Development of a QR Code-Based Locking System for a Delivery Robot Using Firebase Realtime Database

Mahdi Ni'am Ma'ruf[1], and Anugerah Wibisana[2]

Politeknik Negeri Batam

Department of Electrical Engineering, Robotics Engineering Technology Study Program

Jalan Ahmad Yani, Teluk. Tering, Kec. Batam Kota, Kota Batam, Kepulauan Riau 29461, Indonesia

Email: mahdi130803@gmail.com

Abstrak

Mengamankan proses serah terima dalam robot pengiriman otonom sangat penting, karena sistem otentikasi statis seperti RFID rentan terhadap kloning. Mengatasi kerentanan ini, penelitian ini mengusulkan mekanisme penguncian berbasis kode QR dinamis untuk meningkatkan keamanan secara signifikan. Sistem ini menggunakan arsitektur master-slave di mana GUI menghasilkan kode unik dan sensitif waktu setiap 30 detik. *Firebase Realtime Database* berfungsi sebagai broker latensi rendah untuk aplikasi pemindaian Android, sedangkan *Mealy Finite State Machine* (FSM) mengelola logika kontainer melalui sensor ultrasonik dan magnetik. Pengujian fungsional menunjukkan tingkat keberhasilan 100% dalam pembuatan kode dan aktuasi mekanisme. Selain itu, evaluasi kinerja mencapai latensi rata-rata optimal 55,5 ms dalam skenario Wi-Fi dan Jaringan Seluler hibrid. Akibatnya, arsitektur penguncian dinamis ini terbukti menjadi solusi yang sangat responsif dan aman untuk bidang logistik pengiriman otonom yang maju.

Kata kunci: Sistem Pengunci, Kode QR, Antarmuka Pengguna Grafis (GUI), Firebase Realtime Database, Nomor Acak, Mesin Keadaan Terbatas (FSM)

Abstract

Securing the handover process in autonomous delivery robots is critical, as static authentication systems like RFID are prone to cloning. Addressing these vulnerabilities, this study proposes a dynamic QR code-based locking mechanism to significantly enhance security. The system utilizes a master-slave architecture where a GUI generates unique, time-sensitive codes every 30 seconds. Firebase Realtime Database serves as a low-latency broker for an Android scanning application, while a Mealy Finite State Machine (FSM) manages the container logic via ultrasonic and magnetic sensors. Functional testing demonstrated a 100% success rate in both code generation and mechanism actuation. Additionally, performance evaluation achieved an optimal average latency of 55.5 ms in a hybrid Wi-Fi and Mobile Network scenario. Consequently, this dynamic locking architecture proves to be a highly responsive and secure solution for the advancing field of autonomous delivery logistics.

Keywords: Lock System, QR Code, Graphical User Interface (GUI), Firebase Realtime Database, Random Number, Finite State Machine (FSM).

1. Introduction

The rapid advancement of automation and robotics has significantly improved efficiency within academic and industrial environments. A prominent innovation in this domain is the deployment of autonomous delivery robots for distributing logistics and food. Previous development of this robot focused on safety features such as human detection using depth cameras [1]. However, a fundamental challenge remains in ensuring the security of the handover process. Without a robust locking mechanism on the robot's storage container, goods are vulnerable to theft or unauthorized access, potentially undermining user trust in automated services.

To address security concerns, various electronic locking methods have been developed to replace conventional keys. Radio Frequency Identification (RFID) is widely adopted due to its convenience [2]. However, RFID systems suffer from a critical vulnerability the static nature of the credentials. RFID tags utilize fixed unique identifiers (UIDs) that can be easily cloned, lost, or stolen, creating significant security gaps. To mitigate this, a dynamic authentication method where credentials are temporary and unique for each transaction is essential. This concept parallels One-Time Passwords (OTPs), which are extensively used in digital security [3].

Addressing these limitations, this study proposes a novel security system for delivery robot containers using dynamic QR codes. While QR codes have been applied in e-lockers [4] and safes [5], this research integrates a time-sensitive algorithm within a master-slave architecture using Firebase Realtime Database for low-latency synchronization, similar to its implementation in real-time monitoring systems [6]. The proposed system generates a new QR code every 30 seconds, ensuring that access credentials cannot be reused. Combined with a dedicated Android application for scanning [7], this approach establishes a two-factor authentication framework suitable for dynamic environments.

2. Methodology

This section details the approach used to develop the system, structured into three interrelated stages. The overall architecture is visually illustrated in Figure 1. First, the System Architecture establishes a master-slave framework utilizing Firebase Realtime Database as a low-latency broker. Next, Software Design elaborates on the Android application for validation and the Python-based GUI for dynamic QR code generation. Finally, Hardware Design covers electronic and mechanical integration, with operational logic governed by a Mealy Finite State Machine (FSM) to ensure secure access.

