# Developing Embedded Systems for E-Bike Rentals with Logistic Map-Based Token Generation

Ahmad Juniar[1*], Al Kautsar Permana[2], Finna Suroso[3], Gita Mustika Rahmah[4], and Muchamad Taufiq Anwar[5]
[1,3–5]Automotive Industry Information System, Politeknik STMI Jakarta
Jakarta Pusat, Indonesia 10510
[2]Automotive Engineering Technology, Politeknik STMI Jakarta
Jakarta Pusat, Indonesia 10510
Email: [1]ahmadjuniar@stmi.ac.id, [2]alkautsar.permana@stmi.ac.id, [3]finna@stmi.ac.id, [4]gitamustika@stmi.ac.id, [5]taufiq@stmi.ac.id

***Abstract*—The research presents a novel electric bike rental system that operates efficiently without an internet connection. The system integrates a logistic map algorithm into an embedded microcontroller and a web-based rental information system. The logistic map has pseudo-random code generation properties, allowing identical activation tokens to be generated on both the server and the embedded system, enabling robust token validation on e-bike units. Users rent e-bikes by scanning a barcode on the e-bike unit. The barcode then directs the user to a web interface, where they can select the rental duration via a secure payment gateway. The rental system is pay-per-use, eliminating the need for deposits or account registration and enhancing user convenience. Once payment is verified, a message is sent containing the receipt details and an activation token that can be used on the e-bike. The obtained token generation and verification times are 0.68 ms and 0.72 ms, respectively. This test demonstrates the ability to generate tokens in real time without significant latency. However, this approach faces practical issues of token collisions and hardware limitations. Token collisions occur when a 9-digit token appears twice across five different ID tokens, with counters ranging from 1 to 10,000 for the same e-bike ID and rental duration. Rounding to significant digits causes this problem due to the microcontroller's inherent limitations. However, these duplications occur at widely separated counters, making them difficult to exploit. This system effectively balances simplicity, security, and scalability for local e-bike sharing services.**

***Index Terms*—Bike-Sharing System (BSS), Electric Bike Rental, Logistic Map Algorithm, Embedded Systems**

Received: Feb. 17, 2025; received in revised form: Aug. 28, 2025; accepted: Sep. 02, 2025; available online: March 09, 2026.
*Corresponding Author

## I. Introduction

THE emergence of Bike-Sharing Systems (BSS) has changed the electric bike rental market. These systems are commonly equipped with embedded systems based on the Internet of Things (IoT). Generally, IoT-based research in this domain can be categorized into several focus areas, including data acquisition, demand forecasting, scheduling process planning, and electronic security. Data acquisition involves implementing IoT, where bike units are equipped with sensors connected to a server to display the status and location of the units [1–3], as well as energy consumption [4].

In demand forecasting, research uses datasets collected from sensors and customer accounts within the bike-sharing information system to predict and schedule the effective positioning of bikes across large cities or areas using machine learning or deep learning approaches. For instance, the machine learning approach employing gradient boosting methods has been used to predict the number of bikes rented and returned within the next hour [5]. Meanwhile, a deep learning approach using a hybrid Long Short-Term Memory and Gated Recurrent Unit (LSTM-GRU) method has been applied to balance bike rental demand during peak hours [6]. Additionally, the Bidirectional Long Short-Term Memory (Bi-LSTM) method [7] and the Graph Convolutional Neural Network with Data-driven Graph Filter (GCNN-DDGF) method [8] have addressed similar issues. Another IoT application in bike-sharing systems is electronic security systems, which detect vehicle and battery theft [9]. Beyond IoT-based systems, simulators have been developed to improve the

performance of bike-sharing systems by balancing bike availability to prevent shortages and surpluses [10].

The previous study has introduced an e-bike rental service using a website as its ordering platform [3]. Customers need to access the website to locate nearby units. Once a unit is found, the customer scans the QR code on each unit to unlock the bike and make it ready for use. The system deducts the customer's deposit based on per-minute rates set by the system. Another study has examined various fare systems implemented by BSS operators, including membership schemes, subscription rates, and hourly pricing, and has further classified ordering modes into websites, sales points, and hotlines [11]. One of these BSS operators offers an annual membership with a free fare for the first 30 minutes and a fare of 4 CHF (Swiss franc) for each subsequent hour. For daily use, the minimum rate is 6 CHF/hour, with a maximum charge of 40 CHF/day [11]. GPS technology integrated with IoT is critical for tracking bike locations. However, this is challenging in areas without internet access, such as rural areas.

In the research, a novel approach using chaotic map algorithms embedded in the bike microcontroller system and the bike-sharing information system is proposed. The reason for using the chaotic map approach lies in its ability to generate pseudo-random codes. Depending on the n-dimensional nature of the chaotic map, the algorithm uses one or more parameters to produce unique codes [12]. By leveraging the properties of chaotic maps, this algorithm can be implemented on two distinct systems: the bike-sharing information server and the microcontroller. The codes generated by these two systems will be identical each time an electric bike is rented, as long as the parameters used by both systems are kept consistent. This condition ensures that the server and the embedded system can communicate effectively without an Internet connection. Customers who want to use the bike-sharing service are directed to purchase a voucher via the bike-sharing information system. The voucher indicates the hourly rate options for customers to choose from. A pseudo-random code is then generated and input into the bike unit, activating the bike based on the selected rental time. This application is particularly suitable for recreational areas where the absence of GPS in the bike units reduces security concerns. It eliminates the need for human resources to operate the bike-sharing service at a counter.

