of Firebase Realtime Database	36
3.3.3 Development of Android application.....	37
3.4 User evaluation and refinement	39
3.4.1 User Evaluation and Refinement: First Iteration	39
3.4.2 User Evaluation and Refinement: Second Iteration.....	41
3.4.3 User Evaluation and Refinement: Third Iteration.....	43

3.5	Product implementation	45
CHAPTER IV RESULTS AND DISCUSSION.....		46
4.1	Hardware and Software Outputs	46
4.1.1	RFID Smart Door Lock.....	46
4.1.2	Jenaela Android Application.....	52
4.2	Testing.....	61
4.2.1	RFID Smart Door Lock Testing.....	61
4.2.2	Jenaela App Testing	62
CHAPTER V CONCLUSION.....		64
REFERENCES.....		65

LIST OF TABLES

Table 2.1 Related Works.....	7
Table 3.1 Respondent table.....	30
Table 3.2 Observed Available Locking System	33
Table 4.1 Hardware Components	46
Table 4.2 B6193100 Card Type.....	61
Table 4.3 ADE7BC02 Card Type.....	61
Table 4.4 Switch via WAN.....	62
Table 4.5 Jenaela Scenario-based Testing	62

LIST OF FIGURES

Figure 2.1 Prototype Method Stages.....	11
Figure 2.2 IoT Illustration.....	13
Figure 2.3 ESP32 S2 IC Chip	17
Figure 2.4 Espressif Logo	17
Figure 2.5 ESP32 WROOM IC Chip.....	18
Figure 2.6 ESP32 as a board module	18
Figure 2.7 React Native Logo	19
Figure 2.8 Expo Logo	20
Figure 2.9 Firebase Logo	22
Figure 2.10 Realtime Database Logo as a product of Firebase	22
Figure 2.11 PN532 IC Chip by NXP	23
Figure 2.12 PN532 as an NFC module board. A product by Elechouse	23
Figure 2.13 Solenoid Door Lock	24
Figure 2.14 Solenoid Lock Structure	25
Figure 2.15 Relay.....	26
Figure 2.16 Relay Structure	27
Figure 3.1 Prototype Stages	29
Figure 3.2 Quick Design on the software user interface.....	34
Figure 3.3 Quick Design on the hardware side.....	35
Figure 3.4 Database Rules are defined as public	36
Figure 3.5 Firebase Real-time Database JSON Structure.....	37
Figure 3.6 EAS Command.....	38
Figure 3.7 eas.json	38
Figure 3.8 Smart Door Lock Hardware: First Iteration	39
Figure 3.9 Account Page (1 st)	40
Figure 3.10 Home Page (1 st)	40
Figure 3.11 Login Page (1 st)	40
Figure 3.12 Switch Control Page (1 st).....	40
Figure 3.13 RFID Manager Page (2 nd).....	41
Figure 3.14 Home Page (2 nd)	41
Figure 3.15 Successfully sends to change state (2 nd).....	42
Figure 3.16 WAN Mode (2 nd).....	42

Figure 3.17 Account Page (2 nd)	42
Figure 3.18 Home Page (3 rd).....	43
Figure 3.19 Register Page (3 rd)	43
Figure 3.20 Login Page (3 rd).....	43
Figure 3.21 ESP32 Local Search (3 rd)	44
Figure 3.22 Manual Controller Page (3 rd).....	44
Figure 3.23 Custom Automatic Lock Timer (3 rd).....	44
Figure 3.24 RFID Manager Page (3 rd)	44
Figure 3.25 Smart Door Lock Miniature in the Box.....	45
Figure 4.1 Smart Door Lock Block Diagram.....	48
Figure 4.2 Final Smart Door Lock Hardware Miniature	49
Figure 4.3 Abnormality in serial monitor	51
Figure 4.4 Tap untuk Buka	52
Figure 4.5 Button for Switch via WAN	53
Figure 4.6 toggleLockState function.....	54
Figure 4.7 readRFIDCard function	56
Figure 4.8 isRFIDAuthorized function	56
Figure 4.9 saveRFIDToFirebase function.....	57
Figure 4.10 addManualUID function.....	57
Figure 4.11 scanForESP32 function	58
Figure 4.12 deleteUID function	58
Figure 4.13 openTimerModal modal	59
Figure 4.14 saveRFIDTimer function.....	60

CHAPTER I

INTRODUCTION

1.1 Background

The pin tumbler system is one of the most widely used door locking mechanisms. This locking mechanism operates by aligning the key pins and driver pins within the cylinder to unlock the system (Boros et al., 2018). Despite its widespread popularity and acceptance by the public, the pin tumbler system has significant security weaknesses, as it is vulnerable to lock-picking and forced entry (Kagawa, 2004). This vulnerability contributes to a high incidence of burglary cases, accounting for 87.19% of total criminal acts (Sun et al., 2021a). In addition, conventional locking mechanisms typically rely on a single type of key, which poses a risk of access denial if the key is lost by the authorized user. Unlike conventional pin tumbler locks, which are susceptible to lock-picking and duplication of physical keys, the proposed system eliminates the mechanical keyway entirely by employing a solenoid-based locking mechanism controlled through IoT devices. This design removes the most common attack surface exploited in burglary—manual manipulation of pins using lock-picking tools. Since access is granted only via registered RFID credentials or remote control through the mobile application, unauthorized entry using conventional break-in methods becomes significantly more difficult. In practice, this means that common intruders who rely on simple mechanical techniques cannot bypass the system as easily, thereby improving resistance to everyday burglary attempts.

Conversely, the rapid advancement of Internet of Things (IoT) technology has opened new avenues for innovation in home security, especially in the design of smart doors that incorporate layered security mechanisms. These IoT-enabled smart doors commonly combine multiple authentication methods—including RFID cards, biometric sensors, digital keypads, and facial recognition—to enhance security beyond that of traditional systems (Celestine, 2020). Implementing such multi-layered security can lower the risk of burglary by up to 20 times compared to systems with minimal or no protection (Tseloni et al., 2017).

Based on the prior findings, the author developed a miniature IoT-based smart door security system that integrates several key components, including the ESP32 DOIT V1, PN532 Elechouse V3, relay, and solenoid lock. The ESP32 serves as the microcontroller responsible for managing the input and output operations of the smart door's hardware system. The PN532

Elechouse V3, an NFC module, functions as an RFID card reader. The relay acts as a switch that controls the state of the solenoid lock. Finally, the solenoid lock, serving as the locking mechanism, operates in two states: open and closed. The ESP32 is programmed using the C++ programming language via the Arduino IDE.

In previous studies, while there are some developments that develop their own standalone Android applications such as data monitoring (Bagi et al., 2025) and water monitoring system (Hendriawan et al., 2023), IoT related development still were commonly developed using third-party applications such as Blynk. For example, Blynk was employed in the design of an IoT-based visitor monitoring system for community health centers by Ravian and D. Irianto (Ravian & D. Irianto, 2024) in the development of an IoT smart door lock system by Sun et al. (Sun et al., 2021b), automatic room temperature configuration for elderly (Rihhadatulaisy & Irianto, 2024) in the double sensor smart door project by Gamma et al. (Gamma et al., 2025), and smart lamps for the elderly (Sonya, 2025). Blynk provides features that enable users to prototype, deploy, and remotely control devices through mobile, web, and cloud-based platforms without the need for manual programming (Caleb, 2024)

In this study, the author did not utilize Blynk as the application for controlling and monitoring the developed smart door lock. The primary advantage of using a custom-built application over third-party platforms like Blynk lies in its virtually unlimited customization capabilities. The author employed React Native as the development framework and TypeScript as the programming language for the Android application. Additionally, Expo was used as a toolkit and platform to streamline the development and build processes. Finally, Visual Studio Code served as the integrated development environment (IDE) throughout the application development.

1.2 Problem statement

Traditional door locking systems, especially the widely used pin tumbler mechanism, suffer from significant security vulnerabilities such as susceptibility to lock-picking and forced entry. These weaknesses contribute to high burglary rates and present risks such as complete access denial if the physical key is lost. While IoT technology has enabled more secure, multi-layered smart door solutions, many existing implementations still rely on third-party platforms like Blynk, limiting customization and control.

1.3 Research objective

The objective of this research is to develop a basic IoT-based smart door system using ESP32, RFID, and Firebase, with a custom Android app built in React Native and Expo.

1.4 Problem scopes

This study focuses on the development of a basic-level IoT-based smart door security system, designed as a prototype using widely available hardware and software tools. The goal is to demonstrate essential functionality in a compact and accessible form, without delving into advanced-level features.

The scope is defined by the following limitations:

- a. The system includes RFID-based access control and multi-user management, providing foundational smart lock functionality. However, advanced security features such as biometric authentication, encryption algorithms, tamper detection, and audit logging are excluded from the study due to their complexity and the project's focus on core functionality.
- b. The system is built using off-the-shelf components, including the ESP32 microcontroller, PN532 RFID module, relay, and solenoid lock. Custom hardware design (e.g., PCB fabrication or low-level circuit development) is outside the scope of this research.
- c. The project follows a prototype methodology, aimed at producing a functional proof of concept. It is not designed for mass production, industrial durability, or long-term deployment. All testing is conducted in a controlled environment.
- d. The Android application is developed using React Native with Expo as the development toolkit. It supports basic operations such as locking/unlocking and managing users. However, advanced features such as biometric login on the app, role-based access control, and offline capabilities are not implemented, primarily due to time constraints and the focus on foundational functionality.
- e. The system uses Firebase Realtime Database as its backend to handle data synchronization, user management, and remote operations. While this involves cloud connectivity, advanced cloud features (e.g., analytics, push notifications, Firestore integration) are not included. The system also does not use lightweight IoT communication protocols such as MQTT or CoAP due to limited time for optimization and implementation.
- f. The research is conducted within a limited academic timeframe, which restricts the scope of implementation. As a result, the system prioritizes demonstrating a working minimum viable product (MVP) over exploring performance optimization, scalability, or advanced security and communication protocols.

- g. This research did not include burglary or forced-entry testing due to limited time and expertise. The work is intended as an early-stage prototype to demonstrate feasibility, with the main focus on functionality and user interaction

1.5 Research contribution

This study contributes to the development of IoT-based smart door systems through the following:

- a. Provides a variation of smart door system design based on IoT, integrating Firebase and a standalone Android application.
- b. May serve as a reference or comparison for future research on similar door security systems using different approaches or technologies.

1.6 Research methodology

The design of this smart door lock system employs the Prototype Method, which consists of several iterative stages as outlined below:

- a. Requirement Gathering

In this stage, the author gathers relevant data and information to serve as the foundational basis for the study. The methods employed include interviews, observation, and a literature review of available journals and books. The outcome of this stage is an analysis that will be used to guide the system development in the subsequent phases.

- b. Quick Design

In this stage, the author develops a design based on the analysis of the data and information obtained in the previous phase. The design is created in a simple and concise manner, using basic tools such as paper and pen. This stage results in a preliminary design that will serve as a reference for the subsequent development phase.

- c. Prototype Development

Subsequently, the author develops a prototype of the system based on the previously created design. At this stage, both the hardware and software components of the smart door lock system are constructed and implemented.

- d. User Evaluation

After the development of the smart door lock system—both hardware and software—the author presents the prototype to relevant stakeholders for initial feedback and evaluation.

e. Refinements

In this stage, the developed prototype is refined and modified to better meet stakeholder expectations. After the refinement process, the updated prototype is presented again to the stakeholders. This phase is iterative and closely linked to the User Evaluation stage. The iteration continues until the system meets stakeholder expectations, at which point the development proceeds to the Product Implementation phase.

f. Product Implementation

The final stage is Product Implementation. In this phase, the developed system is deployed for possible use in a real-world scenario. During this stage, the hardware, software, and database components are tested and verified to ensure that the system functions properly and reliably in its intended environment.

1.7 Structure of the Thesis

This thesis is systematically organized into five chapters, each of which addresses a specific aspect of the research:

CHAPTER I – INTRODUCTION

This chapter presents the background and rationale for the study. It includes the problem statement, research objectives, scope of the study, research contributions, research methodology, and the structure of the thesis. These sections collectively provide a comprehensive overview of the research context and direction.

CHAPTER II – LITERATURE STUDY

This chapter discusses previous research and studies relevant to the topic. It also provides an overview of the key technologies, tools, and components used in the development of the system, establishing the theoretical and technical foundation for research.

CHAPTER III – METHODOLOGY

This chapter outlines the methodology employed in the development of the smart door lock system. It details the design and implementation of both the hardware (miniature smart door lock) and software (Android-based mobile application), following a prototype-based development approach.

CHAPTER IV – RESULTS AND DISCUSSION

This chapter presents the outcomes of the development process and the results of system testing. It includes analysis and discussion of the system's performance, functionality, and its ability to meet the intended objectives.

CHAPTER V – CONCLUSION

The final chapter summarizes the conclusions derived from the study and provides recommendations for future research. These suggestions are based on the findings and limitations identified throughout the research process.

CHAPTER II

LITERATURE

2.1 Related Works

A number of prior IoT-related studies demonstrate relevance to present research. These studies are summarized in Table 2.1.

Table 2.1 Related Works

No	Research tittle	Used Method	Used Application	Results and analysis
1	Perancangan Sistem Monitoring Pengunjung Puskesmas berbasis IoT (Ravian & D. Irianto, 2024)	Prototype	Blynk	The monitoring system performs as expected; however, it is unable to distinguish between two individuals passing simultaneously—an inherent limitation. Additionally, the use of Blynk restricts customization, resulting in a generic implementation constrained by limited features.
2	IoT-Based Smart Remote Door Lock and Monitoring System Using an Android Application (Sonamoni et al., 2024)	Experimental prototype	Blynk	The ESP32-CAM smart lock system with Blynk integration enables remote control, visual alerts, and theft detection to enhance home security. However, using the legacy Blynk platform limits scalability, lacks multi-user and access logging features, and depends heavily on stable internet and external servers.
3	Implementation of Fingerprint Biometrics on Smart Door Entrance	Experimental prototype	Blynk	The study successfully demonstrates a smart door system integrating fingerprint biometrics, PIN access, and smartphone control via IoT, enhancing home security and user convenience. However, the use of

	Access Integrated with Internet of Things-based PINs (Trihandini et al., 2025)			outdated MIT App Inventor and reliance on ESP8266 with limited encryption and no user management exposes scalability, privacy, and security limitations in real-world deployment.
4	Facial Recognition Implementation in Smart Home Doors Using IoT (Irianto, 2025)	Experimental Prototype	Blynk and web app interface	The system successfully enables secure door access using ESP32-CAM with facial recognition and IoT integration, offering fast response and local processing without relying on cloud servers. However, it suffers from limited face detection range, sensitivity to camera angle and lighting conditions, and dependence on outdated Blynk architecture with weak support for multi-user access, logging, and long-term scalability.
5	Perancangan Aplikasi Mobile Berbasis Flutter untuk Pemantauan Data Sensor IoT Solusi bagi Manajemen Suhu dan Kelembapan (Hendriawan et al., 2025)	Experimental Prototype	Custom standalone Android application built in Flutter	The Flutter-based mobile application for IoT sensor monitoring presents a strong real-time solution for temperature and humidity management, integrating ESP32, MQTT, and MySQL with clean UI (UAT 90%) and responsive notifications. However, its reliance on a basic hosted MySQL setup and manual Web API linkage introduces scalability bottlenecks, limited data security, and lack of advanced user-role management or data analytics features—making it less adaptable for larger industrial deployments or multi-tiered environments.