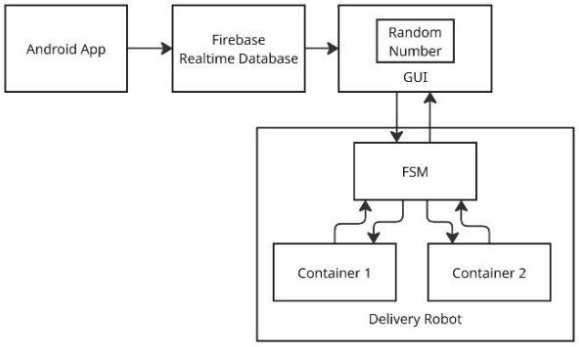


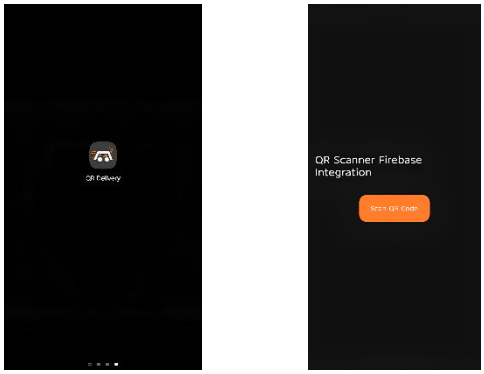
Figure 1: System Architecture

As illustrated in Figure 1, the system utilizes a centralized master-slave architecture connecting a Laptop (Master), two container units (Slaves), and an Android App via Firebase Realtime Database. The workflow begins when the Master generates a unique, dynamic QR code. The user scans this code using the Android App, which immediately synchronizes the data to Firebase. The Master then retrieves and validates this data against the generated code. Upon successful verification, the Master issues a direct command to the specific container to activate the unlocking mechanism, ensuring that all control logic remains centralized and secure.

2.1 Android App

This section outlines the software architecture designed to ensure seamless synchronization between the user interface and robotic hardware. The framework integrates the Android Application for authentication, Firebase Realtime Database as a low-latency bridge, and a Python-based GUI for dynamic credentials. This interconnected structure ensures instantaneous data flow between the client and the master unit. Furthermore, the operational logic is strictly governed by a Finite State Machine (FSM). This logic layer is essential for managing secure state transitions, ensuring the locking mechanism actuates only under validated conditions.

The Android application functions as the primary client-side interface, facilitating a seamless QR code validation process similar to modules found in autonomous delivery systems [8]. Built within Android Studio, the app integrates with the Firebase SDK to establish a robust communication channel. This enables real-time, low-latency data transmission to the cloud, ensuring the master unit receives input without delay. Designed for operational simplicity, the User Interface (UI) focuses on a single core function scanning. As depicted in Figure 2, the minimalist interface features a prominent 'Scan QR Code' button. Upon activation, the app utilizes the device's camera to capture the dynamic code, extracts the encrypted string, and automatically uploads it to the database, effectively minimizing human error during authentication.



(A) App Icon **(B) App Interface**

Figure 2: Android App

Designed for simplicity, the application focuses on a single core function scanning. The visual implementation is illustrated in Figure 2, where Figure 2(A)App Icon displays the 'QR Delivery' launcher icon, and Figure 2(B)App Interface depicts the minimalist main interface featuring the 'Scan QR Code' button. Upon activation, the app triggers the device's camera to capture the code, extracts the data, and automatically transmits it to the Firebase Realtime Database for immediate validation by the master system.

2.2 Firebase Realtime Database

Firebase Realtime Database was used in this study as a cloud-based data communication and synchronization platform. The platform was chosen for its ability to deliver real-time data at low latency, which has proven effective for monitoring applications. To implement this functionality, two different Firebase Software Development Kits (SDKs) are used the Firebase SDK for Android integrated through Android Studio, and the Firebase SDK for Python which is used on the laptop side (GUI).