To summarize, the researchers have several main contributions. First, a non-IoT-based bike-sharing system is proposed, enabling the embedded system to operate without Internet connectivity. Consequently, greater energy efficiency is achieved, and applicability

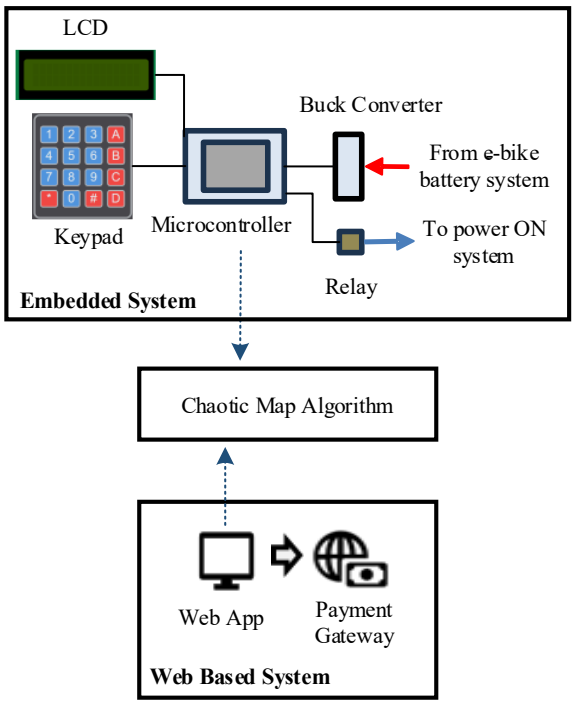



Fig. 1. Architecture of the e-bike rental system.

to remote tourist areas is enhanced, as constant Internet access is no longer required. Second, the logistic map algorithm is implemented in both the web-based rental system and the microcontroller to ensure consistent token generation across the two systems upon customer payment. Third, users only need to utilize built-in smartphone features, specifically the camera for QR code scanning and the web browser. They are not required to install additional applications or register a new account. Fourth, the system is designed to operate on a pay-as-you-go basis, eliminating the need for customers to deposit money that may go unused if they do not frequently visit recreational areas. Fifth, an embedded system for bike units is designed to activate the bike only upon the correct token being entered.

## II. Research Method

### A. System Architecture

Figure 1 illustrates the architecture of the electric bike rental system. When customers wish to rent an electric bike, they can scan the QR code displayed on the bike unit, which redirects them to the electric bike rental information system webpage. The customer must then complete the form on the website with their details and choose between a 30-minute or 60-minute rental period.

The customer pays the rental fee via a secure payment gateway after selecting the rental duration.

Once payment is successfully received, the information system generates a unique 9-digit token. This token must be entered into the electric bike's control unit to activate the system. The bike unit will unlock and become active if the token is verified as valid.

Then, a countdown timer is initiated and displayed on the bike's LCD screen to show the remaining rental time. Once the rental time expires, the electric bike automatically powers down. The bike's battery powers the microcontroller system, ensuring it is always ready for operation. The battery voltage is stepped down using a buck converter to match the microcontroller's power requirements. The implementation of software and hardware for the rental system is depicted in the flowchart in Fig. A1 in Appendix.

In the proposed system, the chaotic map variables include the bike unit ID, rental duration, and a counter. The counter increments with each new bike unit rental, ensuring a unique input for every transaction. Its initial value is set to 0 when the system is first activated. This configuration guarantees consistent output generation across both the microcontroller and the web information system. A detailed explanation of the chaotic map's implementation and its role in the system will be provided in the following section.

### *B. Chaotic Map*

Chaos theory is a branch of mathematics that deals with nonlinear dynamical systems. Nonlinear dynamical systems describe how a system evolves over time using nonlinear equations. A chaotic map is a specific type of nonlinear dynamical system. A chaotic map exhibits deterministic yet seemingly random behavior, characterized by sensitive dependence on initial conditions (where small changes can lead to vastly different outcomes), aperiodic long-term behavior (lacking repeating patterns), and topological mixing (where trajectories eventually spread throughout the entire state space) [12].

One major application is in pseudo-random number generation, which is vital for simulations, modeling, and cryptography [13, 14]. Additionally, chaotic maps are employed in image encryption techniques to ensure secure data transmission and enhance the confidentiality of sensitive information [15]. They also play a significant role in developing secure communication protocols and cryptographic algorithms [16]. Furthermore, they are integrated into chaos-based neural networks, thereby enhancing adaptability and performance in complex systems, such as time-series prediction and anomaly detection [17, 18]. Together, these applications highlight the versatility and importance of chaotic maps in advancing computer science.

Chaotic maps are widely used for pseudo-random number generation in both software, such as MATLAB, C programming, and LabVIEW, as well as hardware, including Arduino, Field-Programmable Gate Array (FPGA), Raspberry Pi, and LabVIEW [14]. There are two types of chaotic maps: continuous and discrete. Chaotic maps are typically described in n-dimensions, with most literature focusing on dimensions up to four. The application of chaotic maps in microcontrollers has been previously explored, particularly in random number generators [19, 20].

The research utilizes a one-dimensional chaotic map because a single parameter is sufficient to generate non-periodic, scrambled values. Additionally, the computational complexity of one-dimensional chaotic maps is simpler compared to two-dimensional or higher-dimensional maps. Chaotic map generators based on the Tent Map and Logistic Map can effectively produce pseudorandom sequences suitable for cryptographic applications. The implementation of these generators on FPGA devices results in reduced hardware usage and power consumption while achieving competitive operational speeds of 35.442 MHz and 26.134 MHz for the Tent Map and Logistic Map, respectively. It is important, particularly in applications where battery life is critical [21]. The randomness of the generated sequences is validated through National Institute of Standards and Technology (NIST) test suites, ensuring their suitability for secure application [21]. For the research, the logistic map is chosen due to its straightforward implementation and computational efficiency.

A logistic map can be defined as Eq. (1). The $x_n$ is the population value at the $n$-th iteration, and $r$ is the parameter that determines the population growth rate. The parameter $r$ affects the system's stability level. In general, the summary of $r$ values is provided in Table I. For $0 < r \leq 1$, the outcome will stabilize at 0. For $1 < r \leq 3$, the outcome will start moving from 0 to a certain $x_n$ value at $r$ = 3. For $3 < r < 3.57$, period-doublings will occur as $r$ increases toward 3.57. Table II illustrates this bifurcation. Lastly, for $r$ = 4, chaos reaches its maximum. However, its inherent properties remain, where both deterministic and stochastic characteristics are part of the chaotic map.

$$x_{(n+1)} = r x_n (1 - x_n). \tag{1}$$

### *C. Microcontroller Unit*

The microcontroller used in this electric bike rental system is selected based on its ability to execute the logistic map algorithm optimally. In the research, the variable influencing the algorithm is $x$. The initial value of $x$ is the 9-digit electric bike ID, represented

TABLE I
RANGE OF $r$ VALUES FOR THE LOGISTIC MAP.

| $r$ | Outcome | Dynamics |
|---|---|---|
| $0 < r \leq 1$ | Population dies off | Stabilizes to 0 |
| $1 < r \leq 3$ | Stable (fixed point) | Converges to a constant value |
| $3 < r < 3.57$ | Periodic oscillations | Bifurcation occurs |
| $3.57 \leq r < 4$ | Chaos, with periodic windows | Dominated by chaotic behavior |
| $r$ = 4 | Maximum chaos | Extreme sensitivity to initial conditions |