6	Designing an Automatic Room Temperature Control System for Smart Homes for the Elderly Using IoT (Rihhadatulaisy & Irianto, 2024)	Experimental Prototype	Blynk	The system successfully maintains room temperature for elderly users using ESP8266, DHT22, and Blynk-based remote control. However, it's limited by Blynk's lack of data logging, offline support, and scalability for advanced smart home features.
7	Sistem Keamanan Pintu Otomatis Berbasis IoT dengan Teknologi (Assyauqi et al., 2025)	Prototype	Android Application (details are not specified)	The research successfully demonstrates the integration of RFID and IoT technologies in an automatic door security system, enhanced by the application of the Fuzzy Mamdani method for dynamic decision-making. Key strengths include remote control via an Android application, real-time notifications, and the use of fuzzy logic to reduce security risks in specific scenarios. However, the paper lacks critical details regarding the development of the Android software itself—there is no mention of the programming language, frameworks used, or the involvement of stakeholders in UI/UX design. Additionally, the absence of RFID card management features, such as adding or removing cards, limits the system's practicality for daily use. The evaluation section is also underdeveloped, with no performance metrics. Lastly, the scalability of the

				system and its adaptability to multi-user or multi-door environments remain unexplored.
--	--	--	--	---

All the studies listed in Table 2.1 employed the prototyping method, though the specific stages and implementation details varied across each work. Most previous studies utilized the Blynk platform to control IoT devices, chosen for their ease of use and minimal programming requirements. However, this convenience introduces limitations, such as dependency on third-party services and restricted customization options—issues that are particularly critical in IoT deployments. In contrast, one newer study developed a custom Android application integrated with RFID and fuzzy logic to enhance door security and decision-making capabilities. While this approach offers greater flexibility and tailored functionality, the paper lacks detailed documentation of the application’s development process, including programming tools, user interface considerations, and stakeholder involvement. Moreover, the system’s scalability remains unaddressed, with no discussion on how it could adapt to multi-user or multi-door environments—an essential factor for broader deployment. This highlights a broader trend: in certain use-case scenarios, application customization and scalability are essential not only for usability but also for ensuring system reliability and contextual relevance.

2.2 Prototype Methodology

Prototyping is a widely adopted iterative development methodology that focuses on refining system functionality through repeated cycles of design, implementation, and stakeholder feedback. Its primary goal is to ensure that the final product aligns closely with user expectations and functional requirements, particularly in cases where initial requirements are uncertain or likely to evolve over time. Unlike traditional waterfall methods, which require full specification before development begins, prototyping embraces flexibility and adaptability, making it suitable for dynamic environments and user-centric applications (Pressman & Maxim, 2014; Sommerville, 2016).

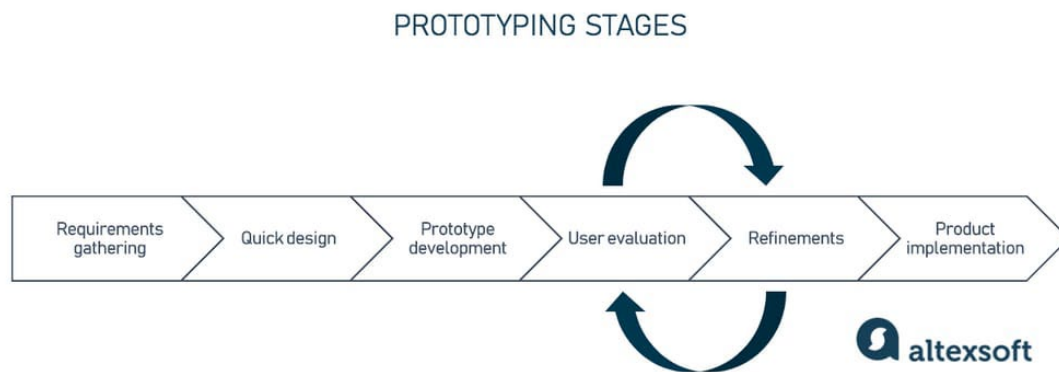


Figure 2.1 Prototype Method Stages

Source: altexsoft.com

The prototyping process is typically divided into several distinct phases: (1) requirements gathering, (2) quick design, (3) prototype development, (4) user evaluation, (5) refinements, and (6) product implementation. In the requirements gathering phase, developers work closely with stakeholders to identify core needs, often using interviews, observations, and use case modeling. However, instead of attempting to capture all requirements upfront, this stage aims to identify essential functionalities to guide the initial prototype. This approach acknowledges that users often struggle to articulate detailed requirements until they have interacted with a working version of the system (Klotins et al., 2023).

The next stage, quick design, focuses on creating a conceptual or preliminary design based on the initial requirements. This includes designing user interfaces, system architecture sketches, and user interaction flows. Speed and simplicity are prioritized in this phase to quickly produce a foundation for prototyping. Following this, the prototype development stage involves the actual construction of a working model of the application. The prototype may not include full functionality or backend integration at this stage—it serves as a functional simulation to allow users to interact with the interface and provide feedback (Bombarda et al., 2021)

In the evaluation and refinement phase, stakeholders interact with the prototype, and their feedback is used to modify the design. This is the most iterative phase, and may involve multiple feedback cycles until the system meets stakeholder expectations. Importantly, this phase allows for early detection of usability issues, misunderstanding of requirements, or technical limitations. Research by Nguyen-Duc et al. (2017) and Ardito et al. (2018)

emphasizes that prototyping significantly improves user satisfaction and design accuracy, especially in systems with complex interfaces. However, without proper iteration controls—such as timeline constraints or evaluation metrics—this phase can become resource-intensive and potentially lead to scope creep.

Once stakeholders are satisfied with the prototype, the system moves into the product implementation phase. Here, the refined prototype may either be evolved into the final product or be completely redeveloped based on the validated design. The decision depends on whether the prototype was built as a disposable proof-of-concept or an evolutionary model intended to serve as the development foundation. While disposable prototyping is faster and more flexible, evolutionary prototyping supports continuous integration but requires greater initial planning to ensure maintainability and scalability (Pressman & Maxim, 2014; Sommerville, 2016).

Despite its limitations—such as possible delays due to excessive iteration and increased developer reliance on user input—prototyping continues to be a favored methodology, particularly in small to medium-sized projects, where teams are agile and stakeholder feedback loops are more manageable. According to recent studies, including the empirically validated model by Bjarnason et al. (2023), prototyping significantly contributes to risk mitigation, user satisfaction, and achieving product–market fit, making it indispensable in iterative and user-focused design processes.

2.3 Internet of Things

The Internet of Things (IoT) has emerged as a central paradigm in modern computing, driven by the seamless integration of physical and digital environments. At its core, IoT enables objects in the physical world to be embedded with sensors, software, and communication capabilities that allow them to generate, exchange, and process data via the internet. These “things” can function independently or collaboratively within a networked infrastructure, making systems smarter, more responsive, and context-aware. (Gubbi et al., 2013) emphasize that IoT goes beyond traditional machine-to-machine communication by introducing cloud computing and big data analytics as foundational pillars for intelligent decision-making across various sectors.



Figure 2.2 IoT Illustration

Source: vecteezy.com

Modern definitions of IoT incorporate not only the connectivity of devices but also their virtual representation and autonomous behavior. According to the International Telecommunication Union (ITU), IoT is defined as a global infrastructure for the information society, enabling advanced services by interconnecting both physical and virtual “things” through interoperable information and communication technologies. This view is echoed by Atzori, Iera, and Morabito (Atzori et al., 2010), who describe IoT as a dynamic network where things are not only identifiable but also capable of interacting with their environment. These

interactions allow systems to respond to stimuli without direct human intervention, significantly expanding the functional boundaries of traditional computing systems.

The architecture of IoT is supported by multiple interrelated technologies. Borgia (2014) categorizes core IoT components into radio frequency identification (RFID), wireless sensor networks (WSNs), embedded systems, and cloud-based data processing. These elements collectively enable real-time monitoring and adaptive control in domains such as healthcare, logistics, agriculture, and urban infrastructure. For example, in a smart city context, IoT devices can detect traffic conditions, optimize energy consumption, and manage waste collection—all through real-time data flow and autonomous decision logic. As IoT deployments become more complex, the need for standardized architectures such as IEEE P2413 becomes crucial to ensure interoperability, scalability, and security across devices and platforms.

While technological advancement remains essential, recent research has shown that human-centered design and usability are equally critical to the success of IoT systems. Ardito et al. (2018) argue that many IoT implementations fail due to poor user interface design and insufficient consideration of user needs, leading to low adoption rates. Their study promotes an end-user development approach, where users can configure or adapt smart environments without deep technical knowledge. This democratization of development fosters broader usability and ensures that systems serve practical, real-world use cases. The emphasis is on making interfaces intuitive, flexible, and accessible—especially in domestic or non-industrial IoT applications.

Complementing this perspective, Alansari et al. (2018) present a five-layer IoT architecture that addresses not just the functionality of the system but also its security and privacy concerns. These layers—perception, network, middleware, application, and business—highlight how data must be protected at every point of transmission and processing. Their study emphasizes the importance of identity authentication, secure communication, and user data governance, particularly in applications where sensitive information (e.g., medical records or behavioral patterns) is collected. This security-by-design approach is essential to maintaining user trust and ensuring the long-term viability of IoT ecosystems.

2.4 Smart Door Lock

The concept of the smart door lock introduces a paradigm shift from conventional mechanical locking mechanisms—typically involving PIN tumbler systems—toward integrated, technology-enabled access control solutions. Unlike traditional locks, smart door locks often incorporate biometric authentication (e.g. fingerprint or face recognition),

passwords, RFID cards, or voice recognition. These advanced features enable more flexible, secure, and user-friendly access control. Research by (Zainuddin et al., 2024) describes a smart door lock system that combines fingerprint recognition and RFID access, controlled via ESP8266 and MQTT-based communication, achieving improved security and real-time monitoring capabilities through mobile integration.

Within IoT-enabled smart home environments, smart door locks must be capable of seamless interactions with other connected devices—most commonly smartphones. For instance, Permana et al., (2024) developed an IoT-based smart lock system integrating fingerprint and keypad authentication with Android-based control via Firebase real-time communication. Their results demonstrated complete success across multiple trials, highlighting the reliability of IoT integration for access control systems.

Technologically, typical smart door locks utilize microcontrollers such as the ESP32 or ESP8266, paired with sensors like fingerprint modules, RFID readers, and keypads. Kurniasyah & Rakhmawati (2025) implemented a BLE-enabled smart door lock prototype that integrates ESP32, RFID, fingerprint, and keypad authentication. They utilize MQTT over BLE for real-time communication, achieving low latency and high throughput, with system responsiveness validated through RSSI analysis. The design also includes proximity validation to enhance security performance.

Of particular importance is the multi-factor authentication and IoT integration that distinguish smart locks from conventional mechanisms. KAYA et al., (2022) present a system design featuring ESP32, fingerprint sensors, RFID card readers, and keypad inputs, all controllable remotely via a mobile app. Such architectures enable versatile control and monitoring capabilities, where users can manage access anywhere via secure Wi-Fi, and receive instant notifications or audit logs—a significant advancement over physical key systems.

Despite promising features, smart door lock implementations face significant security and usability challenges. For example, BLE-based systems can be vulnerable to man-in-the-middle (MITM) attacks, if encryption or pairing protocols are weak (Bapat et al., 2019). Recent studies in smart padlock vulnerabilities, such as Droplock attacks targeting fingerprint modules, further emphasize the necessity of robust security-by-design approaches—including secure biometric storage, encrypted communication channels, and intrusion detection mechanisms(Kerrison, 2024).

2.5 ESP32

The ESP32, including ESP32 WROOM that used in this smart door lock development, is a high-performance, low-power system-on-chip (SoC) microcontroller developed by Espressif Systems and fabricated by TSMC using a 40 nm CMOS process. It represents a significant leap over its predecessor, the ESP8266, in terms of computing power, energy efficiency, and integrated functionality. The original ESP32 includes a dual-core Tensilica Xtensa LX6 processor operating at up to 240 MHz, capable of delivering up to 600 DMIPS, which makes it well-suited for compute-intensive embedded applications. According to a performance analysis by Maier, Sharp, and Vagapov (2017), the ESP32 supports rapid data processing and multitasking, making it ideal for IoT edge devices and automation systems.

Over time, Espressif has expanded the ESP32 family to include additional variants such as the ESP32-S2, S3, and the newer RISC-V-based ESP32-C series (C2, C3, C5, C6, and H2). While the S2 and S3 continue using the Xtensa LX7 core architecture with enhanced peripheral support, the C-series adopts single-core RISC-V (RV32IMC) processors with varying maximum clock speeds. For example, the ESP32-C3 runs at up to 160 MHz, includes 400 KB SRAM, 384 KB ROM, and supports both Bluetooth LE 5.0 and Wi-Fi 802.11 b/g/n. These chips maintain compatibility with the ESP-IDF development environment and offer modern cryptographic security features such as RSA-3072 secure boot and AES-XTS flash encryption (Espressif Systems, 2020).

In terms of architecture, the ESP32 family supports a wide array of integrated peripherals and memory configurations. The ESP32-C3 includes 22 GPIO pins, 12-bit ADC, SPI, UART, I²C, CAN 2.0, and PWM controllers, in addition to a DMA engine for high-speed data transfer. Power efficiency is a standout feature: ultra-low-power modes and an integrated ULP (ultra-low power) coprocessor allow the ESP32 to operate in deep sleep at current levels as low as 5 μ A, which is essential for battery-powered applications. This flexibility allows developers to tailor power usage according to application needs, making the ESP32 a favorite in IoT deployments requiring long operational life with limited power sources (Espressif System, 2020).

Despite lacking a dedicated neural accelerator, the ESP32 demonstrates capability for embedded machine learning. Zim (2021) conducted an analysis of neural network inference performance using the Xtensa LX6 processor, showing that feed-forward artificial neural networks (ANNs) could run effectively on the ESP32 with carefully optimized memory and CPU utilization. The processor's ability to handle basic ML workloads at the edge makes it a

viable platform for TinyML scenarios such as anomaly detection, sensor fusion, and gesture recognition in low-resource environments.

Security is another critical strength of the ESP32 platform which is based on the dual-core Xtensa LX6 microprocessor architecture. Newer variants such as the ESP32-C3 implement hardware-accelerated cryptographic functions including AES, SHA, HMAC, and RSA, alongside secure boot mechanisms and efuse-based key storage to ensure firmware authenticity and device identity protection. This security design supports deployment in sensitive environments such as industrial IoT, smart locks, and wearable devices. The ESP32's blend of processing power, peripheral versatility, wireless integration, and secure architecture makes it one of the most reliable and cost-effective microcontrollers in modern embedded systems.

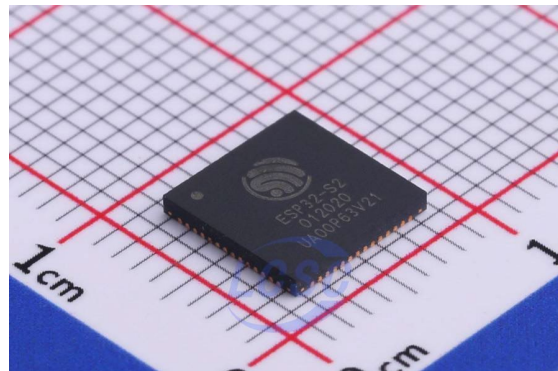


Figure 2.4 ESP32 S2 IC Chip

Source: <https://jlcpcb.com/partdetail/EspressifSystems-ESP32S2/C529594>



Figure 2.3 Espressif Logo

Source: espressif.com

2.6 ESP32 Surface-mount Printed Circuit Module

The ESP32 microcontroller is typically integrated into a module board to facilitate ease of use for end users. These module boards, which are widely available on the market, incorporate the ESP32 integrated circuit (IC) and are designed to provide essential supporting components such as voltage regulators, crystal oscillators, antennas, and USB-to-serial interfaces. This integration simplifies the development process, allowing users to focus on software and application-level development without needing to construct the supporting hardware architecture from scratch.



Figure 2.5 ESP32 WROOM IC Chip

Each ESP32 module board generally differs in terms of features, form factor, GPIO availability, antenna type (PCB or external), memory capacity, and power management circuits, depending on the intended application. Some boards are optimized for compact designs, while others prioritize extended I/O capabilities or wireless communication enhancements. The diversity of these modules enables the ESP32 platform to address a wide range of Internet of Things (IoT) use cases, from simple automation systems to more complex edge computing tasks.



Figure 2.6 ESP32 as a board module

Source: <https://www.voltaat.com/products/doit-esp32-devkit-v1-development-board>

Several manufacturers and vendors produce ESP32 module boards with varying specifications and embedded components. Notable providers include Espressif (the original developer of the ESP32), Ai-Thinker, AnalogLamb, DFRobot, eBox & Widora, MagicBIT, DoIT, and Arduino. These vendors offer development boards such as the ESP32-WROOM, ESP32-WROVER, DevKitC, and application-specific boards like the ESP32-CAM or TTGO LoRa32, many of which are compatible with open-source development environments like Arduino IDE or ESP-IDF.