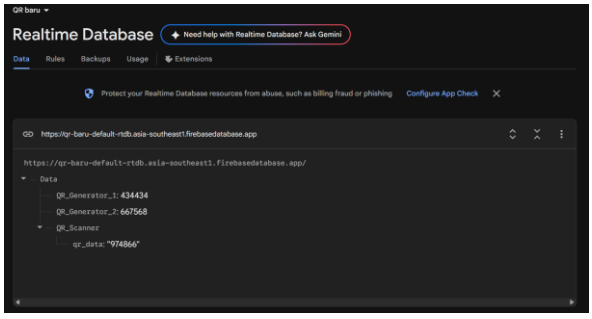


Figure 3: Firebase Data Interface

As shown in Figure 3, the database structure is designed to facilitate validation by segregating entries into QR_Generator_1 and QR_Generator_2 for the respective containers. A separate QR_Scanner node serves as the input buffer for the Android application. The workflow initiates when the master unit updates the generator keys; upon scanning, the app writes the data to the scanner node, which the laptop immediately retrieves and compares against the active code to authorize access. Since the unlocking mechanism relies entirely on this cloud-based synchronization loop, the speed of data transmission is a critical factor for system reliability. A significant delay in this exchange could render the smart lock impractical. Therefore, to evaluate the system's responsiveness, latency testing is conducted by measuring the total round-trip time of the data transmission. Latency is defined as the time elapsed from the moment the scanning application sends the data Time send until the feedback is received by the master unit Time receive. This duration is calculated using Equation (1):

$$Latency = T_{receive} - T_{send} \quad (1)$$

Where Latency represents the total delay in milliseconds (ms), $T_{receive}$ denotes the timestamp when the master unit receives the validation signal, and T_{send} is the timestamp when the Android client transmits the scanned code. The testing process involves three distinct network scenarios (1) Master and Client on public Wi-Fi, (2) Master and Client on private mobile networks, and (3) a Hybrid configuration. These scenarios are designed to simulate diverse real-world operating environments, allowing for a comprehensive analysis of how signal interference and bandwidth congestion affect data transmission. By comparing these specific configurations, the study aims to identify the most stable communication channel that guarantees minimal latency, thereby ensuring that the locking mechanism acts instantaneously and securely during the handover process.

2.3 Graphical User Interface (GUI)

The Graphical User Interface (GUI) functions as the central master control unit for the delivery robot system. It was developed using the Python programming language integrated with the Tkinter library. This specific development approach aligns with methodologies applied in research related to centralized control systems [9], [10]. The primary objective of this interface is to mitigate the security vulnerabilities inherent in static authentication methods, such as RFID cloning. To achieve this, the GUI implements a dynamic credential generation mechanism that directly addresses the critical need for temporary and unique access keys. Specifically, the system utilizes a randomization algorithm to generate a unique six-digit numerical code, which is then continuously rendered as a QR code. This credential is strictly time-sensitive, automatically refreshing every 30 seconds. By adopting a concept analogous to One-Time Passwords (OTP), the system ensures that expired codes cannot be reused for unauthorized access. Furthermore, the operational workflow within the GUI is structured sequentially to guide users intuitively, as illustrated in the subsequent interface views.

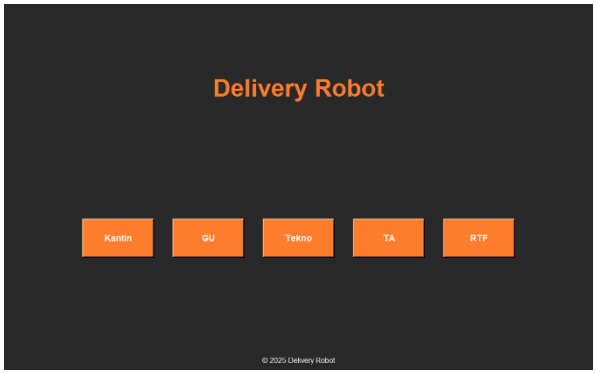


Figure 4: GUI Main Menu

Main Menu Display As illustrated in Figure 4, the initial interface presents the user with a main menu designed to determine the delivery destination (e.g., Canteen, GU, Tekno), which is integrated with the robot's path planning system.

Figure 5: GUI Container Selection

Container Selection Upon successfully selecting the delivery destination, the system immediately transitions to the interface shown in Figure 5. In this operational view, the user is required to manually specify which of the two available storage compartments, Container 1 or Container 2, will be utilized. This explicit selection is critical as it determines which specific independent locking mechanism and sensor array the system will subsequently activate for the transaction, ensuring that only the relevant hardware components are engaged.

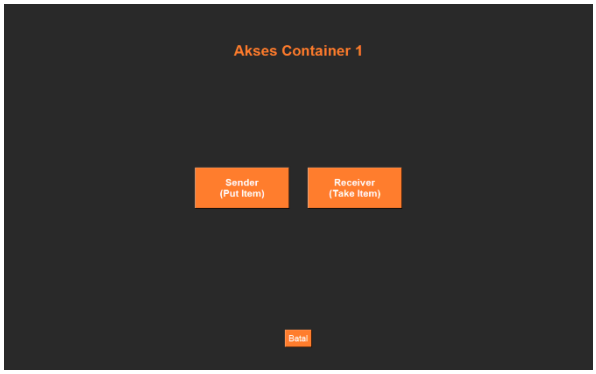


Figure 6: GUI Selection Mode

Selection Mode View As illustrated in Figure 6, this interface presents two primary options that differentiate the user's role. The 'Sender (Put Item)' button enables the sender (e.g., the canteen attendant) to unlock the container for placing the food. While, the 'Receiver (Take Item)' button allows the recipient to retrieve the food items once the delivery is complete.