TABLE II
BIFURCATION OCCURRENCE IN THE LOGISTIC MAP FOR $3 < r < 3.57$ [12].

| $r$ Value | Periodic Oscillations |
|---|---|
| $r_1$ = 3 | 2 |
| $r_2$ = 3.449 | 4 |
| $r_3$ = 3.54409 | 8 |
| $r_4$ = 3.5644 | 16 |
| $r_5$ = 3.568759 | 32 |
| $r_\infty$ = 3.568759 | $\infty$ |

**Algorithm 1: Modified Logistic Map**

```
    input: parameter r, serial no x, n iterations
1   xi ← x
2   for i ← 1 to n do
3       xi ← r * xi * (1 - xi / 1000000000)
4       xi ← floor (xi)
5   end for
6   return xi
```

Fig. 2. Pseudocode of the modified logistic map algorithm to generate a 9-digit integer.

as an integer data type. The choice of a 9-digit format aims to ensure that the logistic map algorithm produces unique outputs in each iteration, minimizing the likelihood of periodic code repetition under operational conditions. As shown in Eq. (1), the values $x_n$ and $x_{n+1}$ in the logistic map are limited to a range between 0 and 1. To obtain an output range of 0 to 999,999,999 (nine-digit integers), it is necessary to adjust the logistic map equation, as depicted in Fig. 2.

However, 8-bit and 16-bit microcontrollers face limitations when implementing this algorithm, particularly regarding the maximum numerical values they can represent. The maximum integer value for an 8-bit microcontroller is 255, while a 16-bit microcontroller has 65,535. These bit-width constraints are insufficient to represent the 9-digit token, causing discrepancies between the token generated by the microcontroller and the one generated by the computer. Rounding errors during iterations further exacerbate the precision issues. A 32-bit STM32 microcontroller is selected to support operations involving large numerical values to address these limitations. This choice ensures accurate and consistent execution of the logistic map algorithm across both the microcontroller and the web-based information system.

The microcontroller unit is also connected to several input and output components, such as an LCD, a keypad, and a relay, as described in the previous section. The microcontroller receives its power supply from the electric bike's battery. Due to the difference in voltage levels, a buck converter and a voltage regulator are utilized to step down the voltage, ensuring compatibility with the microcontroller's operational requirements. The schematic diagram in Fig. A2 in Appendix illustrates the wiring of the microcontroller, along with the electronic circuits of the buck converter and the voltage regulator. The microcontroller acts as the central controller of the e-bike rental system. The voltage source is obtained from the e-bike battery by converting 48 V to 5 V using a buck converter. The 5 V source powers the LCD and keypad, while the microcontroller requires 3.3 V to operate. Therefore, a voltage regulator regulates the 5 V to 3.3 V, providing sufficient power to the embedded system.

If the entered token in the embedded system interface is correct, the microcontroller will activate a relay connected to the electric bike's main switch. Then, it allows the bike to operate. The wiring between the relay and the main switch is relatively simple, as the electric bike's electrical system operates entirely on Direct Current (DC).

### *D. Web-Based Information System*

The electric bike rental information system is built using the PHP programming language with the Laravel framework. To begin a rental, customers scan the unique QR code displayed on the electric bike, which redirects them to the rental webpage shown in Fig. 3. On the webpage, customers are required to select the rental duration, enter their name, and provide their phone number.

After completing these steps, customers are directed to a payment gateway to finalize the transaction. Once the payment is successfully processed, the system sends a confirmation message to the customer's phone number. This message includes a digital receipt and a unique rental token, as shown in Fig. A3 in Appendix. Each bike unit is associated with a distinct QR code, ensuring that rentals are independent and tokens cannot be used across different bikes. The system also offers a feature that allows customers to view their rental history.

Furthermore, the web-based information system offers a non-deposit strategy to customers, as they are

Cite this article as: A. Juniar, A. K. Permana, F. Suroso, G. M. Rahmah, and M. T. Anwar, "Developing embedded systems for e-bike rentals with logistic map-based token generation", CommIT Journal 20(1), 63–75, 2026.

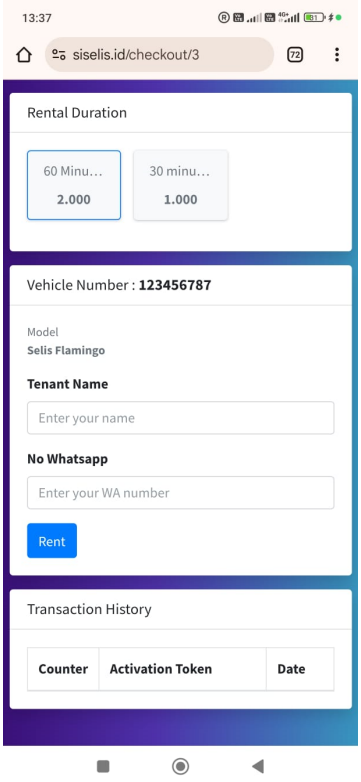



Fig. 3. Web-based e-bike rental interface.

not required to create an account when renting an electric bike. The electric bike rental model operates on a pay-as-you-go basis, which eliminates unused deposit funds after the rental period. Thus, this system provides convenience to customers by allowing them to pay only for the selected rental duration.

## III. Results and Discussion

The logistic map employed in the electric bike rental system is designed to exhibit chaotic behavior with periodic windows, restricting the range of r values to $3.57 \leq r < 4$. The initial value ($x$), represented as a 9-digit number, is selected arbitrarily. To ensure that each rental produces a unique token, the number of iterations ($n$) is varied using additional parameters. In this system, two key variables are introduced: rental duration and a counter. The rental duration, either 30 minutes or 60 minutes, is selected by the customer and used as an additive factor in determining the number of iterations for the algorithm. Similarly, the counter serves as an auto-incrementing variable that increases by one with every bike rental. The counter is initialized to zero when the system begins operation.

By incorporating both the rental duration and the counter as modifiers to the iteration count, the system

**Algorithm 2: Generate a Token**

**input:** parameter $r1$, $r2$, $r3$, serial no $x$, counter, duration
1 **function** generate_token (r1, r2, r3, x, counter, duration)
2 t1 ← modified_logistic_map(r1, serial_no, 50)
3 t2 ← modified_logistic_map (r2, t1, counter)
4 t3 ← modified_logistic_map (r3, t2, duration)
5 **return** t3
6 **end function**

Fig. 4. Pseudocode to generate a token in both embedded systems and web applications.