By offering pre-assembled and pre-certified hardware modules, these development boards significantly reduce the entry barrier for developers, educators, and hobbyists. The availability of extensive documentation, community support, and firmware libraries further enhances the usability of ESP32 module boards in academic research, prototyping, and commercial product development. Their plug-and-play nature, along with robust wireless and peripheral support, makes them a cornerstone in the growing ecosystem of embedded IoT systems.

2.7 React Native

React Native is a robust, open-source framework developed by Meta (formerly Facebook) and officially released in March 2015. Utilizing JavaScript and React, it allows developers to create mobile applications for iOS, Android, web, and desktop platforms from a unified codebase. Developers can achieve near-native performance and UI by leveraging React Native's underlying JavaScript-to-native bridge architecture.



Figure 2.7 React Native Logo

Source: pinpng.com

The key strength of React Native is its capacity for cross-platform code reuse. Peer-reviewed comparative research indicates that developers can share a large portion of application logic and UI components between platforms, leading to major reductions in

development time and maintenance effort when compared to native app development (Shevtsiv et al., 2019).

Performance-wise, React Native generally provides satisfactory results for many mobile applications. However, because it relies on a bridge to relay UI updates between JavaScript and native layers, applications with frequent state changes or heavy UI rendering can encounter latency or frame rate drops—highlighted by studies contrasting React Native with frameworks like Flutter (Shevtsiv et al., 2019).

Regarding accessibility, (Mascetti et al., 2020) conducted an empirical study on React Native’s support for screen-reader APIs. While core accessibility features are supported, some advanced native APIs are not directly exposed, meaning developers must implement platform-specific native modules to achieve full functional accessibility—especially important for inclusive purposes.

React Native has evolved significantly since its launch, and scholarship such as that by (Goli, 2021) highlights best practices including the use of native modules to enhance performance-critical tasks. These additions offer flexibility and scalability, although they introduce further complexity and can impact cross-platform portability—requiring careful architectural design.

2.8 Expo

Expo is an open-source toolchain designed to streamline the development of React Native applications, significantly reducing the complexity of building, deploying, and iterating cross-platform mobile apps. As noted in a technical evaluation by Shiraz Ali and Shubham (2022), Expo enables developers to create universal applications for iOS, Android, and the web using a single JavaScript-based codebase—without requiring configuration with native IDEs like Android Studio or Xcode. This managed workflow accelerates setup, allows instant prototyping via QR codes, and simplifies the development lifecycle while maintaining compatibility with standard React Native capabilities (Shiraz Ali, 2020).



Figure 2.8 Expo Logo

Source: expo.dev/brand

The key feature of Expo is Expo Go, a companion Android/iOS application that serves as a real-time UI preview tool. Developers can instantly view and interact with changes to their application code without manually building native binaries—this immediate feedback loop enhances productivity, reduces iteration time, and makes Expo especially suitable for rapid prototyping and early-stage development (Shiraz Ali, 2020).

Beyond development convenience, the Expo SDK provides a comprehensive set of native modules bundled in the managed framework. These modules include access to device capabilities such as camera, push notifications, geolocation, and local storage, eliminating the need for developers to independently configure or write platform-specific native modules. The abstraction ensures high portability and minimizes the need for ejecting to native tooling—leading to lower overhead and consistent deployment behavior across environments.

Expo also supports integrated build and distribution services, including over-the-air (OTA) updates, that allow developers to push changes to deployed apps instantly without app store re-submissions. This enables bug fixes and feature enhancements to be delivered rapidly, enhancing responsiveness to user feedback and reducing time-to-market.

Expo is broadly recognized in the developer community for offering a production-grade React Native framework, especially for teams that require speed and simplicity without sacrificing app quality. According to Pragmatic Engineer analysis, Expo's toolset—including the Expo CLI, managed app workflows, and cloud build services—constitutes a mature ecosystem that balances ease of use with scalability for production applications (Orosz & Nilsson, 2025).

2.9 Firebase

The Firebase platform originated from a San Francisco startup named Envolv, founded in 2011 by James Tamplin and Andrew Lee. Initially, Envolv provided a real-time chat widget API, which users repurposed to synchronize application data such as game states across clients. Recognizing this demand, Tamplin and Lee pivoted and formally launched Firebase in April 2012, introducing the Firebase Realtime Database, an API that enabled real-time synchronization of data across iOS, Android, and web clients without a custom back-end server.



Figure 2.10 Firebase Logo

Source: firebase.google.com



Figure 2.9 Realtime Database Logo as a product of Firebase

Source: firebase.google.com

In 2014, Google acquired Firebase and fully integrated it into its cloud ecosystem, transforming it into a comprehensive Backend-as-a-Service (BaaS) platform. This transition allowed Firebase to expand beyond real-time data sync, adding essential services such as Firebase Hosting, Authentication, and later Cloud Firestore, Cloud Functions, Machine Learning, and Cloud Storage, while maintaining its real-time capability and client-centered approach (Saraf, 2022).

Firebase's Realtime Database works on a NoSQL JSON tree structure that automatically synchronizes data with all connected clients and handles offline persistence using local caching. An IOP Conference Series study by (Sudiarta et al., 2020) demonstrated that Firebase significantly reduces latency in IoT-based tourist tracking applications compared to relational databases, especially under mobile network variability, thereby facilitating near real-time updates even in constrained environments.

As demonstrated in multiple academic case studies, Firebase Realtime Database and Firestore are widely adopted in IoT systems for remote control and monitoring. For example, (Padmaja et al., 2021) implemented a scalable smart-home system using Firebase as the central data backend, leveraging real-time updates for seamless control via a consumer portal interface.

This prototype illustrates Firebase's integrative capacity and underscores its suitability for cloud-connected IoT applications.

2.10 PN532

The PN532, developed by NXP Semiconductors, is a highly integrated NFC transceiver chip designed for 13.56 MHz contactless communication. It is built on an 80C51 microcontroller core and equipped with 40 KB of ROM and 1 KB of RAM, enabling it to support a broad range of NFC communication protocols. These include ISO/IEC 14443A/MIFARE, ISO/IEC 14443B (reader/writer mode only), FeliCa, ISO/IEC 18092 NFCIP-1 Peer-to-Peer, as well as card emulation modes for both MIFARE and FeliCa standards (NXP Semiconductors, 2017). The chip is capable of operating at data rates of up to 424 kbit/s, making it suitable for a variety of contactless communication applications.



Figure 2.11 PN532 IC Chip
by NXP

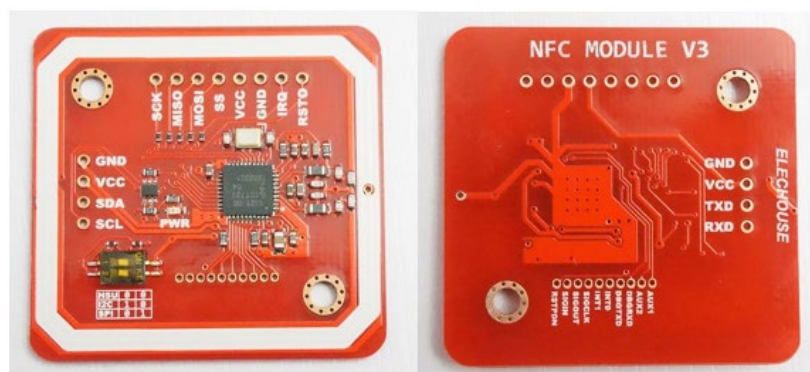


Figure 2.12 PN532 as an NFC module board.

Designed with interface flexibility in mind, the PN532 supports communication via SPI, I²C, and High-Speed UART (HSU), allowing for easy integration with a wide range of

microcontrollers and development boards (NXP Semiconductors, 2017). Additionally, it features an integrated low-dropout voltage regulator, which permits direct operation from battery sources within a voltage range of 2.7 V to 5.5 V. This power versatility simplifies the hardware design process for embedded systems and IoT applications, where low power consumption and compact circuit integration are critical (NXP Semiconductors, 2017).

2.11 Solenoid Lock

The solenoid lock operates via an electromagnetic coil (solenoid) that, when energized, generates a magnetic field to actuate a plunger—physically extending or retracting a latch. This linear motion mechanism enables robust and rapid locking/unlocking actions. The term “solenoid” was first coined in the early 1820s by André-Marie Ampère in his electrodynamics work, though the use of solenoids in automatic locking mechanisms became practical much later. The modern direct-pull electromagnetic lock, leveraging solenoid technology, was invented in 1969 by Sumner “Irving” Saphirstein, and widely deployed in secure door installations such as at the Montreal Forum. Saphirstein’s design remains foundational in contemporary electromagnetic locking systems—particularly for security-critical access control applications (Viatch, 2021).

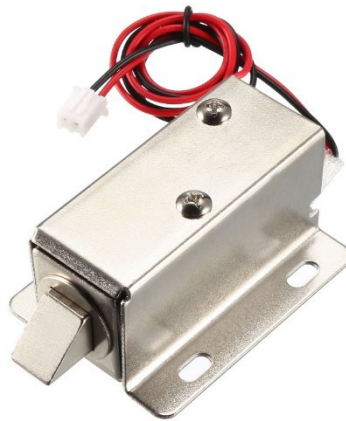


Figure 2.13 Solenoid Door Lock

Source: <https://www.walmart.com/ip/DC12V-1-1A-11-4mm-Electromagnetic-Solenoid-Lock-Assembly-for-Electirc-Lock-Cabinet-Door-Lock/208199926>

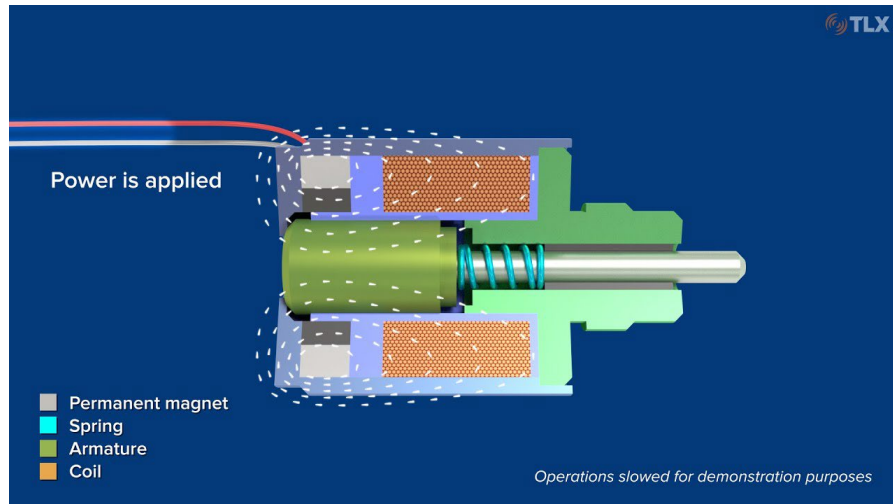


Figure 2.14 Solenoid Lock Structure. Source: TLX Tech

By the time of this study, solenoid locks had become a popular locking mechanism in IoT-based smart door lock projects. These systems typically integrate a 12 V DC solenoid lock, a relay module, and a microcontroller (e.g., ESP8266 or ESP32) to enable remote control.

Solenoid locks offer significant advantages for IoT integration due to their simple electrical interface and fast mechanical response. In IoT prototypes such as those using ESP-01 or ESP32 controllers, the solenoid lock connects via a relay module and is activated by digital control signals. One research project leveraged Arduino Uno, ESP-01, and an RFID reader to authenticate users via RFID tags and then operate the solenoid lock—achieving effective and scalable access control with minimal hardware complexity.

From an engineering standpoint, solenoid lock modules are valued for their low cost, simplicity, and ease of control. They are typically driven by 5 V or 12 V DC supplies via relays, and require no complex feedback mechanisms. Their quick actuation makes them suitable for systems requiring immediate locking/unlocking, such as smart door locks that respond to remote commands or sensor triggers. These features, together with reliable lock holding forces and small form factor, make solenoid locks ideal for embedded IoT environments.

2.12 Relay

Electromechanical relays function as electrically controlled switches that rely on a copper coil wound around an iron core. When energized, the coil generates a magnetic field that attracts an iron armature, which in turn actuates a switch contact. Once the coil is de-energized, a spring mechanism returns the armature to its default position, opening or closing the circuit accordingly. This arrangement allows precise control over circuits with minimal input power and provides reliable electrical isolation between control and load circuits (Vinod et al., 2018).

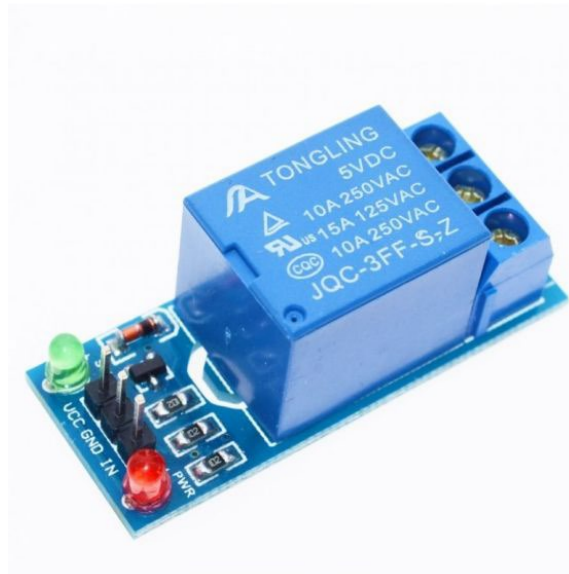


Figure 2.15 Relay

The essential components of a relay include the coil, armature, spring, and one or more switch contacts. The coil produces the magnetic flux; the armature is the moving ferromagnetic element; the contacts, actuated by the armature, open or close the circuit; and the spring ensures the armature returns to its rest position upon deactivation. This mechanical ensemble ensures both operational responsiveness and contact reliability.

Electromechanical relays employ two principal contact states—Normally Open (NO) and Normally Closed (NC). The NO contact remains open until activation, at which point the armature closes the circuit. Conversely, the NC contact remains closed until energized, at which point the circuit is opened. These configurations emulate traditional on/off switches and are foundational to control logic designs in industrial and IoT systems.

Relay contact configurations are described using the terminology of pole and throw. The term pole refers to the number of isolated circuits that a relay can switch simultaneously (e.g.

SP, DP). Meanwhile, throw indicates the number of output positions per pole (e.g. single or double throw). Common configurations include SPST (single-pole single-throw), SPDT, DPST, and DPDT—each offering distinct application flexibility and isolation capabilities (Vinod et al., 2018). Common relay structure can be seen in Figure 2.16.

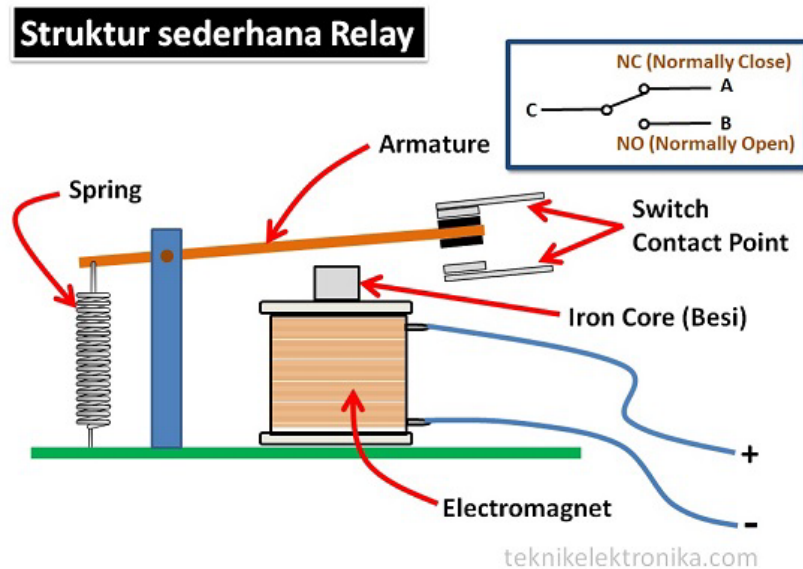


Figure 2.16 Relay Structure

Source: teknikelektronika.com

Based on the combination of poles and throws, relays are typically grouped into several standardized categories. The first is the Single Pole Single Throw (SPST) configuration, which is the simplest type of relay. It consists of four terminals: two for the switch (normally open or normally closed) and two for the coil. This relay operates as a basic on/off switch for a single circuit:

- a. The Single Pole Single Throw (SPST) configuration, which is the simplest type of relay. It consists of four terminals: two for the switch (normally open or normally closed) and two for the coil. This relay operates as a basic on/off switch for a single circuit
- b. The Single Pole Double Throw (SPDT) relay includes five terminals: three assigned to the switch—comprising a common contact, a normally open (NO), and a normally closed (NC) terminal—and two terminals for the coil. This configuration allows one input (pole) to toggle between two outputs (throws), making it suitable for changeover applications.