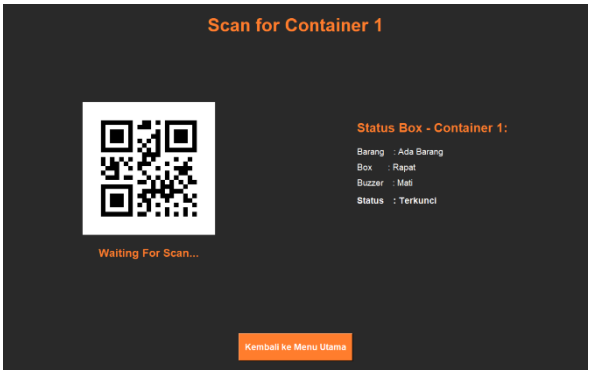


Figure 7: GUI Scanning Mode

The Operational View, as illustrated in Figure 7, serves as a central dashboard divided into two functional zones. The left section displays the dynamic, time-sensitive QR code used for authentication. Simultaneously, the right section features the 'Status Box' panel, which provides real-time telemetry by monitoring critical parameters such as item presence ('Barang'), door security ('Box'), and lock engagement ('Status'). This layout ensures the user can effectively verify the system's physical integrity throughout the delivery process.

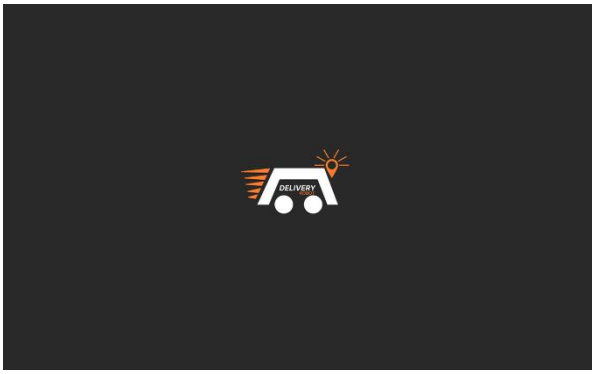


Figure 8: GUI In-Transit Interface Display

Travel Display During the transit phase, as illustrated in Figure 8, the system activates a dedicated interface to signify the robot is navigating to the destination. This display serves as a status indicator, informing users that delivery is in progress and the system is locked for safety. Crucially, this view disables all touch interactions to prevent accidental triggering of the locking mechanism while the robot is in motion. The system remains in this state until the navigation subsystem signals arrival, at which point the interface automatically reverts to the scanning screen, ready for the handover process.

2.4 Delivery Robot

The Delivery Robot used in this study, as illustrated in Figure 9, is an autonomous mobile unit designed to operate within the Politeknik Negeri Batam campus environment. Its primary function is to facilitate food delivery services, autonomously transporting meals from canteen to various buildings across the campus

Figure 9: Delivery robot

As an unmanned system navigating through public spaces, the robot requires a highly secure storage mechanism to protect the goods during transit. This operational requirement serves as the real-world implementation case for the proposed smart lock system, ensuring that the cargo remains locked while moving and can only be accessed by the authorized recipient upon arrival.

2.4.1 Hardware Specification

In order to ensure the reliability and reproducibility of the test results, the hardware environment used in this research is explicitly defined. The system relies on a central processing unit (Master) hosted on a laptop to handle the Python-based GUI and database synchronization. Complementing this, a mobile phone is utilized as the client device to run the Android scanning application. Table III presents the technical specifications for both devices, highlighting critical parameters such as processor speed, memory and connectivity as these capabilities are significant variables in the network latency analysis.

TABLE III
HARDWARE SPECIFICATIONS

Laptop	
Processor	AMD Ryzen 7 7730U w/ Radeon Graphics (16 CPUs). ~2.0GHz
RAM	16384MB
Storage	512GB M.2 NVMe™ PCIe® 3.0 SSD
Graphic Card	AMD Radeon™ Graphics
Wi-Fi	Wi-Fi 6E (802.11ax)
Mobile Phone	
Processor	MediaTek Dimensity 7300 Ultimate
RAM	8 GB
Storage	256 GB
Wi-Fi	Wi-Fi 802.116E
Connection	5G

The specifications detailed in Table III were selected to minimize hardware-induced latency, ensuring that the performance analysis focuses primarily on network conditions rather than device processing limitations. The laptop, functioning as the Master Control Unit, utilizes a high-performance multi-core processor and substantial RAM to execute the Python-based GUI and background Firebase synchronization threads without processing bottlenecks. On the client side, the mobile device is equipped with a 5G-enabled chipset and Wi-Fi 6 capabilities. These connectivity standards are critical for the experiment, as they allow the scanning application to transmit data at optimal speeds, thereby isolating network stability as the primary variable in the subsequent latency experiments.