**Algorithm 3: Verify a Valid Token in the Embedded System**

**read from memory:** parameter $r1$, $r2$, $r3$, serial no $x$, counter
**input from keypad:** token
t30 ← generate_token(r1, r2, r3, x, counter, 30)
t60 ← generate_token(r1, r2, r3, x, counter, 60)
if (token == t30) then
  switch_on('30 minutes')
  counter ← counter + 1
  save_to_memory(counter)
else if (token == t60) then
  switch_on('60 minutes')
  counter ← counter + 1
  save_to_memory(counter)
else
  message('Invalid token')
end if

Fig. 5. Pseudocode to verify a valid token in the embedded system.

ensures that the generated tokens are unique for each transaction. This approach not only enhances operational consistency but also strengthens the security of the rental system by reducing the likelihood of token duplication or misuse. Figure 4 illustrates the token generator algorithm for generating unique tokens sequentially on both the embedded system and the web application side.

On the embedded system side, the activation token input by the user must be verified for its validity using an algorithm in Fig. 5. The communication between the web-based information system and the embedded system, facilitated by the chaotic map, eliminates the need for an Internet connection. Both systems independently generate matching tokens, enhancing robustness and reliability.

To evaluate the performance of the proposed system, algorithmic and server performance analyses, security

assessments, and robustness and reliability evaluations are conducted. Algorithm execution latency is quantitatively measured during the activation token generation and verification phases, while server response latency is evaluated at the communication layer, specifically within the end-user–server interaction channel, to characterize network-induced and processing delays. Then, security analysis is examined through avalanche testing to evaluate diffusion characteristics, and collision test is conducted to examine whether identical outputs could reoccur despite variations in the input parameters. In addition, robustness and reliability are systematically analyzed by formulating and evaluating three representative failure-case scenarios, enabling the identification of potential fault propagation mechanisms and system recovery behavior.

### A. Algorithm and Server Performance

For algorithm performance evaluation, the algorithm execution response time in generating and verifying activation tokens is first measured. The measurement is conducted on a Personal Computer (PC) equipped with an Intel Core i3-7020U CPU at 2.3 GHz, dual-core, 64-bit architecture, and 12 GB RAM. The token generation time is 0.68 ms, while the token verification time is 0.72 ms. Even though these results are faster than those obtained on the STM32, the data remains valid as a relative benchmark. They demonstrate the computational efficiency of the algorithm.

The second measurement is server response time. The communication layer between the end-user and the server is identified to derive the response time from the server. A web server hosted on a Virtual Private Server (VPS) with 2 CPUs, 2 GB of RAM, and 20 GB of disk storage is used. The operating system used is Ubuntu 24.04 LTS with NGINX 1.18 as the web server. PHP 8.3 with Laravel 11 is utilized for application development. Figure 6 illustrates the communication between end users, the web server, and the payment gateway. Four parameters are measured to determine the server response time: web request, snap token request, notification callback, and web response, along with WhatsApp notification. The response times obtained for those four parameters are 15 ms, 421 ms, 212 ms, and 200 ms, respectively. It is observed that among the four parameters, the snap token request represents a bottleneck. However, it measures the time required to complete several tasks: processing detailed transaction data, verifying the server ID through an Application Programming Interface (API) key, validating all data in the proper format, creating the token, and sending the token details back to the web server, all of which contribute to the bottleneck. Therefore, the time required to complete the request is relatively long.

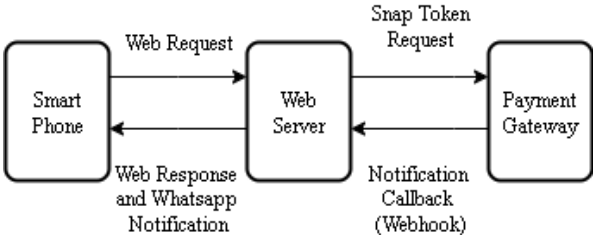



Fig. 6. Communication layer between the end user, web server, and payment gateway.

### B. Security Analysis

The security analysis consists of two tests: the avalanche effect test and the collision test. The avalanche effect is a property of cryptographic algorithms. It is performed to evaluate whether a slight change in the input variable (e.g., one bit or one digit) causes a significant and unexpected change in the output [22]. In the generate_token algorithm, the counter is an incrementing value, potentially leading to an avalanche effect. In an attempt to set up the avalanche test, the following scenarios are prepared:

1) Compare two different activation tokens based on similar inputs,
2) Define a 9-digit arbitrary random value for the bike ID,
3) Generate the first activation token using the generate_token function with input (bike ID, counter=50, duration=30). Here, 50 and 30 represent the counter value and rental minutes, respectively,
4) Generate the second activation token using generate_token function with input (bike ID, counter=51, duration=30),
5) Convert both activation codes from decimal values to binary values,
6) Employ the Hamming distance function to compare both tokens in binary,
7) Repeat the process for 1,000 trials with different bike IDs,
8) Calculate the average difference between the first and second tokens over the 1,000 trials.

The generate_token function output is in decimal form. Since nine digits represent a decimal value, the number of state spaces is $10^9$, equal to approximately $2^{30}$. Therefore, the Hamming distance function checks and calculates 30 bits of the first token against the second. To illustrate the conversion of a decimal token to binary, assume a 2-digit token for simplicity. The first and second tokens are 25 and 30, respectively. In binary, 25 is 11001 while 30 is 11110. Comparing both tokens in binary, 3 out of 5 bits are different, or 60% in percentage terms. The results of the avalanche