- c. The Double Pole Single Throw (DPST) relay has six terminals—four used for switching two separate circuits simultaneously (i.e., two pairs of switch terminals) and two for the coil. This category is often employed in scenarios requiring the activation of two independent loads using a single control signal.
- d. The Double Pole Double Throw (DPDT) relay features eight terminals: six used as two independent SPDT switches and two dedicated to the coil. The two SPDT mechanisms operate in parallel, simultaneously toggled by a single electromagnetic coil. This configuration is widely used in applications that require dual channel switching with inversion or crossover functionality.

Additionally, there are custom relay configurations designed for specific applications that fall outside the typical SPST, SPDT, DPST, and DPDT classifications. These relays are engineered for unique circuit requirements and are not commonly found in general-purpose control systems. Their contact arrangements and terminal counts may vary significantly depending on the logic or redundancy required in the application.

CHAPTER III

METHODOLOGY

This study adopts the prototyping method in the development process of an IoT-based smart door security system and its corresponding Android application. The prototyping approach was selected due to its iterative nature, which allows continuous refinement and revision of the prototype until a final functional version is achieved.

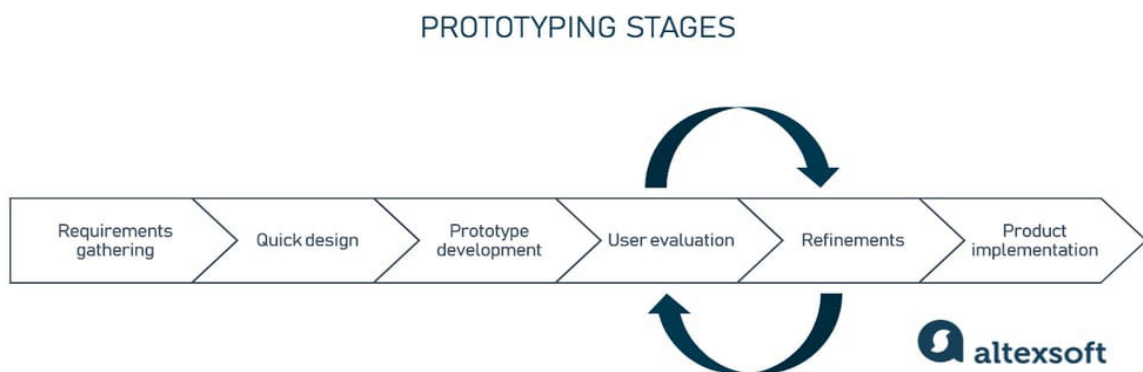


Figure 3.1 Prototype Stages

Source: altexsoft.com

3.1 Requirements gathering

The initial stage in the prototyping method is requirement gathering. At this phase, all system development needs are identified, including hardware components (ESP32, RFID module, relay, etc.), software elements (Android application, Firebase), and user interface design.

Data collection techniques included:

- a. Interviews with individuals from diverse backgrounds (students, workers, homemakers, etc.) to gather a wide range of perspectives regarding user expectations and needs.
- b. Observations of the surrounding environment to understand real-world usage contexts.
- c. Literature study to support technical understanding and provide references to similar studies [14]

3.1.1 Interviews

The interview respondents in this study consisted of five individuals with diverse backgrounds. Details of the respondents are presented in Table 3.1.

Table 3.1 Respondent table

Respondent	Job	Relevant background
R1	Technician and security service at Cozy Rooms Sleman	- Independently managed and implemented all security systems at Cozy Rooms, including smart door locks and solar-powered outlets.
R2	Staff at Informatics UII Sleman	- Responsible for the integration and deployment of smart door lock systems within the Informatics laboratories.
R3	Student at UII Sleman	- A student user who relies on a smart door lock as the primary access mechanism to her boarding residence on a daily basis.
R4	General Manager at Water Shop Dafaro Boyolali	- Had frequent exposure to smart door lock systems through regular use during hotel accommodations.
R5	Photographer	- Had frequent exposure to smart door lock systems through regular use during hotel accommodations.

The interviews for this study were conducted either in person or remotely with individuals who actively use, operate, or maintain locking systems that can be classified as smart door locks. The primary purpose of these interviews was to gather insights, user feedback, and personal experiences directly from those engaged with smart locking technologies.

In total, five respondents were successfully interviewed, each possessing relevant experience and/or knowledge related to smart door lock usage. The respondent profiles include:

(R1) a network and security technician responsible for managing a modern boarding facility located approximately one kilometer from UII, (R2) a staff at Informatics UII Sleman, (R3) a female student from the Informatics Department at Universitas Islam Indonesia (UII), (R4) a general manager at a gallon drinking water company Dafaro, and (R5) a photographer.

The first interview was conducted on Monday, 11 November 2024, with a faculty member from UII who has direct experience with the administration, installation, and maintenance of smart door lock systems in the Informatics laboratory building of the Faculty of Industrial Technology. The smart lock systems under this respondent's supervision include features such as RFID, fingerprint recognition, and password-based access. When asked about their impressions and any concerns, the respondent expressed general satisfaction, stating that the system has enhanced security within their working environment. When prompted to choose between feature-rich systems versus simpler yet reliable configurations, the respondent favored the latter—indicating a preference for minimal yet dependable functionality.

The second interview took place on Saturday, 3 May 2025, with a female student who uses a smart door lock system as the primary means of entry to her boarding room. The discussion focused on her experience, complaints, and suggestions for improving the smart lock system developed in this study. Regarding her experience with RFID technology specifically, the respondent expressed mixed feelings. Although she acknowledged the convenience of smart locks, she criticized the dependency on RFID cards, particularly when cards are forgotten or lost, requiring her to borrow a master RFID card from the building's security staff. Another concern was the inconsistent responsiveness of the RFID reader, which occasionally made access difficult. When asked about her preferred access method, she suggested a system that would allow her to unlock the door directly via a mobile phone application, indicating a need for smartphone integration.

The third interview was conducted on Monday, 5 May 2025, with a technician who also serves as the head of security at a modern boarding house near UII. The questions centered on his experience with maintaining the smart door lock systems in that facility and on user complaints received from residents. According to the respondent, the maintenance process is generally straightforward. However, residents have reported several issues. First, the system occasionally fails to scan RFID cards properly, leading to access delays. Second, lost RFID cards often result in user's room being locked out and needing to borrow a master card from the security staff. Third, the respondent mentioned the battery limitations of the smart lock

units. When battery levels are low, the locks tend to become unresponsive, sometimes requiring the use of physical backup keys or external power sources to regain access.

The final interview was conducted on Saturday, 10 May 2025, with the fourth and fifth respondents, who were interviewed simultaneously. Both individuals shared a similar background relevant to the study, namely frequent usage of smart door lock systems in hotel environments. When asked about their general impressions of the smart door locks they had used, both respondents expressed a high level of satisfaction. The type of system they most commonly encountered was a smart door lock integrated with RFID card access. Furthermore, when asked whether they had experienced any notable issues or complaints while using these systems, both respondents reported no significant problems to date.

The requirements obtained through the interviews include: the need to incorporate a remote access feature, integration with systems that are more accessible to end users, and a prioritization of reliability over an abundance of features.

3.1.2 Observations

The observation, that was conducted on October 2024 until June 2025, was employed by the author to examine events and phenomena occurring in the surrounding environment in order to collect insights from a visual, perception-based perspective. Through direct observation, the author identified a number of advantages and disadvantages associated with both conventional door security systems and smart door lock systems, as detailed in Table 3.2.

Table 3.2 Observed Available Locking System

Type of Locking System	Advantages	Disadvantages
Smart Door Lock	<ul style="list-style-type: none"> - Fast and convenient access - Keys are customizable (e.g., PIN, fingerprint, smartphone app) - Can be controlled and monitored remotely via mobile or web platforms - Supports multi-factor authentication for improved security - Provides access logs and real-time monitoring (on supported platforms) - Integration with smart home ecosystems and automation platforms 	<ul style="list-style-type: none"> - Higher initial installation cost - May experience failures due to battery depletion or software glitches - Dependent on stable power supply and/or internet connectivity - Requires technical knowledge for setup and maintenance - Some users report reduced reliability in extreme weather or signal interference
Conventional Lock	<ul style="list-style-type: none"> - Simple and easy to operate - Low cost and widely available - No reliance on electricity or network infrastructure 	<ul style="list-style-type: none"> - Uses fixed physical keys, which cannot be changed without replacing the entire lock - Physical keys are prone to being lost, misplaced, or duplicated without consent - Offers limited security—susceptible to lock picking and forced entry - No access monitoring or remote-control capability

The key currently used in daily door-opening scenarios emphasizes strong reliability and simplicity. In addition, fast response time is highly valued. Therefore, the requirements derived from observation are that the locking system must be simple and provide a quick response. Another requirement identified is the use of a non-traditional locking mechanism, avoiding

conventional systems such as the pin tumbler. This requirement eliminates the potential for breaches through relatively easy conventional methods.

3.1.3 Literature study

A literature study was conducted, around January 2025 until March 2025, to collect relevant information and support data for the development of the smart door lock system. The study drew from diverse sources, both printed and digital, including scholarly websites, open-access journals, and the institutional repository of Universitas Islam Indonesia (UII). This approach ensured that the proposed system design was grounded in existing research, established best practices, and recent technological advancements in smart access control systems.

From this literature study, several key requirements were identified. These include the development of a system that employs a standalone application independent of Blynk, as well as the implementation of an IoT-based architecture to enhance functionality and reliability.

3.2 Quick design

The next stage is quick design, which involves the rapid creation of an initial system design based on the gathered requirements. This preliminary design was developed in the form of

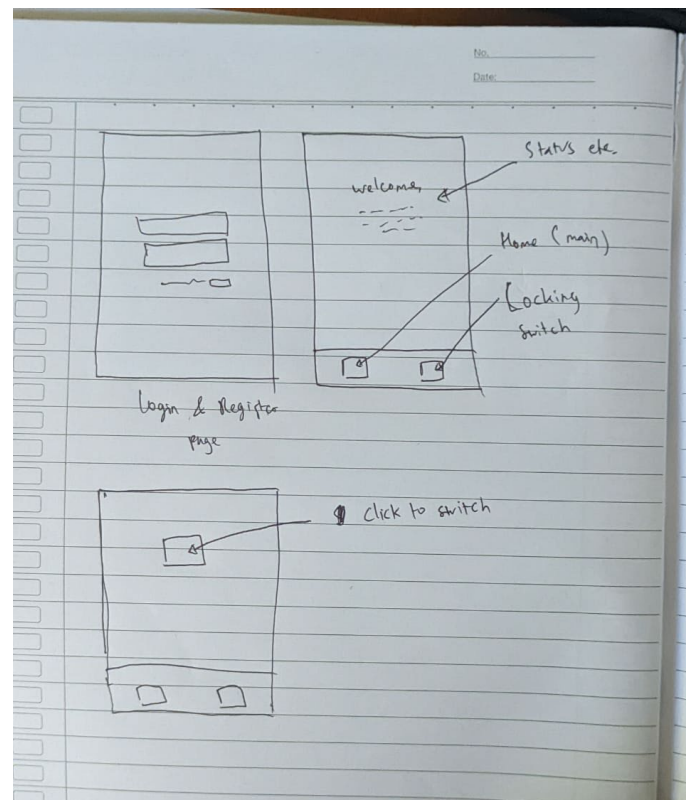


Figure 3.2 Quick Design on the software user interface

hand-drawn sketches. The focus at this stage was on the basic layout and flow of the system rather than technical implementation details.

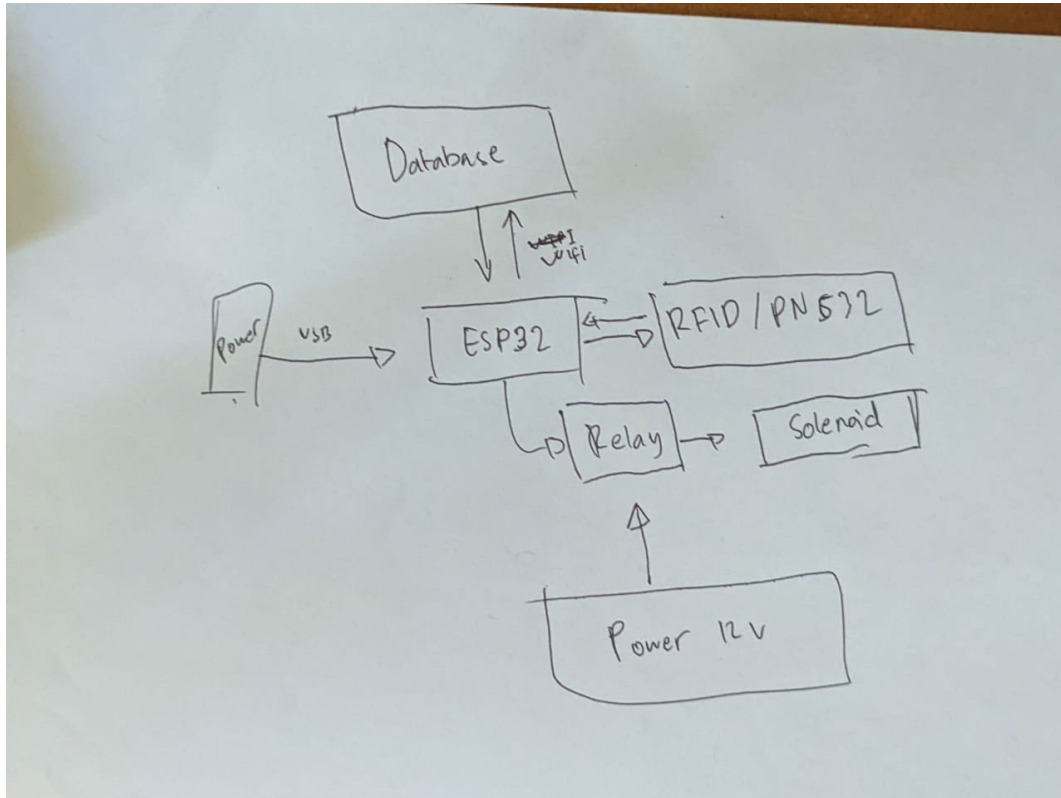


Figure 3.3 Quick Design on the hardware side

3.3 Prototype development

Following the quick design, the prototype development stage was conducted. The developed prototype included:

- Implementation of the ESP32 system for controlling the door lock and reading RFID UIDs.
- Integration of Firebase Realtime Database to facilitate communication between the mobile application and the hardware system.
- Development of the Android application using the React Native framework.

3.3.1 Implementation of the ESP32

The first stage of implementation based on the previously developed design involved the deployment of the hardware components for the smart door lock system. At the core of this implementation is the ESP32-WROOM microcontroller, which functions as the central processing unit responsible for managing sensor operations and facilitating communication

between the smart door hardware, the Android application, and the cloud database. This implementation phase encompassed the wiring of electrical connections as well as the programming of the ESP32-WROOM using the C/C++ programming language via the Arduino IDE development environment.

3.3.2 Integration of Firebase Realtime Database

The next stage involved preparing the database infrastructure to serve as the communication bridge between the smart door hardware and the Android application. This implementation included configuring the database structure, setting up essential parameters such as security rules, API key, authDomain, databaseURL, and projectId, among others.