2.4.2 Finite State Machine

The operational logic for each container on the robot is modeled as a Finite State Machine (FSM). This approach is used to systematically manage transitions between conditions, ensure safe operation, and respond intelligently based on input from sensors. The FSM approach is also commonly used in automated systems such as vending machines [11] and signal detectors [12]. Specifically, the system implements the FSM Mealy model, where the system's output is determined by the current state and the sensor input in real-time. The following diagram illustrates the general state transitions that occur in robot locking systems

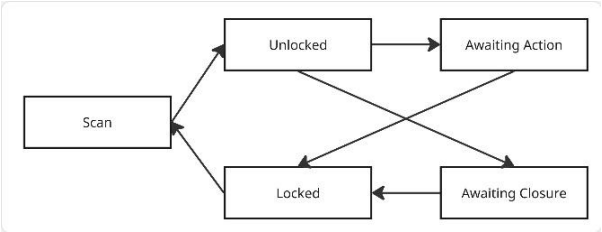


Figure 10: FSM Diagram

The system's operational logic, illustrated in Figure 10, centers on five primary states. The cycle begins in the Locked state, keeping the container secured. Access validation triggers the Scan state, which transitions to Unlocked upon verification, releasing the solenoid. Next, the system enters Awaiting Action, using ultrasonic sensors to monitor interactions. Crucially, the validation logic adapts to the workflow Receiver Mode confirms completion by detecting an increased distance reading indicating item removal, while Sender Mode waits for a distance decrease to verify package placement. Once the specific activity is completed, the state shifts to Awaiting Closure. Here, an auditory alert prompts the user to close the door before the system resets to the locked position. To effectively accommodate the robot's dual-purpose functionality, these state transitions are strictly governed by two distinct truth tables, specifically tailored for Receiver Mode and Sender Mode respectively.

Table I presents the system logic for the Receiver Mode as the robot delivers the packet to its destination. In this mode, the system detects a change in the condition of the compartment from Occupied to Empty.

TABLE I
RECEIVER MODE

Condition	Item	Lid	Buzzer	Lock	Status
(Start)	Occupied	Closed	Off	Locked	Standby
(Scan)	Occupied	Open	Off	Unlock	Waiting for Pickup
(Action)	Empty	Open	On	Unlock	Item Taken
(End)	Empty	Closed	Off	Locked	Completed

In the third (Action) condition, the Buzzer will activate automatically when the sensor detects an item has been taken, providing audio feedback for the user to immediately close the door.

Sender Mode Table II shows the system logic when the robot receives a packet from the sender. In contrast to receiver mode, the system detects a change in condition from Empty to Occupied.

TABLE II
SENDER MODE

Condition	Item	Lid	Buzzer	Lock	Status
(Start)	Empty	Closed	Off	Locked	Standby
(Scan)	Empty	Open	Off	Unlock	Waiting for Item
(Action)	Occupied	Open	On	Unlock	Item Placed
(End)	Occupied	Closed	Off	Locked	Completed

Similar to receiver mode, the security system in the third condition (Action) will prevent the robot from locking the door if the door is closed but the item has not been detected entering (preventing the delivery of an empty box), or if the door has not been tightly closed. This condition table applies to each container individually, allowing the robot to manage different states for container 1 and container 2 simultaneously (Independent).

2.4.3 Container

The robot is equipped with two distinct storage compartments, containers. Each container functions as an independent secure module designed to protect the delivered items from unauthorized access during transit. Physically, the container serves two main purposes providing a hygienic space for food storage and acting as a housing for the electronic locking subsystems. The design ensures that the control components are isolated from the cargo area to prevent damage and maintain system integrity.

As illustrated in Figure 11, the internal electronics of each container unit are managed by an Arduino Nano, acting as a slave microcontroller responsible for receiving commands from the master laptop and executing FSM logic.