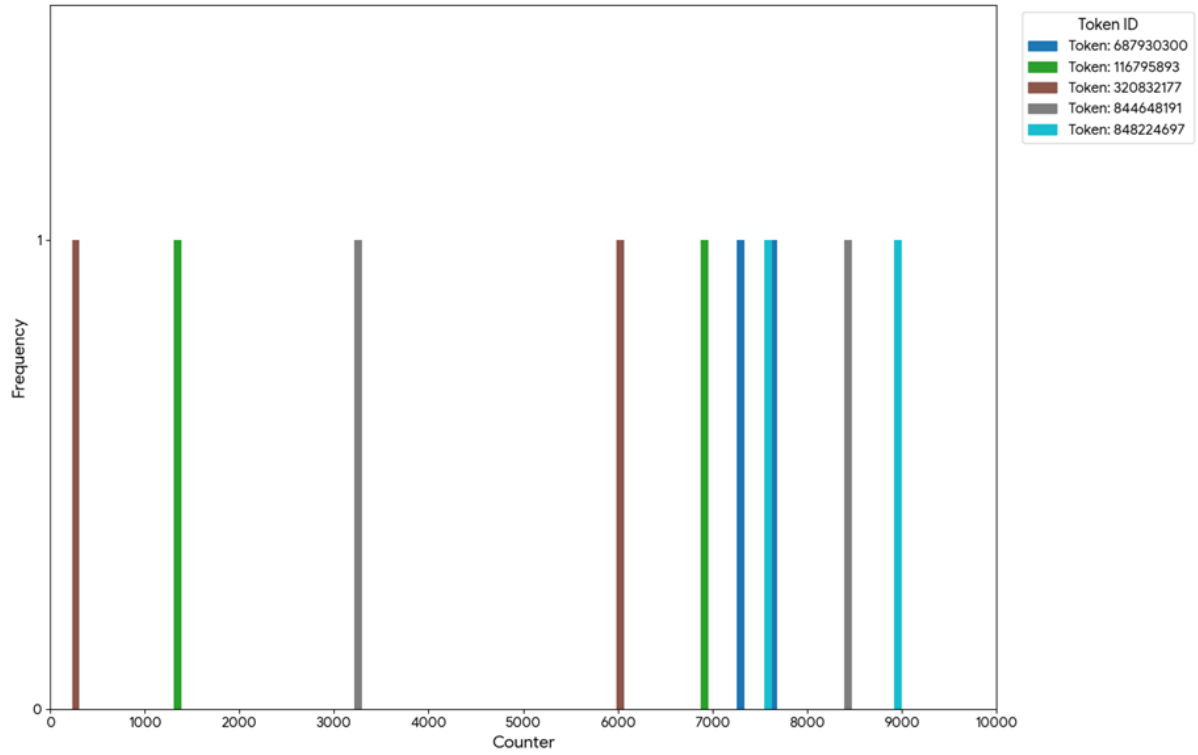



Fig. 7. The results of the collision test.

effect test on 1,000 token pairs show that a single-digit change in the input of the token generation function yields an average change of 49.81% in the output token bits. The test results indicate excellent diffusion characteristics (bit spread).

A collision test is conducted on the modified logistic map function to examine whether identical outputs can reoccur despite variations in the input parameters. The objective of this evaluation is to assess the uniqueness [23]. To run the test, the following scenarios are prepared:

1) Define one 9-digit bike ID at a constant value (e.g., 123456789),
2) Define the rent duration (e.g., 30 minutes),
3) Loop the generate_token function from counter 1 to 10,000,
4) Check for any duplicates among the activation tokens.

Based on the scenarios, five tokens appear twice at different counter numbers. Figure 7 illustrates the token duplication at varying counter values. The x-axis represents the counter, while the y-axis represents the duplication frequency. Although duplications occur, they are mostly at distinguishable distances, except for token ID 687930300. The duplication of token ID 687930300 is observed at counters 7,301 and 7,642. This scenario means that one bike with the same rental duration will experience five token duplications up to 10,000 usages.

These results exhibit a high collision rate since a 9-digit token should provide billions of possible combinations. Finding five duplicates in just 10,000 trials indicates that the algorithm's adequate output space is much smaller than it should be. A rounding operation is introduced in the modified_logistic_map function to limit the result to 9 digits. This rounding can quickly fall into a repeating cycle. However, this is the trade-off between security and hardware. A microcontroller usually provides lower bit tolerance than a PC. STM32 is employed due to its capability to generate up to 9-digit tokens, whereas Arduino or ESP32, operating at a lower bit-width than STM32, are limited to generating fewer than 9-digit tokens. Therefore, a lower bit-width microcontroller forces the token to round into smaller digits, resulting in a faster repeating cycle.

On the other hand, a PC can produce a token with billions of possibilities without worrying about rounding. Changing the microcontroller to a higher bit width may reduce the probability of duplicate tokens, but it makes the tokens longer (more than 9 digits). At this point, it is impractical to provide a prolonged activation token to keep the e-bike lit.

### C. Robustness and Reliability

Robustness and reliability are analyzed in two parts: failure case studies and scalability. Three failure case studies are identified in the research. The first is when the user enters an invalid token. If a user enters the wrong token, the system will not update the counter variable on the e-bike and will not turn on the ignition system. Only the correct and matching token provided by the server will activate the e-bike. The second one is when the user device's Internet connection is lost mid-transaction. No token will be generated if no payment is made during the connection loss. If payment is made successfully before the connection loss, the token and receipt message will be delivered once the connection is restored. The third one is when the bike's power is suddenly cut during verification. The embedded system will store the token's last state. The existing token acquired after payment will still match and can activate the e-bike.

The second part is scalability, which examines the system's capabilities if it is developed on a large scale. As discussed, each bike operates independently, with its ID number registered in the system. The system is inherently scalable because the main logic resides on the server. E-bikes do not affect each other because the tokens generated by each e-bike are independent, as the bike ID is an input to the logistic map algorithm. Although token collisions can occur between e-bikes, where two e-bikes generate the same activation token for a specific counter and rental duration, in practice, the end user cannot identify these two e-bikes.

This system is particularly suitable for small-scale environments. If a unit is not returned to its original dock, park management can still locate it. In a different scenario, if a unit is left at a random location, it can still be rented by the next customer. This situation reduces the need for a complex bike-sharing management system, such as those required in larger-scale implementations like cities [24–28].

## IV. Conclusion

The research presents a novel approach to an electric bike rental system designed to address the limitations of current rental systems, particularly in their implementation in small-scale environments. Utilizing the logistic map, which is part of chaotic maps, offers robustness and security for tokens generated in web-based and embedded systems. The proposed system can replace the docking-based approach that functions as a terminal for bike borrowing and returning, and a charging station for bikes after use. This system eliminates the typical need for human resources to operate rental counters.

In addition, users only need to utilize the built-in features available on their smartphones, specifically the camera with QR code scanning capability and the web browser. Users are not required to install any additional applications or register a new account for first-time use. The proposed system also applies a pay-as-you-go rental model, allowing customers to rent bikes without a deposit. Overall, the system offers strong usability due to its accessible interface, low learning curve, and secure operations. It is well-suited for small recreational environments without the need for extensive user training.