During the development phase, the database rules were temporarily set to public access, shown in Figure 3.4. This configuration was intentionally applied to facilitate rapid prototyping and debugging, ensuring seamless data exchange during system testing. However, it is important to note that such a configuration is not recommended for production environments due to security concerns, and appropriate access restrictions should be applied in later stages of deployment. The database structure is shown in Figure 3.5.

```
{
  "rules": {
    ".read": true,
    ".write": true
  }
}
```

Figure 3.4 Database Rules are defined as public

```
{
  "authorizedRFIDs": {
    "[deviceId]": {
      "[rfidTag]": true
    }
  },
  "deviceHeartbeat": {
    "[deviceId]": {
      "freeHeap": Number,
      "ip": String,
      "lastSeen": String, // or Number (uptime in seconds)
      "rssi": Number,     // e.g. -42
      "state": Boolean,   // e.g. true or false
      "uptime": Number    // in seconds
    }
  }
}
```

```

},
"deviceMeta": {
  "[deviceName]": {
    "createdAt": String,    // ISO 8601 timestamp
    "createdBy": String,    // user UID
    "name": String,         // same as device name key
    "type": String          // e.g., "tv", "door", etc.
  }
},
"deviceStates": {
  "[deviceId]": {
    "isLocked": Boolean,    // true = locked
    "lastCommand": String,  // e.g., "manual_lock", "unlock"
    "timestamp": Number,    // UNIX timestamp in ms
    "triggeredBy": String   // user UID
  }
},
"rfidTimers": {
  "[deviceId]": {
    "[rfidTag]": Number     // fractional timer or duration in
minutes/hours
  }
},
"users": {
  "[userId]": {
    "devices": {
      "[deviceName]": {
        "createdAt": String, // ISO timestamp
        "createdBy": String, // UID of creator (usually same as user)
        "logs": [
          String             // human-readable log message
        ],
        "mode": String,      // e.g., "solid", "pulse", etc.
        "name": String,      // device name again
        "state": Boolean,    // e.g., true = on, false = off
        "type": String       // e.g., "tv", "esp", etc.
      }
    }
  }
}
}

```

Figure 3.5 Firebase Real-time Database JSON Structure

3.3.3 Development of Android application

Following the completion of hardware implementation for the smart door lock and the configuration of the real-time database, the next phase involved the development of the software component targeting the Android operating system. The Android application was developed using the TypeScript programming language in conjunction with the React Native

framework, which supports cross-platform mobile development. To streamline the development process, the Expo toolkit was utilized, offering an optimized development environment and simplified access to native device functionalities.

For styling the user interface, the application incorporated the Tailwind CSS utility-based library, allowing for consistent and responsive design throughout the Android application. The development environment employed for this stage was Visual Studio Code (VS Code), a widely used integrated development environment known for its compatibility with TypeScript and React Native.

The final deployment of the Android application was carried out using the EAS (Expo Application Services) build system, specifically with the command that shown in Figure 3.6, which compiles the project for Android devices under the specified build profile.

```
npx eas build -p android --profile preview
```

Figure 3.6 EAS Command

```
{
  "cli": {
    "version": ">= 16.13.2",
    "appVersionSource": "remote"
  },
  "build": {
    "development": {
      "developmentClient": true,
      "distribution": "internal"
    },
    "preview": {
      "distribution": "internal"
    },
    "production": {
      "autoIncrement": true
    }
  },
  "submit": {
    "production": {}
  }
}
```

Figure 3.7 eas.json

3.4 User evaluation and refinement

Once the prototype was developed, it was subjected to user evaluation. This evaluation aimed to gather feedback on usability, comfort, and alignment with user needs. Data were collected through direct surveys and informal user testing sessions.

If feedback revealed deficiencies or misalignments, improvements were made accordingly. This evaluation and refinement cycle was repeated iteratively until the prototype met user expectations and functional requirements.

This study involved three primary iterations throughout the development process.

3.4.1 User Evaluation and Refinement: First Iteration

During the first iteration, the system was developed with only basic features, such as remote switching functionality and a minimalist application interface. Based on feedback from stakeholders, requests were made to extend the system's capabilities by integrating additional technologies, specifically RFID and a PIN-based keypad module. Following an evaluation of available resources, it was decided that RFID integration would be implemented, while the PIN keypad feature was postponed due to material and hardware limitations faced by the author during this phase.

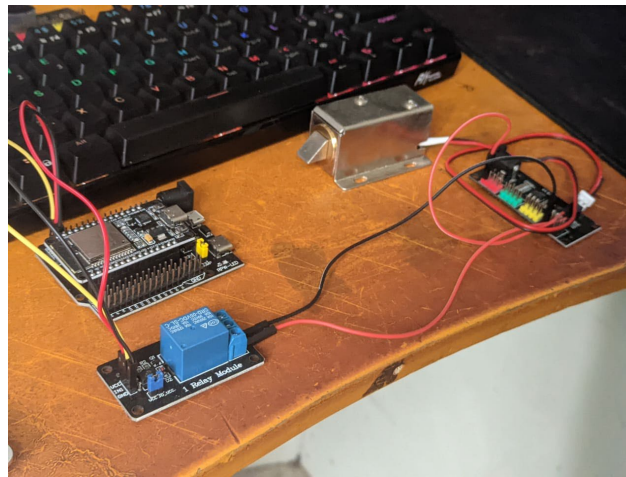
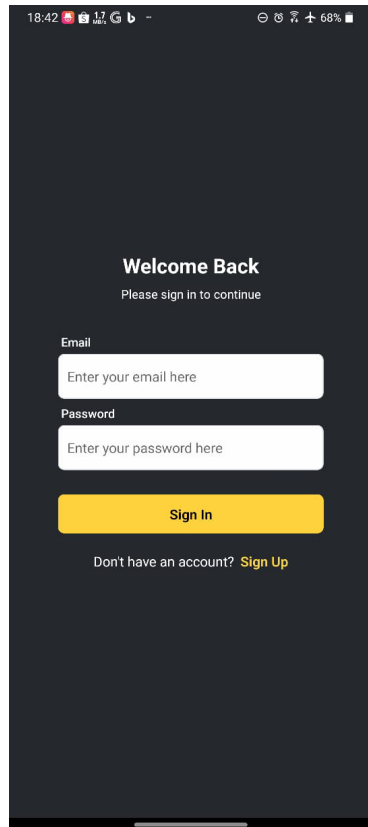
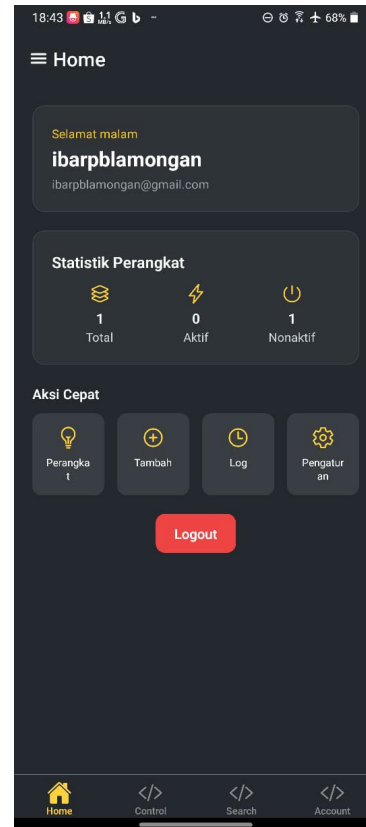
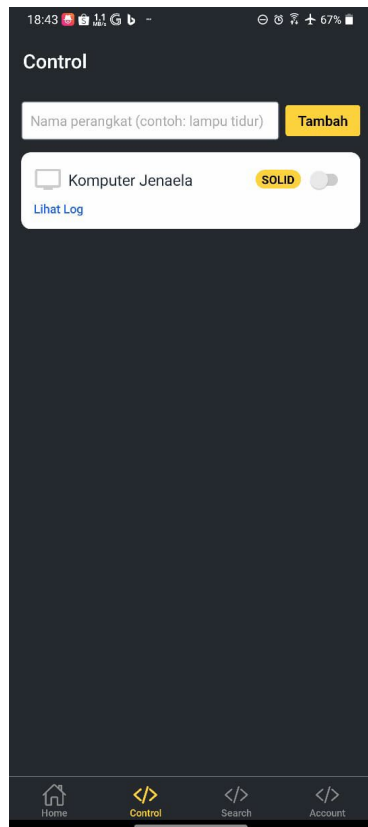
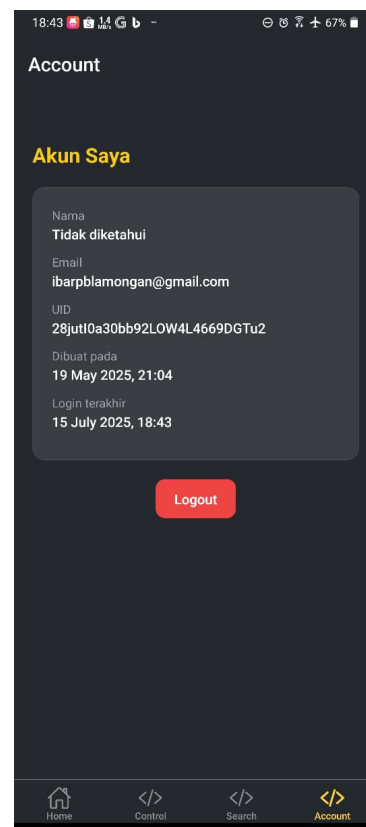


Figure 3.8 Smart Door Lock Hardware – First Iteration

At this stage, the primary features of the Android application included remote switch control over a wide area network (WAN) and a basic form of user authentication. These core functionalities provided the foundation for further feature expansion in subsequent development iterations.

Figure 3.10 Login Page (1st)Figure 3.9 Home Page (1st)Figure 3.12 Switch Control
Page (1st)Figure 3.11 Account Page
(1st)

3.4.2 User Evaluation and Refinement: Second Iteration

This second iteration involved a complete overhaul of the user interface (UI) and user experience (UX) of the Android application, aligning the design more closely with stakeholder feedback and usability expectations. In addition to visual and structural improvements, several new features were introduced to enhance functionality and meet stakeholder requirements. Furthermore, the database structure was reconfigured to improve data management, scalability, and system reliability.

Key features implemented during this iteration included: Tap to Unlock (RFID-based access control), Tap to Register (RFID enrolment mechanism), remote switch control via WAN, user authentication, RFID deletion functionality, and local ESP32 device discovery. These enhancements collectively contributed to a more robust and user-centric smart door lock application.

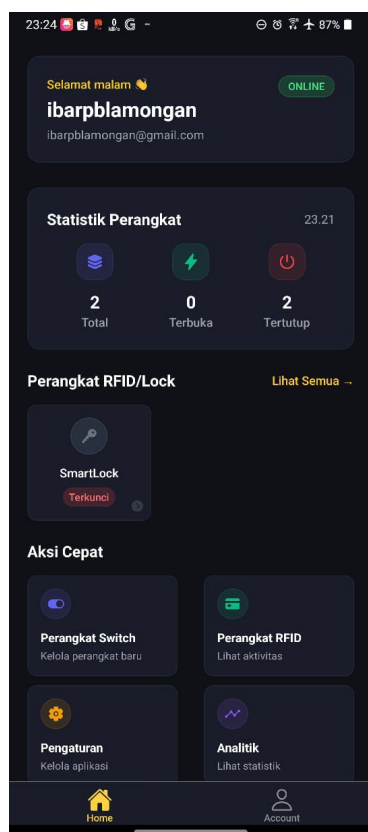
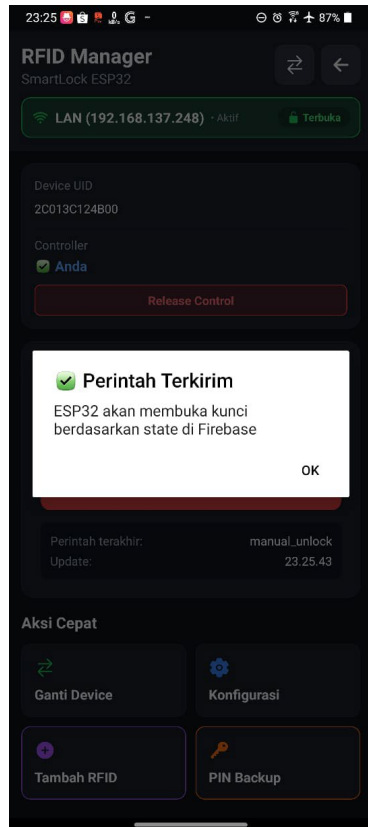
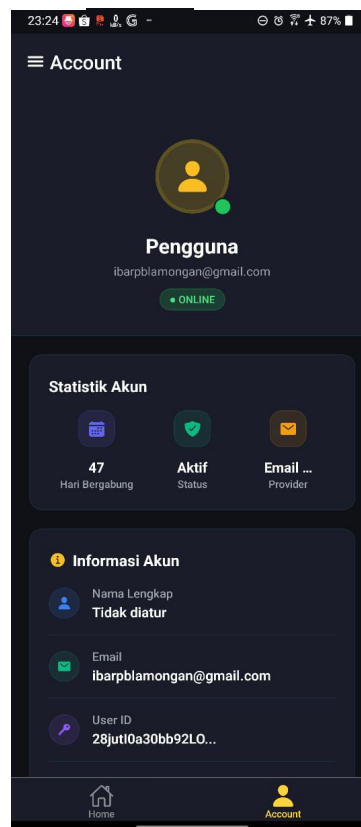


Figure 3.13 Home Page
(2nd)



Figure 3.14 RFID Manager
Page (2nd)

Figure 3.17 Command sent (2nd)Figure 3.15 WAN Mode (2nd)Figure 3.16 Account Page (2nd)

3.4.3 User Evaluation and Refinement: Third Iteration

The third iteration focused on refining the UI/UX design and optimizing the overall performance of the smart door lock system. Additionally, a new feature was requested by stakeholders: a customizable automatic locking timer, which would allow users to configure a delayed locking mechanism directly through the mobile application. While the feature was successfully developed and integrated into the software, it could not be fully deployed due to a hardware-related issue. Specifically, as a result of an error during the development process, the TX pin on the ESP32 microcontroller became damaged, preventing further firmware updates to the device. Consequently, although the feature was functionally complete, it could not be fully tested or embedded into the final smart door lock system.

At this stage, the core features of the application included: Tap to Unlock (RFID), Tap to Register (RFID), remote switching via WAN, user authentication, RFID tag deletion, ESP32 local device discovery, and the partially implemented custom automatic locking feature.

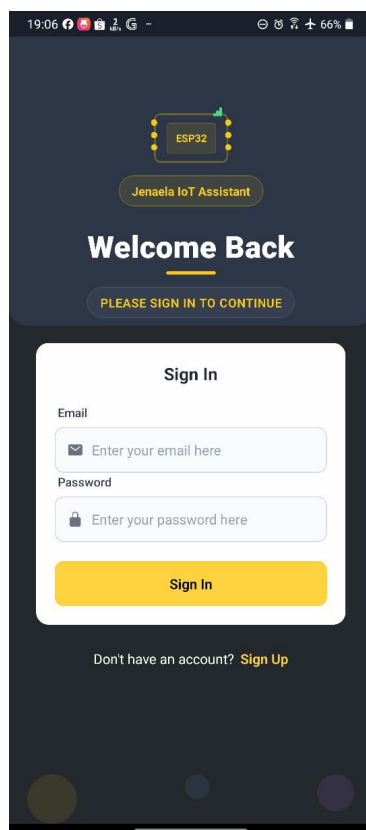


Figure 3.20 Login Page
(3rd)

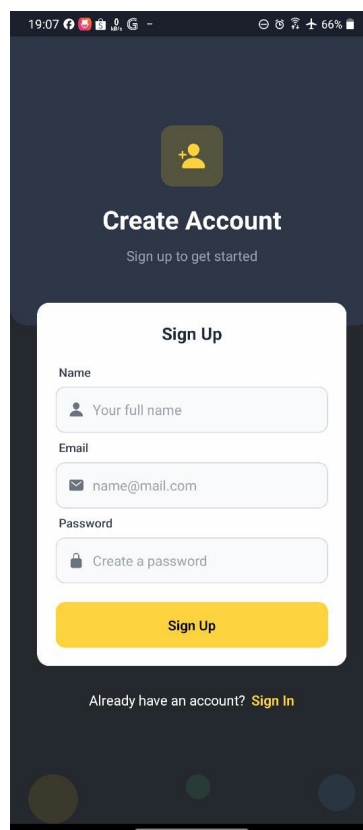


Figure 3.19 Register Page
(3rd)