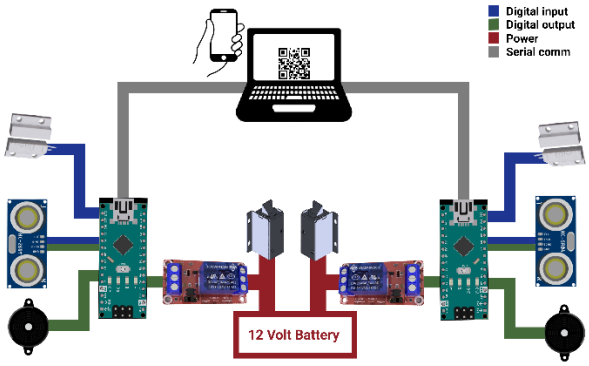


Figure 11: Wiring Diagram

As illustrated in Figure 11, the system architecture utilizes a specific color-coded wiring scheme to distinguish between data signals and power distribution. Blue lines (Digital Input) transmit environmental data from the Ultrasonic Sensor, used for cargo detection, and the Magnetic Sensor, used for lid status verification, directly to the microcontroller. In response, the Arduino triggers actuators via green lines (Control Signal) to activate the buzzer and the relay module. The relay serves as a critical isolator and switch for the red lines (12V Power), allowing the high-current battery supply to energize the Solenoid Lock safely. Finally, the grey line (Serial Comm) integrates the subsystem with the main unit, ensuring that QR code unlocking commands are executed with real-time synchronization.

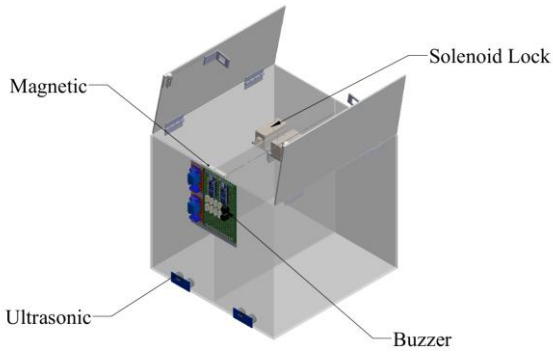


Figure 12: Mechanical Design & Component Placement

Mechanically, the system utilizes a delivery box divided into two independent containers, as illustrated in Figure 12, where the layout of sensors and actuators is strategically positioned to maximize functionality and reliability. For the locking mechanism, the solenoid lock is installed on the upper center of the container bulkhead with its latch attached to the side of the lid, ensuring a precise and sturdy engagement. Regarding sensor placement, a Magnetic Sensor is positioned at

the top of the bulkhead to accurately monitor the closure of the cover, while an Ultrasonic Sensor is mounted on the bottom wall to optimize the detection range for identifying the presence or absence of goods. Finally, the Buzzer is integrated directly onto a custom PCB to simplify wiring complexity and ensure close proximity to the microcontroller and power sources.

3. Results

This section presented the comprehensive experimental findings, structured into distinct functionality and performance evaluations. The analysis rigorously validated the reliability of the random code generation algorithms and the physical locking mechanisms. Furthermore, the study assessed real-time system responsiveness through detailed network latency measurements across varying connection scenarios. Ultimately, these results conclusively confirmed the system's robustness and validated its suitability for implementation in secure autonomous delivery logistics.

3.1 Experimental Testing

Functionality testing aimed to verify two core functions (a) the system's ability to generate unique random codes, and (b) the system's success in unlocking after QR code validation. Complementing these functional tests, the research further evaluated system performance through (c) latency testing, which measured the real-time responsiveness of data transmission across different network scenarios.

3.1.1 Random Number Testing

The security integrity of the proposed locking mechanism relied heavily on the absolute unpredictability of access credentials. Therefore, the initial stage of functional testing rigorously evaluated the reliability of the Random Number Generator algorithm. The primary objective was to verify that the system consistently produced unique and non-repetitive 6-digit codes for both Container 1 and Container 2 across multiple iterations. The resulting data from these generation trials were detailed in Table IV.

TABLE IV
RANDOM NUMBER QR GENERATOR TEST RESULTS

Test	Container 1	Container 2
1	627429	378073
2	104369	616320
3	603608	662355
4	818648	762339
5	110635	244293
6	591259	624122
7	989351	375248
8	628857	501930
9	603181	858084
10	380557	652071

As presented in Table IV, the results illustrated the outcomes of 10 distinct random code generation attempts. These results empirically validated that the GUI successfully executed the Random Number Generator algorithm to produce unique and heterogeneous 6-digit codes for both containers in each test iteration. Furthermore, the data indicated a zero-collision rate, confirming that the system maintained high entropy and unpredictability. This capability was fundamental to the system's security architecture, ensuring that access credentials remained distinct for every transaction and effectively preventing potential unauthorized access through pattern recognition.

3.1.2 Lock System Testing

Following the verification of code generation, the next phase evaluated the physical response of the locking mechanism, as demonstrated in the attached Demo Video. This test aimed to validate the end-to-end workflow of the system, ensuring that a valid QR code scan triggered the appropriate solenoid lock to open. The testing process involved scanning the generated QR codes using the Android application and observing the state of the mechanical lock on the container unit.