The research offers practical contributions and solutions for energy-efficient bike-sharing systems by leveraging the capabilities of chaotic maps as an intermediary between the two systems. It provides an alternative solution to existing traditional IoT-based systems. With this approach, the electronic device attached to each e-bike unit does not require continuous internet connectivity. Instead of maintaining persistent online communication, synchronization is achieved through a hash function generated by both server and e-bike unit. The embedded system does not continuously consume electrical energy for network transmission.

However, the system is limited to producing the same token at a particular counter due to hardware precision constraints. The generated tokens should remain unique and non-repetitive to ensure higher security and system robustness. Future research directions can focus on implementing a cryptographic algorithm capable of compensating for hardware limitations while maintaining security. Such approaches should preserve synchronization between two ends while minimizing repetition.

## Acknowledgement

The authors acknowledge the financial support of Badan Pengembangan Sumber Daya Manusia Industri (BPSDMI), Ministry of Industry, Republic of Indonesia, under the SPIRIT 2024 Grant, Decree No. 45/2024.

## Author Contribution

Conceived and designed the analysis, A. J. and A. K. P.; Collected the data, A. J., A. K. P., and G. M. R.; Contributed data or analysis tools, A. J. and F. S.; Performed the analysis, A. J., A. K. P., and M. T. A.; and Wrote the paper, A. J., A. K. P., and M. T. A.

## Data Availability

The data that support the findings of the research are available from the corresponding author, Ahmad Juniar, upon reasonable request. The dataset is not

publicly accessible due to intellectual property considerations and the security-sensitive nature of the system implementation.

## REFERENCES


[1] A. R. Al-Ali, R. Aburukba, A. H. Riaz, A. Al Nabulsi, D. Khan, S. Khan, and M. Amer, "IoT-based shared community transportation system using e-bikes," in *2021 5th International Conference on Smart Grid and Smart Cities (ICSGSC)*. Tokyo, Japan: IEEE, June 18–20, 2021, pp. 61–65.

[2] D. Croce, D. Garlisi, F. Giuliano, A. L. Valvo, S. Mangione, and I. Tinnirello, "Performance of LoRa for bike-sharing systems," in *2019 AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE)*. Turin, Italy: IEEE, July 2–4, 2019, pp. 1–6.

[3] S. Hameed, N. Junejo, Faraz Jafri, D. Rashid, and F. Shoaib, "Rent-a-cycle (Smart bicycle sharing service-IoT based)," *Journal of Robotics and Mechanical Engineering*, 2021.

[4] J. G. Ochoa, J. F. Mendieta, F. Astudillo-Salinas, D. Ochoa-Correa, and V. Iñiguez-Morán, "An IoT-based measurement device for e-bike tracking performance analysis," in *2024 IEEE Eighth Ecuador Technical Chapters Meeting (ETCM)*. Cuenca, Ecuador: IEEE, Oct. 15–18, 2024, pp. 1–6.

[5] J. Qian, M. Comin, and L. Pianura, "Data-driven smart bike-sharing system by implementing machine learning algorithms," in *2018 Sixth International Conference on Enterprise Systems (ES)*. Limassol, Cyprus: IEEE, Oct. 1–2, 2018, pp. 50–55.

[6] X. Wu, J. Lin, Y. Yang, and J. Guo, "A digital decision approach for scheduling process planning of shared bikes under Internet of Things environment," *Applied Soft Computing*, vol. 133, 2023.

[7] E. Collini, P. Nesi, and G. Pantaleo, "Deep learning for short-term prediction of available bikes on bike-sharing stations," *IEEE Access*, vol. 9, pp. 124 337–124 347, 2021.

[8] L. Lin, Z. He, and S. Peeta, "Predicting station-level hourly demand in a large-scale bike-sharing network: A graph convolutional neural network approach," *Transportation Research Part C: Emerging Technologies*, vol. 97, pp. 258–276, 2018.

[9] A. M. Joy, A. Ravi, J. Roy, G. V. Karthikeyan, and T. P. Rajan, "IoT enabled electronic security & remote monitoring solution for electric bikes," in *2022 IEEE International Power and Renewable Energy Conference (IPRECON)*. Kollam, India: IEEE, Dec. 16–18, 2022, pp. 1–6.

[10] S. J. S. Bakker, M. B. Ahmed, A. Djupdal, L. Natvig, H. Andersson, M. Jahre, and K. Fagerholt, "FOMOsim: An open-source simulator for rigorous analysis of micromobility planning problems," *Expert Systems with Applications*, vol. 264, pp. 1–18, 2025.

[11] A. Audikana, E. Ravalet, V. Baranger, and V. Kaufmann, "Implementing bikesharing systems in small cities: Evidence from the Swiss experience," *Transport Policy*, vol. 55, pp. 18–28, 2017.

[12] S. H. Strogatz, *Nonlinear dynamics and chaos: With applications to physics, biology, chemistry, and engineering*, 2nd ed. CRC Press, 2018.

[13] M. Francois, T. Grosges, D. Barchiesi, and R. Erra, "A new pseudo-random number generator based on two chaotic maps," *Informatica*, vol. 24, no. 2, pp. 181–197, 2013.

[14] R. B. Naik and U. Singh, "A review on applications of chaotic maps in pseudo-random number generators and encryption," *Annals of Data Science*, vol. 11, no. 1, pp. 25–50, 2024.

[15] S. Benaissi, N. Chikouche, and R. Hamza, "A novel image encryption algorithm based on hybrid chaotic maps using a key image," *Optik*, vol. 272, 2023.

[16] H. A. Abdullah and H. N. Abdullah, "A new chaotic map for secure transmission," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 16, no. 3, pp. 1135–1142, 2018.

[17] M. Ardalani-Farsa and S. Zolfaghari, "Synergy of chaos theory and artificial neural networks in chaotic time series forecasting," *International Journal of Applied Management Science*, vol. 3, no. 2, pp. 121–142, 2011.

[18] R. Barrio, Á. Lozano, A. Mayora-Cebollero, C. Mayora-Cebollero, A. Miguel, A. Ortega, S. Serrano, and R. Vigara, "Deep learning for chaos detection," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 33, no. 7, 2023.

[19] S. Çiçek, "Microcontroller-based random number generator implementation by using discrete chaotic maps," *Sakarya University Journal of Science*, vol. 24, no. 5, pp. 832–844, 2020.

[20] B. Stoyanov and T. Ivanova, "CHAOSA: Chaotic map based random number generator on Arduino platform," in *AIP Conference Proceedings*, vol.

2172, 2019.