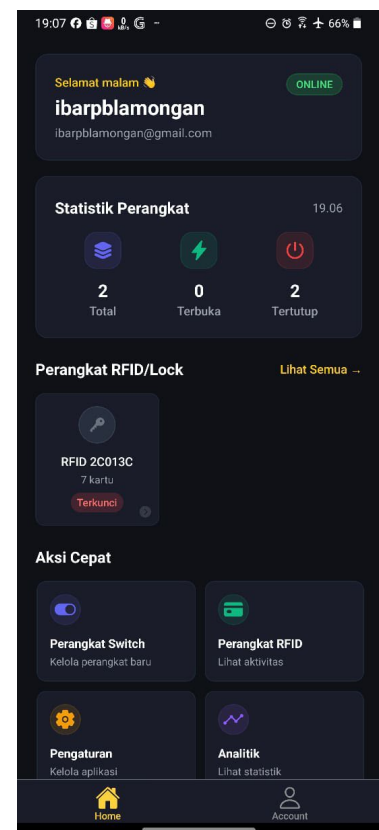


Figure 3.18 Home Page (3rd)

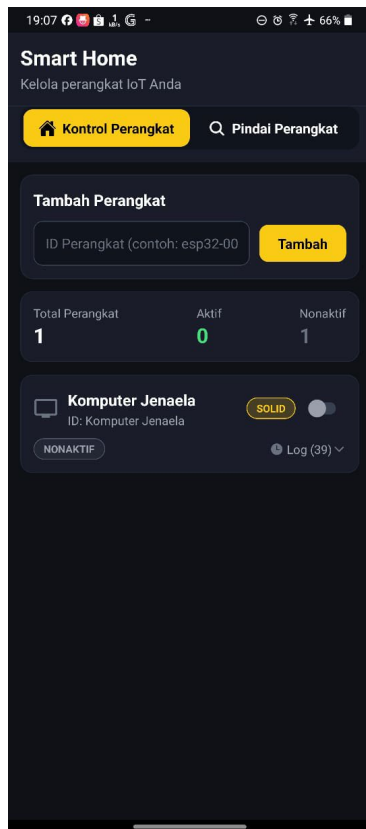


Figure 3.22 Manual
Controller Page (3rd)

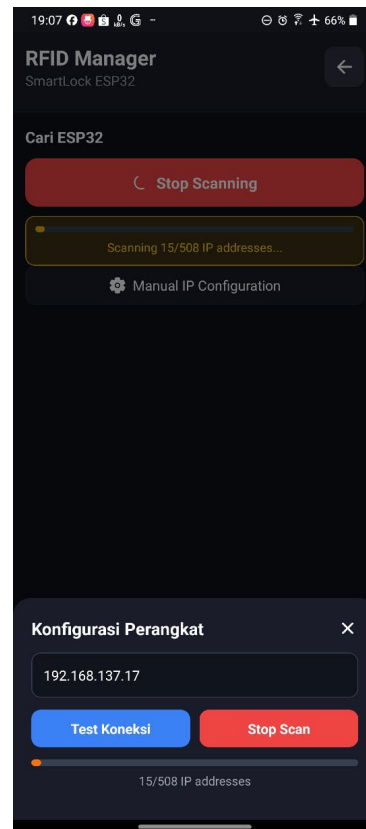


Figure 3.21 ESP32 Local Search
(3rd)

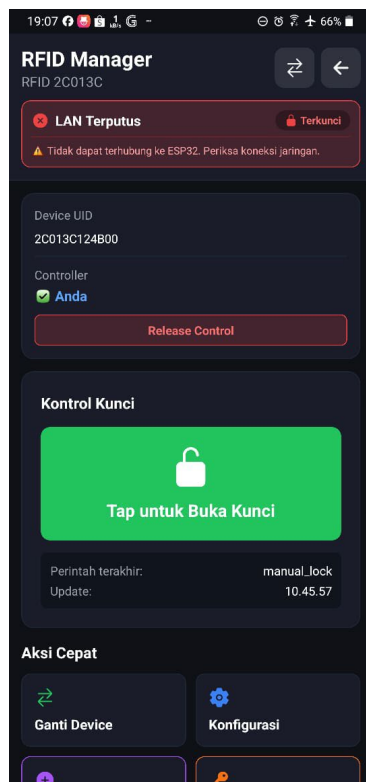


Figure 3.24 RFID Manager
Page (3rd)

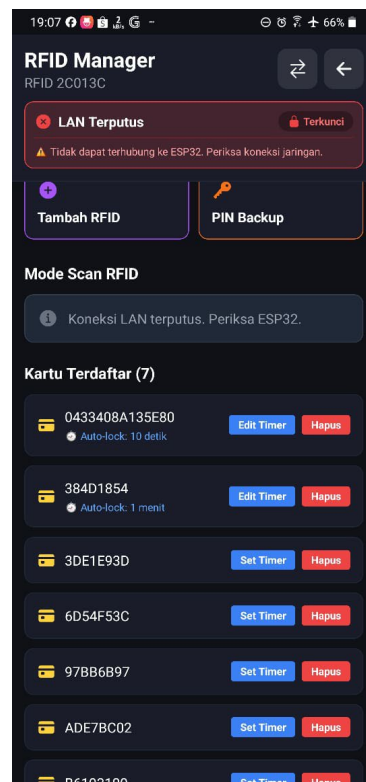


Figure 3.23 Custom
Automatic Lock Timer (3rd)



Figure 3.25 Smart Door Lock Miniature in the Box – Third Iteration

3.5 Product implementation

The final stage is product implementation, in which the refined system is deployed for use in a real-world scenario. This step ensures that all components—including hardware, software, and real-time database integration—function as intended.

CHAPTER IV

RESULTS AND DISCUSSION

The final stage is product implementation, in which the refined system is deployed for use in a real-world scenario. This step ensures that all components—including hardware, software, and real-time database integration—function as intended.

4.1 Hardware and Software Outputs

4.1.1 RFID Smart Door Lock

RFID was selected as the primary access method in this prototype for several reasons. First, interviews with stakeholders indicated a preference for RFID-based access, as it is widely used in daily environments such as boarding houses and hotels. Second, the project was constrained by limited budget and time, requiring the use of readily available hardware components; therefore, the RFID module was prioritized over a keypad module. Finally, during development, the ESP32 device used in this project experienced hardware failure, preventing further modification of the existing firmware. Although the Android application was designed with PIN-based features in mind for future iterations, the current prototype focuses on RFID implementation as the most feasible solution under these conditions.

As a result, the first outcome of this system development was a functional miniature prototype of a smart door lock, constructed using the components listed in Table 4.1.

Table 4.1 Hardware Components

No.	Component Name	Details
1	ESP32 WROOM DevKit DOIT V1	<ul style="list-style-type: none"> - ESP32 WROOM 32 (Tensilica Xtensa LX6, 32-bit dual-core CPU, up to 240 MHz clockspeed, TSMC 40nm CMOS Process Node) - 512 KB RAM (448 KB usable for code)

		<ul style="list-style-type: none"> - 4 MB ROM - Classic and BLE Bluetooth support - WiFi IEEE 802.11 b/g/n (WiFi 4) - USB Type C interface - 3.3V
2	PN532 Moudule V3 Elechouse	<ul style="list-style-type: none"> - NXP PN532 (Node fabrication is not specified, 13.56 MHz operating frequency) - MIFARE cards support - ISO/IEC Type A and B - Supports SPI, I2C, and UART communication - Reading ranges up to 5 cm (tag and antenna dependant) - 3..3V/5.5V DC operation
3	ESP32 WROOM DevKit DOIT V1 Expansion Board	<ul style="list-style-type: none"> - 3.3V/5V DC Switch
4	Micro DC to DC Step Up	<ul style="list-style-type: none"> - 2.5V – 5V DC input - 5V/8V/9V/12V output

5	Relay	- Optocoupler integrated
6	Solenoid Lock	- 12V DC operation
7	Jumper Cables	- Copper (Further details like N value and etc. are not specified)

The ESP32-WROOM microcontroller was selected for this system due to its status as the highest-performing device available to the author at the time of development. To ensure stable connections and reduce signal interference issues commonly encountered with breadboards, an expansion board was used as a more reliable alternative. The PN532 Elechouse V3 NFC module was chosen based on its widespread availability in the Indonesian market, making it a practical and accessible component for local prototyping.

All components were assembled in accordance with the system block diagram presented in Figure 4.1, ensuring consistency between the design specifications and the physical implementation.

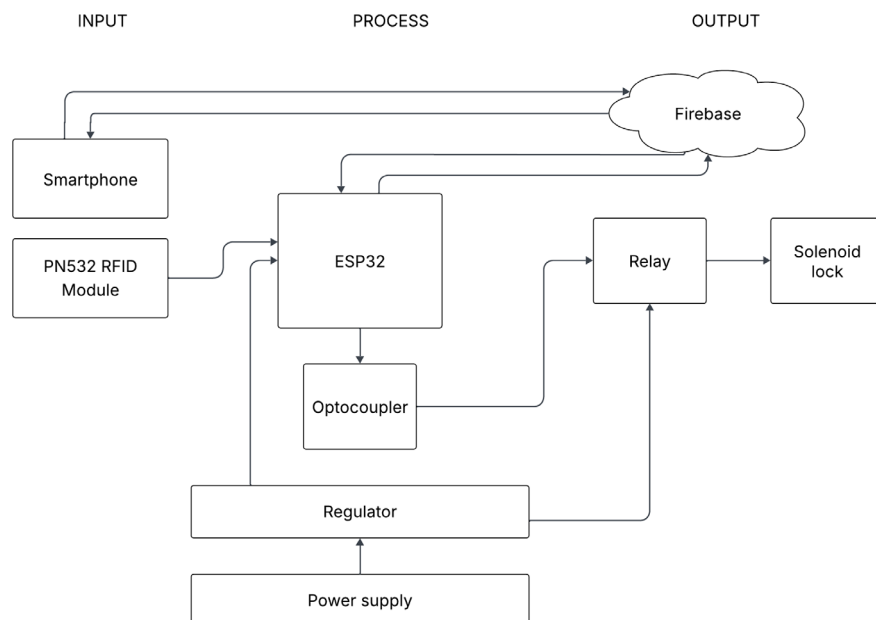


Figure 4.1 Smart Door Lock Block Diagram

Furthermore, the projection of the component block diagram for the miniature smart door lock system, as illustrated in Figure 4.1, is visually represented in its physical assembly in Figure 4.2.

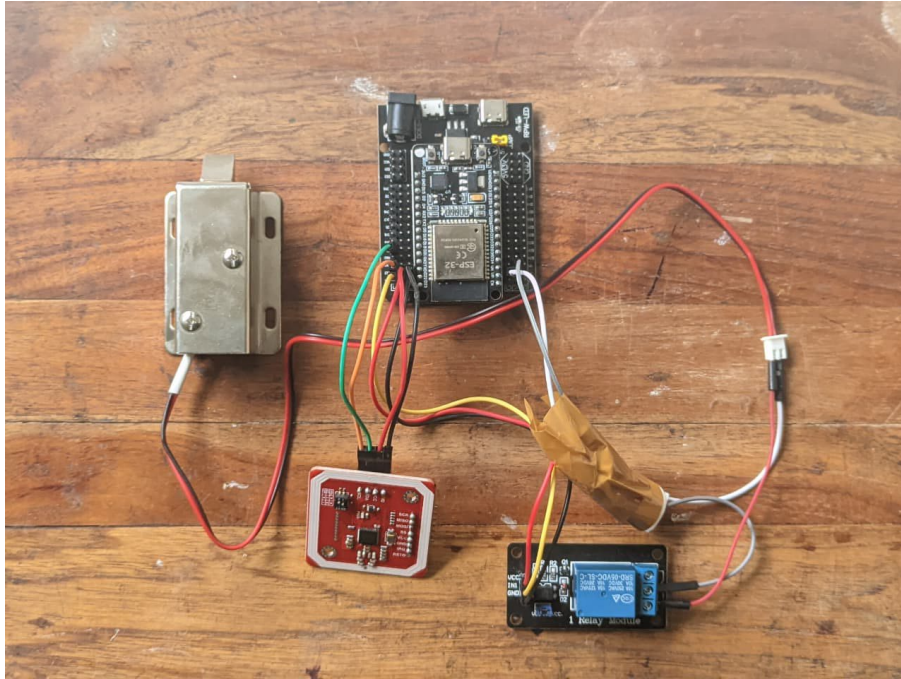


Figure 4.2 Final Smart Door Lock Hardware Miniature

The initial plan to utilize a voltage regulator was cancelled and replaced with a USB Type-C connection directly linked to a laptop or computer. This decision was made because the available voltage regulator did not meet the necessary specifications required for the miniature prototype designed by the author.

During the construction and testing of the prototype, the author encountered several findings, particularly concerning the voltage sensitivity of the ESP32 module. In one experiment, the author attempted to power the circuit using a 5V DC charger with a 2.5 mm barrel jack. Upon initial startup, the serial monitor displayed normal operation without any apparent issues. However, when the relay was triggered—which in turn activated the solenoid lock—the ESP32 experienced a critical malfunction. This was indicated by abnormal output on the serial monitor, as shown in Figure 4.3.

```

❑!Cleaned      data:      '{"0433408A135E80":uque-!195E1955!9uque-
!1EE1E91E!9true,"6D54F53C":true,"97BB6B97":usue-#AEE7CC13#;usue-
#C7193111#;true}'
☑ JSON parsed successfummy
❑!Qqmaeqqime!QEIE!emuqies:

```

```

📄 UID: 0433408A135E80 -!穀!Acuiwe

!!☐!UIE;!395E1954 - ☒ Active
📄 UID: 3DE1E91E!-!ヲ!Aauiue
!!☐!UIE9!6D54F53C - ☒ Active
📄 UIE;!97CC7C97!-!穀!Acuiwe

!!☐ UID: ADE7BC02 - ☒ Active
V!UIE9!A5191111!-!ヲ!Aauiue

📊 RFID sync summary:
Previous!couou;!7

!!Oew!couou;!7

!!Status: ☒ Success
☒ Auto-syma!ammqmeuee!quaaeqqeummy
E (120687) i2c.master: I2C hardwase!OACK!eeuecuee

E!)131797)!i3c.master: I2C transaction unexpeauee!maai!eeueauee
E!)111595)!i2c.master:
s_i2c_synchronous_traosacuioo)935);!I3C!usaosacuioo!gailed
E (120693) i2c.master: i2a]maqueq]qaeiue)1151)9!I1A!uqamsaction
failed
E!)133597)!i3c/masues;!I3C!iaseware NACK detected
E (122586) i2a-maqueq9!I1A!uqamqaauimm!umeyqected nack detected
E (122586) i3c/masues;!s_i3c_syocisooous_usansaction(924): I2C
transaction faimee
E!)111591)!ila-maqueq9!ilc_master_receive(1240): I2C
transacuioo!gaimee

E (124485) i2c.master: I2C hardwaqe!MAAI!eeueauee
E!)115595)!ilc.master: I2C transaction unexpecuee!oack!eeuecuee

E!)135595)!i2c.master:
s_i2c_synchronous_tramqaauimm)915)9!I1A!uqamqaauimm!eailed
E (124491) i2c.master: i2c_masues_seceiwe)1351);!I3C!usaosaction
failed
☐!Qeqemqmime!auum-qyma!)5!qeaond interval)...
☐!Auuo-syoc!SGIE!uqaeue!)5!second interval)
🔍 Raw data reaeiuee9!%y!1511519A115E91!9uque-
"384D1854":true,"3DE1E93D":true,#7E55G53C#;usue-#97CC7C97#;usue-
"ADE7BC02":true,"B6193100":true}%
☐!Eaua!memeui9!119
☐!Aleaned data: '{"0433408A135E80":usue-#395E1955#;usue-
#3EE1E93E#;true,"6D54F53C":true,"97BB6B97":uque-!AEE5AA11!9uque-
!A5191111!9true}'
☒ JSON parsed successfummy

☐!Qsocessiog!SGIE!eousies:

📄 UID: 0433408A135E80 -!ヲ!Aauiue

!!☐!UIE9!195E1954 - ☒ Active
📄 UID: 3DE1E93E!-!穀!Acuiwe

```

```

!!☐!UIE;!6D54F53C - ☑ Active
  📄 UIE9!95AA5A95!-!7!Aauieue
!!☐ UID: ADE7BC02 - ☑ Active
  📄 !UIE;!C7193111!-!穀!Acuiwe

📶 RFID sync summary:
  Previmuq!amumu9!5
!!Meu!amumu9!5
!!Status: ☑ Success
☑ Auto-syoc!comqmeuee!successgummy

E!)136391) i2c.master: I2C hardware NAAI!eeueauee
E!)115191)!ila-maqter: I2C transaction unexpected oack!eeuecuae

E!)137393)!i3c/master:
s_i2c_synchronous_transactimm)915)9!I1A!uqamqaauimm!eaimee
E (126400) i2c.master: i2c_masues_seceiwe)1351);!I3C!usaosacuion
failed
E!)119191)!ila-maqueq9!I1A!iaqueuare NACK detected
E (128293) i2c/masues;!I3C!usaosacuioo!uoeyqected nack detected
E (128293) ila-maqueq9!q]ila]qymaiqmmmuq]uqansaction(924): I2C
transaction faimee

E!)139399)!i3c/masues;!i3c_master_receive(1240): I2C
tranqaauimm!eaimee
📶 Real-time RFID update receiwee

☐!Saw!eaua!seceiwee;!'
event: keep-alive
data: null

```

Figure 4.3 Abnormality in serial monitor

It is suspected that the power supply was unable to maintain a stable 5V output, especially during the momentary current surge when the solenoid lock was engaged. This likely caused the ESP32 to receive voltage outside its acceptable operating range, leading to electrical stress on the microcontroller's components.