TABLE V
LOCK OPENING TEST RESULT

Test	Container 1	Container 2
1	Success	Success
2	Success	Success
3	Success	Success
4	Success	Success
5	Success	Success

Table V presented the reliability results of the lock opening mechanism, tested across five iterations for both containers. The system achieved a 100% success rate, successfully actuating the solenoid in every trial. This confirmed the robustness of the end-to-end communication pipeline, validating that the integration between the Android app, Firebase, and master-slave hardware ensured accurate and instantaneous locking mechanism control.

3.1.3 Latency Testing

To assess real-time responsiveness, a latency analysis was conducted to measure communication delays under varying network conditions. Given the system's reliance on cloud synchronization, minimizing the interval between the QR scan and solenoid actuation was critical for practical usability. Therefore, the experiment measured round-trip transmission times across three distinct scenarios, ranging from congested public Wi-Fi to private mobile networks. This approach aimed to identify the most stable network configuration by isolating specific connection variables. The quantitative testing findings were detailed in the following Table IV.

TABLE VI
LATENCY TEST USING Wi-Fi

Test	Latency (ms)
1	62
2	120
3	50
4	70
5	366
6	153
7	126
8	1350
9	315
10	137
11	345
12	1196
13	49
14	259
15	115

As detailed in Table VI, Scenario 1 recorded the highest average latency of 313.2 ms. In this scenario, the latency fluctuated significantly, with a minimum value of 49 ms and an extreme spike of up to a maximum of 1350 ms. This very high variability was directly due to the use of the same single network source, i.e., campus public Wi-Fi, by both devices (master and client) simultaneously. Given that this network was a public facility that was massively accessed by students in the campus environment, there was significant bandwidth sharing and high data traffic load. As a result, both system devices had to compete for data allocation on the same channel, which inevitably led to connection fluctuations and a sharp increase in latency as data exchange occurred.

TABLE VII
LATENCY TEST USING MOBILE NETWORK

Test	Latency (ms)
1	42
2	52
3	50
4	42
5	40
6	61
7	54
8	41
9	58
10	42
11	64
12	43
13	44
14	158
15	42

As detailed in Table VII, in Scenario 2, the system showed excellent and stable performance with an average latency of 112.1 ms. Latency values ranged from a minimum of 63 ms to a maximum of 212 ms. Although both devices (Smartphones and Laptops) used the same type of internet source, namely the Mobile Network, the resulting stability was much better than the use of public Wi-Fi. This was because mobile networks were private and provided a more dedicated allocation of resources for each user, in contrast to public Wi-Fi which shared bandwidth in aggregate to many users at a single access point. This private network characteristic minimized interference due to the density of other users, thus ensuring more consistent data transmission reliability.

TABLE VIII
LATENCY TEST USING Wi-Fi & MOBILE NETWORK

Test	Latency (ms)
1	128
2	87
3	63
4	84
5	198
6	148
7	111
8	80
9	103
10	109
11	117
12	212
13	68
14	80
15	94

As explicitly detailed in Table VIII, Scenario 3 demonstrated the most superior performance metrics among all configurations, achieving the lowest average latency of 55.5 ms. The data revealed a consistent range, with a minimum delay of 40 ms and a maximum of 158 ms. This exceptional stability was primarily achieved through the strategic separation of internet connection sources, where the master unit connected via campus Wi-Fi while the client utilized an independent mobile network. By leveraging these two distinct network infrastructures, the system effectively eliminated bandwidth competition that typically occurred at a single access point. Consequently, with data paths strictly isolated from each other, traffic load was distributed more efficiently, preventing potential data bottlenecks and ensuring a significantly faster, more robust system response.

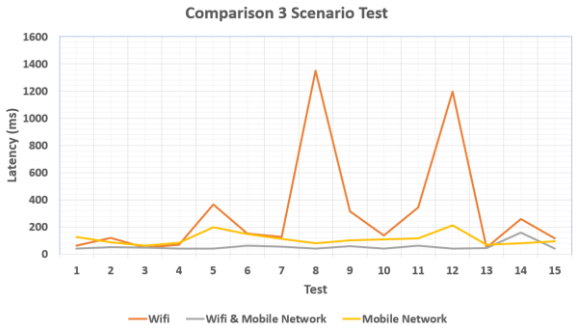


Figure 13: Comparison of three Scenarios

Comparative Analysis, as visually illustrated in Figure 13, revealed significant performance disparities among the three scenarios. Scenario 3 (Wi-Fi & Mobile Network) proved to be the most efficient architecture, where the separation of network sources effectively negated internal traffic interference, resulting in the lowest average latency of 55.5 ms. Scenario 2 (Mobile Network & Mobile Network) followed with 112.1 ms, benefiting from the inherent stability of private mobile networks over public ones. In contrast, Scenario 1 (Wi-Fi & Wi-Fi) was the most vulnerable to fluctuations, averaging 313.2 ms with a peak of 1350 ms, due to reliance on a congested public source. These findings confirmed that connection source management, specifically path separation, was a key determinant factor in optimizing system latency performance.