[21] Izharuddin, O. Farooq, and M. Q. Rafiq, "Implementation of hardware efficient chaotic generators for signal security in portable systems," in *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies*. Udaipur, India: Association for Computing Machinery, March 4–5, 2016, pp. 1–8.

[22] W. Stallings, *Cryptography and network security: Principles and practice*, 8th ed. Pearson, 2020.

[23] J. Katz and Y. Lindell, *Introduction to modern cryptography*, 3rd ed. CRC Press, 2020.

[24] K. Luo, Y. Song, Z. Shi, Q. Yu, G. Wang, and Y. Shen, "A dynamic electric fence planning framework for dockless bike-sharing systems based on inventory prediction," *Computers & Industrial Engineering*, vol. 198, 2024.

[25] K. Hosseini, T. P. Choudhari, A. Stefaniec, M. O'Mahony, and B. Caulfield, "E-bike to the future: Scalability, emission-saving, and eco-efficiency assessment of shared electric mobility hubs," *Transportation Research Part D: Transport and Environment*, vol. 133, pp. 1–20, 2024.

[26] S. T. Jin and D. Z. Sui, "A comparative analysis of the spatial determinants of e-bike and e-scooter sharing link flows," *Journal of Transport Geography*, vol. 119, pp. 1–15, 2024.

[27] M. C. M. Silva, D. Aloise, and S. D. Jena, "On the simultaneous computation of target inventories and intervals for bimodal bike-sharing systems," *Transportation Engineering*, vol. 16, pp. 1–9, 2024.

[28] Q. Li, E. Zhang, D. Luca, and F. Fuerst, "The travel pattern difference in dockless micromobility: Shared e-bikes versus shared bikes," *Transportation research part D: Transport and Environment*, vol. 130, pp. 1–18, 2024.

## APPENDIX

The Appendix can be seen in the next page.

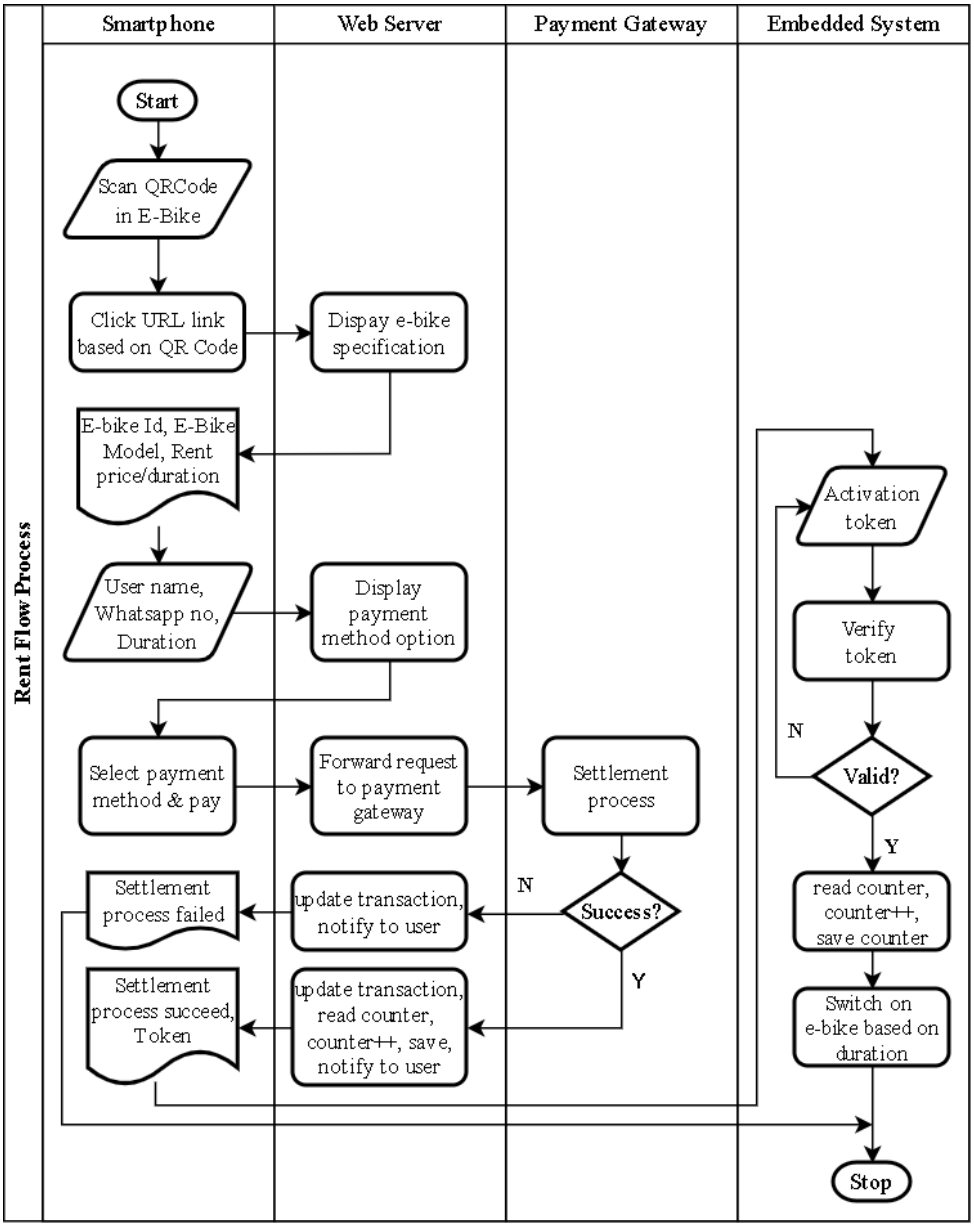



Fig. A1. Flowchart of the e-bike rental system.