As a result of this incident, the ESP32 module suffered what is believed to be a permanent malfunction, specifically affecting the TX (transmit) pin, which exhibited abnormal behavior and rendered firmware updates via USB interface non-functional. This issue persisted even after resets and reprogramming attempts, suggesting hardware-level damage.

Consequently, the inability to update the ESP32's firmware halted further development and iteration, particularly the integration of additional features intended for the final version of the smart door lock prototype.

4.1.2 Jenaela Android Application

The software output of this study is an Android application named “Jenaela.” The application was developed using the TypeScript programming language and the React Native framework, integrated with the Expo build environment, and deployed using EAS Build Preview. The application includes several key features, such as:

a. Switch via WAN

One of the core features of the developed application is the Switch via WAN functionality. This feature enables remote control of devices over a wide area network, allowing it to be applied across various usage scenarios. Several scenarios were successfully tested by the author, demonstrating the feature's reliability and effectiveness. For instance, the application was able to remotely power on a computer and toggle the open/close state of the miniature smart door lock system developed in this study.

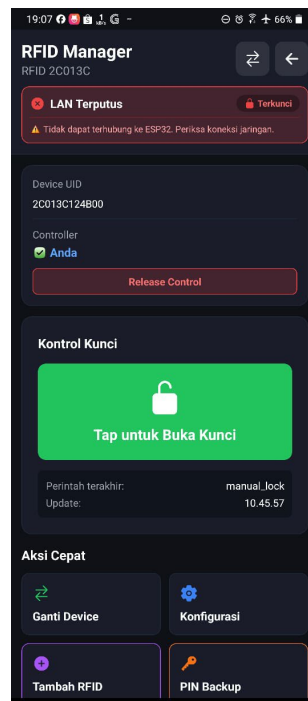


Figure 4.4 Tap untuk Buka

```

<Pressable
  onPress={toggleLockState}
  className={`p-6 rounded-xl ${!canControlDevice()
    ? 'bg-gray-700'
    : isUpdatingState
    ? 'bg-gray-700'
    : deviceState.isLocked
    ? 'bg-green-500'
    : 'bg-red-500'
  }}
  disabled={!canControlDevice() || isUpdatingState}
>
  <View className="items-center">
    {isUpdatingState ? (
      <ActivityIndicator color="white" size="large" />
    ) : (
      <Ionicons
        name={deviceState.isLocked ? "lock-open" : "lock-closed"}
        size={48}
        color="white"
      />
    )}
    <Text className="text-white font-bold text-xl mt-3">
      {!canControlDevice()
        ? 'Terkunci oleh pengguna lain'
        : isUpdatingState
        ? 'Mengirim perintah...'
        : deviceState.isLocked
        ? 'Tap untuk Buka Kunci'
        : 'Tap untuk Kunci'}
    </Text>
  </View>
</Pressable>

```

Figure 4.5 Button for Switch via WAN

```

const toggleLockState = async () => {
  if (!actualDeviceUid) {
    Alert.alert("✖ Error", "Perangkat belum terhubung");
    return;
  }

  if (!canControlDevice()) {
    Alert.alert(
      "✖ Akses Ditolak",
      `Perangkat ini sedang dikontrol oleh pengguna lain.\n\nController: ${deviceState.triggeredBy || 'Unknown'}`
    );
    return;
  }
}

```

```

    if (isUpdatingState) {
        return; // Prevent multiple simultaneous commands
    }

    setIsUpdatingState(true);

    try {
        // Claim device if it's not claimed yet
        await claimDeviceIfAvailable();

        const newLockState = !deviceState.isLocked;
        const newState = {
            isLocked: newLockState,
            lastCommand: newLockState ? "manual_lock" :
"manual_unlock",
            timestamp: Date.now(),
            triggeredBy: user?.uid || "unknown"
        };

        // Update Firebase state - ESP32 will listen to this
        await set(ref(db, deviceStatePath), newState);

        Alert.alert(
            "☑ Perintah Terkirim",
            `ESP32 akan ${newLockState ? 'mengunci' : 'membuka
kunci'} berdasarkan state di Firebase`
        );

    } catch (error) {
        Alert.alert("✗ Error", "Gagal mengirim perintah ke
Firebase");
        console.error("Firebase state update error:", error);
    } finally {
        if (isMounted.current) {
            setIsUpdatingState(false);
        }
    }
};

```

Figure 4.6 toggleLockState function

When the green button shown in Figure 4.4 is pressed, using the program logic displayed in Figure 4.5, the function `toggleLockState` (as defined in Figure 4.6) is triggered. This function initiates a command to update the boolean value (true/false) stored in the cloud database, representing the lock state. The updated value is then read by the ESP32 module, which responds accordingly by actuating the locking mechanism.

During the update process, the application displays a loading spinner animation to indicate that a state change is in progress. Once the update is complete and the system

is no longer in the updating state, the interface dynamically renders an icon based on the current lock state: a green icon if the state is true (indicating an unlocked or active state), and a red icon if the state is false (indicating a locked or inactive state).

b. User Authentication

In addition to the Switch via WAN feature, the application also incorporates a user authentication mechanism. This feature serves to restrict access, ensuring that only authorized users are permitted to interact with and control the connected smart door lock system. It acts as a security layer to prevent unauthorized individuals from accessing or modifying device functions through the application.

c. Tap to Unlock

Another key feature implemented in this system is RFID card integration. This feature includes several functionalities, such as RFID card registration, smart door lock access using registered cards, and the configuration of a custom delay timer that determines how long the system waits before automatically relocking the door. The implementation of this feature is directly tied to the miniature smart door lock prototype developed as part of this research, enabling secure and user-friendly physical access control.

As shown in Figure 4.7, the `readRFIDCard` function is responsible for reading the UID (Unique Identifier) from the scanned RFID card and converting it into a string format. Following this, the system initiates the `isRFIDAuthorized` function, illustrated in Figure 4.8. This function performs a string comparison between the scanned UID and the list of pre-registered RFID card identifiers stored in the system. If a match is found—indicating that the card is authorized—the relay is activated, allowing the smart door lock to open.

```
void readRFIDCard() {
// ...existing code...
uint8_t uid[] = {0,0,0,0,0,0,0}, uidLength;
// Try to read card
bool cardDetected = false;
try {
    cardDetected = nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A,
uid, &uidLength, 50);
} catch (...) {
    cardDetected = false;
}
// ...existing code...
String rfidUID = "";
```

```

for (uint8_t i = 0; i < uidLength; i++) {
    if (uid[i] < 0x10) rfidUID += "0";
    rfidUID += String(uid[i], HEX);
}
rfidUID.toUpperCase();
// ...existing code...
bool granted = isRFIDAuthorized(rfidUID);
if (granted) {
    Serial.println("☑ Access Granted for: " + rfidUID);
    activateRelay();
} else {
    Serial.println("☒ Access Denied for: " + rfidUID);
}
// ...existing code...
}

```

Figure 4.7 readRFIDCard function

```

bool isRFIDAuthorized(String rfidUID) {
    for (int i = 0; i < rfidCount; i++) {
        if (authorizedRFIDs[i].uid == rfidUID &&
authorizedRFIDs[i].authorized) return true;
    }
    return false;
}

```

Figure 4.8 isRFIDAuthorized function

d. Register RFID

Another key feature of the system is the RFID registration function, which allows users to register their personal RFID tags and store the corresponding data in the Firebase Realtime Database. This functionality plays a critical role in enabling personalized access control within the smart door lock system.

The application provides two methods for registering RFID tags. The first method involves scanning the RFID tag by tapping it on the smart lock device, which has its scan mode activated. The second method allows for manual registration by directly entering the RFID tag code into the application. As shown in Figure 4.9, the `saveRFIDToFirebase` function represents the first method. This function is triggered when the user presses the save button on the RFID Manager page. Upon activation, the function stores the scanned RFID UID into the database after completing necessary validation steps.

The second method, manual input, is handled by the addManualUID function, which is illustrated in Figure 4.10. This function follows a similar logic structure, allowing users to add an RFID UID string directly to the database without requiring physical scanning. Together, these two approaches provide a flexible and user-friendly mechanism for managing authorized access through RFID authentication.

```
const saveRFIDtoFirebase = async (rfidUID: string) => {
  // ...validasi...
  await set(ref(db, `${uidPath}/${rfidUID}`), true);
  // ...notifikasi sukses/gagal...
}
```

Figure 4.9 saveRFIDtoFirebase function

```
const addManualUID = async () => {
  // ...validasi...
  await set(ref(db, `${uidPath}/${cleanUID}`), true);
  // ...notifikasi sukses/gagal...
}
```

Figure 4.10 addManualUID function

e. ESP32 Local Discovery

This feature searches for the available ESP32 within the local network using IP /discover.

```
const scanForESP32 = async () => {
  // Membuat daftar IP yang akan discan
  const networkRanges = ['192.168.137', '192.168.1'];
  const priorityIPs = [185, 17, 1, 100, ...];
  const allIPs = networkRanges.flatMap(baseIP =>
    [...priorityIPs, ...Array.from({ length: 254 }, (_, i) => i + 1)].map(lastOctet => `${baseIP}.${lastOctet}`)
  );
  // Scan IP satu per satu, cari ESP32 dengan endpoint /discover
  for (let ip of allIPs) {
    const response = await fetch(`http://${ip}/discover`);
    if (response.ok) {
      const data = await response.json();
      if (data.type === "rfid_relay" && data.uid) {
        setDeviceIP(ip);
      }
    }
  }
}
```

```

        setActualDeviceUid(data.uid);
        setConnectionStatus('connected');
        break;
    }
}
};

```

Figure 4.11 scanForESP32 function

f. RFID Tag Deletion

```

const deleteUID = async (uid: string) => {
// filepath: c:\a backup\a\app\app\drawer\tabs\rfidManager.tsx
  if (!canControlDevice()) {
    Alert.alert("✗ Akses Ditolak", "Perangkat ini sedang dikontrol oleh pengguna lain");
    return;
  }

  Alert.alert("Konfirmasi", `Hapus UID ${uid}?`, [
    { text: "Batal", style: "cancel" },
    {
      text: "Hapus",
      style: "destructive",
      onPress: async () => {
        try {
          await remove(ref(db, `${uidPath}/${uid}`));
          Alert.alert("☑ Berhasil", "UID berhasil dihapus");
        } catch (error) {
          Alert.alert("✗ Error", "Gagal menghapus UID");
          console.error('Error deleting UID:', error);
        }
      },
    },
  ],);
};

```

Figure 4.12 deleteUID function

g. Custom Automatic to Lock Timer

This feature is designed to modify the auto-lock timer configuration, which determines the delay before the smart door lock re-engages after being unlocked using an RFID tag. When the user presses the “Set Timer” or “Edit Timer” button, a modal interface is displayed to allow timer adjustment. This modal is illustrated in Figure 4.13.

Once the user confirms the new timer value by pressing the “Save Timer” button within the modal (as shown in Figure 4.13), the system triggers the function depicted in Figure 4.14. This function is responsible for storing the updated timer configuration, ensuring that the smart door lock applies the specified delay interval for automatic re-locking following RFID-based access.

```
const openTimerModal = (uid: string) => {
  if (!canControlDevice()) {
    Alert.alert("✖ Akses Ditolak", "Perangkat ini sedang dikontrol oleh pengguna lain");
    return;
  }

  setSelectedRFIDForTimer(uid);
  // Pre-fill dengan timer yang sudah ada jika ada
  const existingTimer = rfidTimers[uid];
  setTimerMinutes(existingTimer ? existingTimer.toString() : "");
  setShowTimerModal(true);
};
```

Figure 4.13 openTimerModal modal

```
const saveRFIDTimer = async () => {
  if (!selectedRFIDForTimer || !actualDeviceUid) {
    Alert.alert("✖ Error", "RFID atau perangkat tidak valid");
    return;
  }

  if (!canControlDevice()) {
    Alert.alert("✖ Akses Ditolak", "Perangkat ini sedang dikontrol oleh pengguna lain");
    return;
  }

  const minutes = parseFloat(timerMinutes.trim());

  if (isNaN(minutes) || minutes < 0) {
    Alert.alert("✖ Error", "Masukkan waktu yang valid");
    return;
  }
}
```

```

    }

    if (minutes > 1440) { // Maksimal 24 jam
        Alert.alert("✗ Error", "Maksimal 1440 menit (24 jam)");
        return;
    }

    try {
        if (minutes === 0) {
            // Hapus timer jika 0
            await remove(ref(db,
`${rfidTimersPath}/${selectedRFIDForTimer}`));
            Alert.alert("☑ Berhasil", "Timer dihapus - pintu akan tetap
terbuka");
        } else {
            // Simpan timer dalam satuan menit (boleh desimal untuk detik)
            await set(ref(db,
`${rfidTimersPath}/${selectedRFIDForTimer}`), minutes);

            const displayTime = minutes < 1
                ? `${Math.round(minutes * 60)} detik`
                : formatTimer(minutes);

            Alert.alert("☑ Berhasil", `Timer diset ${displayTime} untuk
RFID ${selectedRFIDForTimer}`);
        }

        setShowTimerModal(false);
        setSelectedRFIDForTimer("");
        setTimerMinutes("");
    } catch (error) {
        Alert.alert("✗ Error", "Gagal menyimpan timer");
        console.error('Error saving RFID timer:', error);
    }
};

```

Figure 4.14 saveRFIDTimer function

h. History Log

During the early stages of development, a history log feature was implemented in the first prototype iteration to record access events. The functionality was intended to be refined and reintroduced in the fourth iteration with improved data handling. However, due to hardware failure in the ESP32 device, further code modifications could not be deployed, and as a result the history log was not included in the final version of the prototype. Despite this limitation, the Android application already provides the necessary interface and database structure to support event logging.

4.2 Testing

System testing was conducted using the Black Box testing method. This approach focuses on evaluating the functional aspects of the system without requiring knowledge of the underlying code or implementation details (Utomo et al., 2020).

4.2.1 RFID Smart Door Lock Testing

The performance testing results of the hardware locking mechanism are presented in Table 4.2 and Table 4.3. This test aimed to evaluate the snappiness and reliability of the developed device. The testing was conducted using a digital stopwatch and a ruler to measure response times and RFID detection distances.

The relative distance between the PN532 sensor surface and the RFID tag surface was measured, with distance intervals ranging from 1 cm to 5 cm. For each distance, the test was repeated three times by placing the RFID tag at the specified distance and recording the response time using the stopwatch.

Table 4.2 B6193100 card type

Test	Locking Mode	Code	Physical Form	Relative Distance	Average Response Time	Result Description
1	RFID	B6193100	Card	1 cm	~20 ms	Working
2				2 cm	~20 ms	Working
3				3 cm	~20 ms	Working
4				4 cm	~30 ms	Working
5				5 cm	~30 ms	Working

Table 4.3 ADE7BC02 card type

Test	Locking Mode	Code	Physical Form	Relative Distance	Average Response Time	Result Description
1	RFID	ADE7BC02	Key chain	1 cm	~20 ms	Working
2				2 cm	~20 ms	Working
3				3 cm	~30 ms	Working
4				4 cm	-	Not working
5				5 cm	-	Not working

Although the RFID-based unlocking method yielded positive test results—with the worst-case average response time remaining within approximately 30 milliseconds—a different

outcome was observed during the evaluation of the Switch via WAN unlocking method. The performance results of the Switch via WAN feature are presented in Table 4.4.