3 Conclusion

This study successfully developed a secure locking mechanism for autonomous delivery robots using dynamic QR codes to address the vulnerabilities of static keys. The proposed system integrates a time-sensitive code generator, an Android application, and Firebase Realtime Database within a microcontroller master-slave architecture. Operational logic is efficiently managed by a Mealy Finite State Machine. Experimental validation demonstrated exceptional reliability, yielding a 100% success rate in both code generation and physical unlocking tests. Furthermore, performance analysis confirmed high responsiveness, achieving an optimal average latency of 55.5 ms in the hybrid network scenario. These findings prove the system's effectiveness for secure logistics. Future work will prioritize implementing a secure Login System for user authentication and enforcing stricter database encryption rules to further mitigate unauthorized access risks and enhance overall system integrity.

Acknowledgement

The authors gratefully acknowledge the Robot Delivery Team for their solid teamwork and contribution. We thank Politeknik Negeri Batam for facilitating and funding this research under the Project-Based Learning (PBL) framework. Special appreciation is extended to our lecturers and Project Manager for their guidance, as well as to BRAIL (Barelang Robotics and Artificial Intelligence Lab) for providing the essential workshop facilities.

References

[1] W. P. S. Simanjuntak and A. Wibisana, "Depth camera-based human detection using YOLOv5," in Proceedings of the 7th International Conference on Applied Engineering (ICAE 2024), Atlantis Press, 2024, pp. 150–162. doi: 10.2991/978-94-6463-620-8_12.

[2] Y. Hasan, Abdurrahman, Y. Wijanarko, S. Muslimin, and R. Maulidda , "The automatic door lock to enhance security in RFID system", Journal of Physics: Conference Series, 1500(1), 2020.

[3] M. Fahrizal and A. Solichin, "Pengamanan m-commerce menggunakan one time password metode Pseudo Random Number Generator (PRNG)", RABIT: Jurnal Teknologi dan Sistem Informasi Univrab, 5(2), pp. 108-116, 2020.

[4] N. P. Hadiansyah, R. Tulloh, and R. M. Negara, "Design and implementation of e-locker using QR code and website monitoring based on Internet of Things", e-Proceeding of Applied Science, 6(1), pp. 499-512, 2020.

[5] A. Ismangil, Mulyati, and Erniyati, "Sistem brankas menggunakan barcode scanner berbasis Android", JIPETIK: Jurnal Ilmiah Penelitian Teknologi Informasi & Komputer, 1(2), pp. 69-72, 2020.

[6] F. C. Y. Pratama, N. M. Adriansyah, and V. S. W. Prabowo, "Penerapan Firebase Realtime Database untuk monitoring cuaca secara realtime", e-Proceeding of Engineering, 11(6), pp. 6526-6529, 2024.

[7] F. Ayu and A. Mustofa, "Sistem aplikasi absensi menggunakan teknologi barcode scanner berbasis Android", IT Journal Research and Development (ITJRD), 4(2), pp. 94-103, 2020.

[8] G. Novalisza, E. Susanto, and I. M. Rodiana, "Sistem pendeteksian kode QR pada robot pengantar makanan", e-Proceeding of Engineering, 11(5), pp. 5464-5467, 2024.

[9] Sihtijaturiman, H. Darmanto, and N. E. Sabiliat, "Implementasi pengendali on/off peralatan listrik rumah/gedung terpusat berbasis Raspberry Pi menggunakan Python dan Tkinter", Jurnal INSTEK (Informatika Sains dan Teknologi), 1(2), pp. 8-16, 2016.

[10] P.-S. Tsai, T.-F. Wu, J.-Y. Chen, and C.-L. Tsai, "Development of automated optical inspection and classification systems", Sensors and Materials, 34(10), pp. 3895-3910, 2022.

[11] G. V. Saragih, A. Faisal, and W. Gata, "Desain vending machine rokok dengan mengimplementasikan Finite State Automata terintegrasi dengan e-KTP", MATICS: Jurnal Ilmu Komputer dan Teknologi Informasi, 12(1), pp. 55-60, 2020.

[12] P. Garapati and S. Musala, "Moore and Mealy negative edge detector a VHDL example for Finite State Machine", 2020 International Conference on Communication and Signal Processing (ICCSP), India, pp. 1159-1161, 2020.