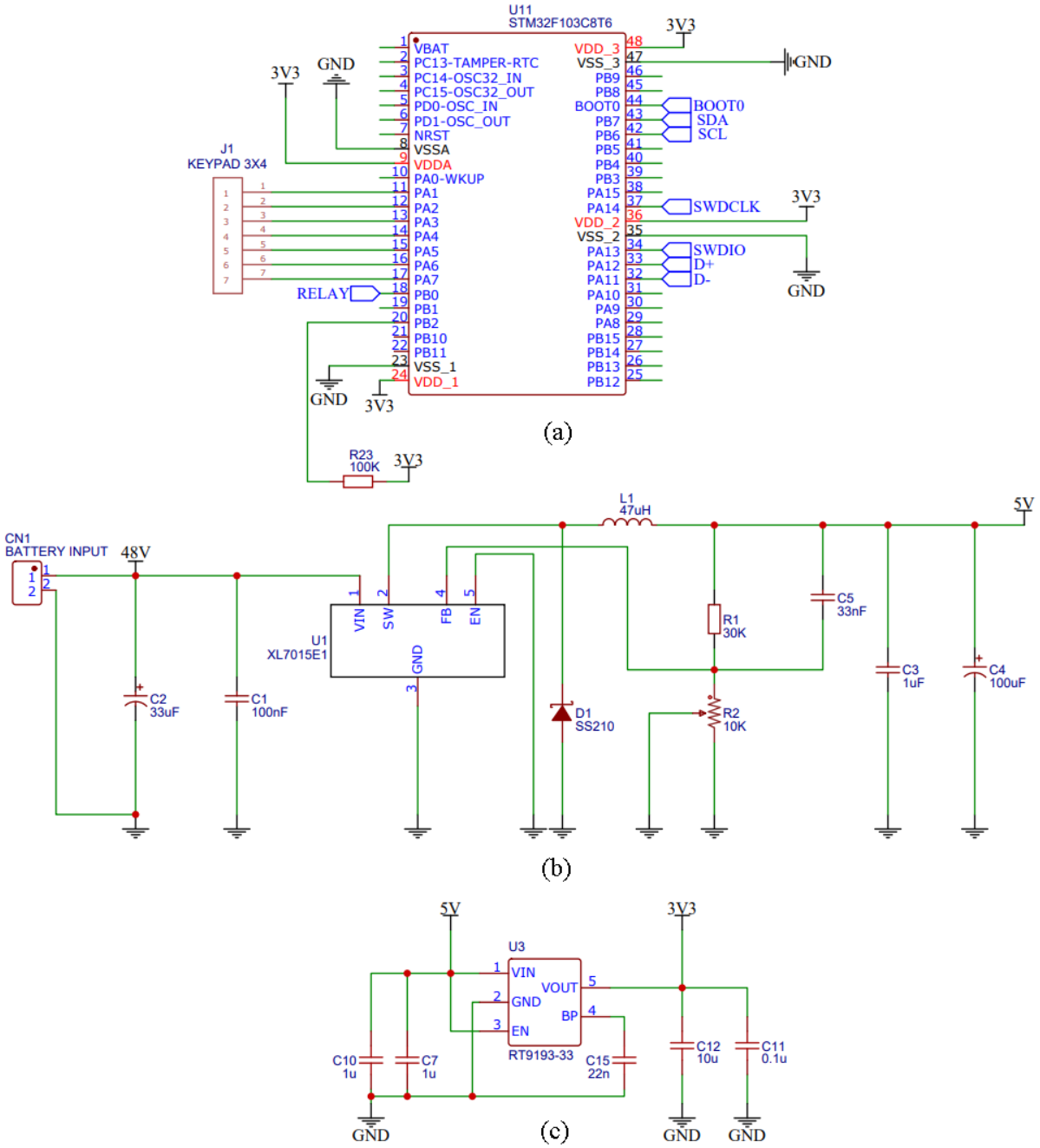
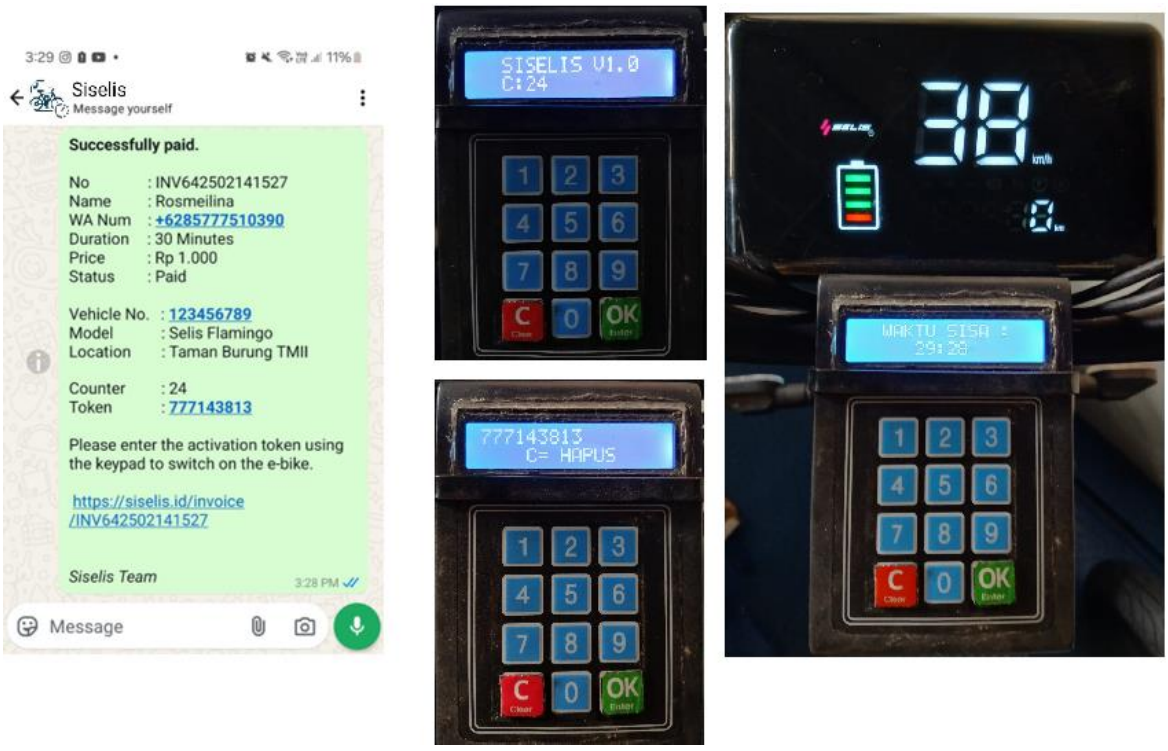



Fig. A2. The schematic diagram of (a) microcontroller, (b) buck converter, and (c) voltage regulator. Note: Ground (GND), Voltage In (VIN), and Voltage Out (VOUT).




(a) (b) (c)

Fig. A3. Implementation of the system: (a) message containing the digital receipt and activation token, (b) e-bike rental interface in standby status, (c) e-bike rental interface in active status with the e-bike powered up. Note: *hapus* means delete, and *waktu sisa* is remaining time.