Table 4.4 Switch via WAN

Test	Locking Mode	Lock Response Time	Result Description
1	Switch through WAN	~1800 ms	Working
2		~800 ms	Working
3		~1000 ms	Working
4		~300 ms	Working
5		~600 ms	Working

4.2.2 Jenacla App Testing

The Jenacla application was tested through several scenarios considered representative of real-world use cases in managing IoT hardware such as smart door locks. These scenarios were designed to reflect key functionalities of the Android application. The testing scenarios and their expected outcomes are presented in Table 4.5. The Result Description is marked as “Working” if the actual outcome matches the expected result.

Table 4.5 Jenacla scenario-based testing

No	Testing Scenario	Expected Outcome	Result Description
1	User opens the Sign-Up page, fills in personal details (email, password), and clicks the Sign-Up button.	The user is successfully registered in the database and is able to log in to the application.	Working
2	User opens the Sign-In page, enters email and password, and clicks the Sign-In button.	If the credentials are correct and the user is registered, the application navigates to the home page.	Working
3	User scans for available IoT devices using the ESP32 Scan feature on the RFID Manager page.	The application displays a list of available IoT devices.	Working
4	User opens the saved device list and selects a device from the RFID Manager page.	A pop-up displays the saved device information. The device can then be	Working

		selected, redirecting the user to the RFID Manager page.	
5	User clicks the "Tap untuk Buka Kunci/Tutup Kunci" button on the RFID Manager page.	The application updates the <isLocked> value in the Firebase Realtime Database to reflect the current lock state.	Working
6	User clicks the "Aktifkan Mode Scan" button to register an ID tag on the RFID Manager page.	A card UI appears and displays the scanned ID tag code when the user taps the tag on the PN532 sensor.	Working
7	User clicks the "Hapus" button next to a registered ID tag in the list.	The selected ID tag is removed from the database, and the list is refreshed to reflect the updated data.	Working
8	Jenaela automatically scans for available IoT devices within the local area network (LAN) upon application launch.	The application detects IoT devices present within the local area network (LAN).	Working

CHAPTER V

CONCLUSION

This study focused on developing a smart door lock system that integrates IoT-based hardware with a custom-built Android application. The objective was to design a secure, flexible, and user-friendly solution that addresses the limitations of traditional mechanical locks as well as third-party platforms like Blynk.

The system implementation, named Jenaela, employed a standalone Android application developed with React Native and real-time communication via Firebase. This approach demonstrated the feasibility and effectiveness of achieving full customization and control. The hardware integration—including the ESP32 DOIT V1, PN532 RFID module, relay, and solenoid lock—produced a compact yet functional prototype capable of remote access and RFID-based authentication.

The findings confirm that custom-built applications provide greater adaptability and control compared to preconfigured IoT platforms, making them more suitable for long-term deployment across diverse user environments. Furthermore, the system's dual-mode connectivity (LAN and WAN) enhances usability under varying network conditions.

However, this prototype does not yet include features such as intrusion notifications, battery status monitoring, or handling of network errors. These were excluded due to time constraints and limited budget, as the main focus of this project was to demonstrate core functionality and integration between IoT hardware and the custom Android application. A history log feature was implemented during the first iteration and prepared for reintroduction in the fourth iteration, but due to hardware failure in the ESP32 device, further updates could not be applied, and the feature was not included in the final prototype. Despite this limitation, the Android application was designed with the necessary interface to support event logging in future versions once reliable hardware becomes available.

For future work, potential improvements include incorporating biometric authentication, mobile push notifications, and offline data synchronization to strengthen robustness and scalability in real-world smart home applications. In addition, a more refined electrical design—particularly in power regulation and circuit protection—would be critical to ensure long-term stability, reducing hardware failure risks, and supporting advanced feature.

REFERENCES

- Alansari, Z., Soomro, S., Belgaum, M. R., & Shamshirband, S. (2018). The rise of Internet of Things (IoT) in big healthcare data: Review and open research issues. *Advances in Intelligent Systems and Computing*, 564, 675–685. https://doi.org/10.1007/978-981-10-6875-1_66
- Ardito, C., Buono, P., Desolda, G., & Matera, M. (2018). From smart objects to smart experiences: An end-user development approach. *International Journal of Human Computer Studies*, 114, 51–68. <https://doi.org/10.1016/j.ijhcs.2017.12.002>
- Assyauqi, H., Andriansyah, M. F., Nurul Ulla, L., & Sucipto, A. (2025). Sistem Keamanan Pintu Otomatis Berbasis IoT dengan Teknologi RFID dan Aplikasi Mobile Menggunakan Metode Fuzzy Mamdani. *Jurnal Informatika dan Sains Teknologi*, 3. <https://doi.org/https://doi.org/10.62951/modem.v3i2.405>
- Atzori, L., Iera, A., & Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks*, 54(15), 2787–2805. <https://doi.org/https://doi.org/10.1016/j.comnet.2010.05.010>
- Bagi, S., Suhu, M., Kelembapan, D., Setia, A., Ariyanto, S., Setiawan, I., Negara, M., Triwibowo, D. N., Feriyanto, Y., Sains, F., & Teknologi, D. (2025). Perancangan Aplikasi Mobile Berbasis Flutter untuk Pemantauan Data Sensor IoT. *JSAT: Journal Scientific and Applied Informatics*, 8(1). <https://doi.org/10.36085>
- Bapat, C., Baleri, G., Inamdar, S., & Nimkar, A. V. (2019). *Smart-Lock Security Re-engineered using Cryptography and Steganography*. https://doi.org/10.1007/978-981-10-6898-0_27
- Bjarnason, E., Lang, F., & Mjöberg, A. (2023). An empirically based model of software prototyping: a mapping study and a multi-case study. *Empirical Software Engineering*, 28(5). <https://doi.org/10.1007/s10664-023-10331-w>
- Bombarda, A., Bonfanti, S., Gargantini, A., & Riccobene, E. (2021). Developing a prototype of a mechanical ventilator controller from requirements to code with ASMETA. *Electronic Proceedings in Theoretical Computer Science, EPTCS*, 349, 13–29. <https://doi.org/10.4204/EPTCS.349.2>
- Borgia, E. (2014). The Internet of Things vision: Key features, applications and open issues. *Computer Communications*, 54, 1–31. <https://doi.org/https://doi.org/10.1016/j.comcom.2014.09.008>

- Boros, M., Siser, A., Kekovic, Z., & Mazal, J. (2018). Mechanical characteristics of cylinder pin tumbler locks as they relate to resistance testing. In *Communications - Scientific Letters of the University of Žilina* (Vol. 20, Issue 2, pp. 96–101). University of Žilina. <https://doi.org/10.26552/com.c.2018.2.96-101>
- Caleb, F. (2024, September 2). *Understanding Blynk IOT*. Electronic Ideas.
- Celestine, D. (2020). Smart Lock Systems: An Overview. *International Journal of Computer Applications*, 177(37), 40–43. <https://doi.org/10.5120/ijca2020919882>
- Espressif System. (2020). *Product Overview ESP32-C3 is an low-power and highly-integrated MCU-based solution that supports 2.4 GHz Wi-Fi and Bluetooth® Low Energy (Bluetooth LE)*. Espressif ESP32-C3 Wi-Fi + Bluetooth® Low Energy SoC Power consumption Normal Low power consumption components capable of working in Deep-sleep mode *Wireless Digital Circuits*.
- Gamma, W., Putri, R., Wiseto, I., & Agung, P. (2025). *Iot Smart Door Lock System Menggunakan Double Sensor Berbasis Mikrokontroler Esp32* (Vol. 8, Issue 1).
- Goli, V. R. (2021). React Native Evolution, Native Modules, And Best Practices. *Internasional Journal of Computer Engineering and Technology*, 12(2), 73–85. https://doi.org/10.34218/ijcet_12_02_009
- Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7), 1645–1660. <https://doi.org/https://doi.org/10.1016/j.future.2013.01.010>
- Hendriawan, M., Haryono, H., & Budiman, T. (2023). Development of Water Level Monitoring Applications in Smart Home Systems using Flutter. *Journal of Information System, Informatics and Computing*, 7(2), 213. <https://doi.org/10.52362/jisicom.v7i2.1197>
- Hendriawan, M., Haryono, H., & Budiman, T. (2025). Development of Water Level Monitoring Applications In Smart Home Systems Using Flutter. *Journal of Information System, Informatics and Computing*, 7(2), 213. <https://doi.org/10.52362/jisicom.v7i2.1197>
- Irianto, K. D. (2025). *Facial Recognition Implementation in Smart Home Doors Using IoT*. <https://doi.org/10.13140/RG.2.2.30931.69924>
- Kagawa, C. (2004). *Attacks Against The Mechanical Pin Tumbler Lock*. <https://api.semanticscholar.org/CorpusID:53797585>

- Kaya, S., Askar, A., & Ayyildiz, M. (2022). Smart Door Lock Design with Internet of Things. *International Journal of 3D Printing Technologies and Digital Industry*, 6(2), 201–206. <https://doi.org/10.46519/ij3dptdi.1074468>
- Kerrison, S. (2024). *Fingerprint Theft Using Smart Padlocks: Droplock Exploits and Defenses*. <http://arxiv.org/abs/2407.21398>
- Klotins, E., Boeva, V., Wnuk, K., Unterkalmsteiner, M., Gorschek, T., & Jansen, S. (2023). *A Method for Identification and Ranking of Requirements Sources*. <http://arxiv.org/abs/2309.17034>
- Kurniasyah, I. M., & Rakhmawati, L. (2025). Smart Door Lock Innovation Using Integration of Bluetooth Low Energy and MQTT IoT Protocol. *Indonesian Journal of Electrical and Electronics Engineering (INAJEEE)*, 8(1), 41–46. <https://doi.org/10.26740/inajeee.v8n1>
- Mascetti, S., Ducci, M., Cantù, N., Pecis, P., Cantù, N., & Ahmetovic, D. (2020). *Developing Accessible Mobile Applications with Cross-Platform Development Frameworks*. <https://doi.org/10.48550/arXiv.2005.06875>
- Nguyen-Duc, A., Wang, X., & Abrahamsson, P. (2017). What influences the speed of prototyping? An empirical investigation of twenty software startups. *Lecture Notes in Business Information Processing*, 283, 20–36. https://doi.org/10.1007/978-3-319-57633-6_2
- NXP Semiconductors. (2017). *General description PN532/C1 Near Field Communication (NFC) controller*.
- Orosz, G., & Nilsson, E. (2025, February 5). *Cross-platform mobile development with Expo*. The Pragmatic Engineer.
- Padmaja, B., Mahurkar, S., Krishna, E., & Patro, R. (2021). Google Firebase based Modern IoT System Architecture. *International Journal of Engineering Research & Technology (IJERT)*. <https://doi.org/DOI:10.17577/IJERTCONV9IS08023>
- Permana, K. A. K., Piarsa, I. N., & Wiranatha, A. A. K. A. C. (2024). IoT-Based Smart Door Lock System with Fingerprint and Keypad Access. *Journal of Information Systems and Informatics*, 6(3), 2086–2098. <https://doi.org/10.51519/journalisi.v6i3.844>
- Picking, Rich. (2017). *2017 Internet Technologies and Applications (ITA) : proceedings of the Seventh International Conference: Tuesday 12th - Friday 15th September 2017, Wrexham Glyndŵr University, Wales, UK*. IEEE.
- Pressman, R., & Maxim, B. (2014). *Software Engineering: A Practitioner's Approach*, 8th Ed.

- Ravian, A. Z., & D. Irianto, K. (2024). Perancangan Sistem Monitoring Pengunjung Puskesmas Berbasis Iot. *Technologia : Jurnal Ilmiah*, 15(3), 560. <https://doi.org/10.31602/tji.v15i3.15361>
- Rihhadatulaisy, Z. H., & Irianto, K. D. (2024). Designing an Automatic Room Temperature Control System for Smart Homes for the Elderly Using IoT. *International Journal Software Engineering and Computer Science (IJSECS)*, 4(2), 758–766. <https://doi.org/10.35870/ijsecs.v4i2.2844>
- Saraf, P. R. (2022). A Review on Firebase (Backend as A Service) for Mobile Application Development. *International Journal for Research in Applied Science and Engineering Technology*, 10(1), 967–971. <https://doi.org/10.22214/ijraset.2022.39958>
- Shevtsiv, N., Shvets, D., & Karabut, N. (2019). Prospects for Using React Native for Developing Cross-platform Mobile Applications. *Central Ukrainian Scientific Bulletin. Technical Sciences*, 2(33), 208–213. [https://doi.org/10.32515/2664-262X.2019.2\(33\).208-213](https://doi.org/10.32515/2664-262X.2019.2(33).208-213)
- Shiraz Ali, S. (2020). App Development using React Native, Expo and AWS. *International Journal of Trend in Scientific Research and Development (IJTSRD)*. <https://expo.io/@shirazali1997/image-upload>
- Sommerville, Ian. (2016). *Software engineering*. Pearson.
- Sonamoni, J. S., Sikdar, R., Akib, A. S. M. A. S., Islam, M. S., Sourov, S., Al Ahasan, M. A., Islam, M., Habib, M. A., & Mridha, M. F. (2024). IoT-Based Smart Remote Door Lock and Monitoring System Using an Android Application. *Engineering Proceedings*, 76(1). <https://doi.org/10.3390/engproc2024076085>
- Sonya, M. A. (2025). Sistem Pencahayaan Otomatis pada Smart Home untuk Lansia berbasis IoT. *Edusaintek: Jurnal Pendidikan, Sains Dan Teknologi*. <https://doi.org/https://doi.org/10.47668/edusaintek.v12i1.1366>
- Sudiarta, I. K. G., Indrayana, I. N. E., Suasnawa, I. W., Asri, S. A., & Sunu, P. W. (2020). Data Structure Comparison between MySQL Relational Database and Firebase Database NoSql on Mobile Based Tourist Tracking Application. *Journal of Physics: Conference Series*, 1569(3). <https://doi.org/10.1088/1742-6596/1569/3/032092>
- Sun, K. Y., Fernando, Y., & Safari, M. I. (2021a). Perancangan Sistem IoT pada Smart Door Lock Menggunakan Aplikasi BLYNK. *JUTSI (Jurnal Teknologi Dan Sistem Informasi)*, 1(3), 289–296. <https://doi.org/10.33330/jutsi.v1i3.1360>

- Sun, K. Y., Pernando, Y., & Safari, M. I. (2021b). Perancangan Sistem IoT pada Smart Door Lock Menggunakan Aplikasi Blynk. *JUTSI (Jurnal Teknologi Dan Sistem Informasi)*, 1(3), 289–296. <https://doi.org/10.33330/jutsi.v1i3.1360>
- Trihandini, W., Endri, J., & Salamah, I. (2025). Implementation of Fingerprint Biometrics on Smart Door Entrance Access Integrated with Internet of Things-based PINs. *Indonesian Journal of Artificial Intelligence and Data Mining (IJAIMD)*, 8(1), 102–109. <https://doi.org/10.24014/ijaidm.v8i1.31738>
- Tseloni, A., Thompson, R., Grove, L., Tilley, N., & Farrell, G. (2017). The effectiveness of burglary security devices. *Security Journal*, 30(2), 646–664. <https://doi.org/10.1057/sj.2014.30>
- Viatech. (2021). *A Brief History of Locks – From Catacombs to Modern Homes*. Viatech.Com.
- Vinod, M., Devadasan, S. R., Rajanayagam, D., Sunil, D. T., & Thilak, V. M. M. (2018). Theoretical and industrial studies on the electromechanical relay. In *Int. J. Services and Operations Management* (Vol. 29, Issue 3). <https://www.researchgate.net/publication/328927190>
- Zainuddin, A. A., Ammar Daniel Abd Rahman, Rizal Mohd Nor, Amir Aatief Amir Hussin, Nik Nor Muhammad Saifudin Nik Mohd Kamal, Abu Ubaidah Shamsudin, & Muhamad Syariff Sapuan. (2024). Innovative IoT Smart Lock System: Enhancing Security with Fingerprint and RFID Technology. *Malaysian Journal of Science and Advanced Technology*, 360–365. <https://doi.org/10.56532/mjsat.v4i4.335>
- Ziaul, M., & Zim, H. (2021). *TinyML: Analysis of Xtensa LX6 microprocessor for Neural Network Applications by ESP32 SoC*.