Leveraging the Internet of Things to Enhance the Security of Two-Wheeled Vehicles

Muhammad Daffa Ramadhan[1], Muhammad Riza Rabbani[2], Muhammad Ihsan[3], Erna Hikmawati[4]

[1, 2, 3, 4] Faculty of Applied Science, Telkom University

mdaffaramadhan@student.telkomuniversity.ac.id

Submitted: May 25, 2025; Accepted: May 5, 2026

Abstract

This study aims to develop an Internet of Things (IoT)-based security system for two-wheeled motor vehicles to help reduce motorcycle theft. The system is built on an ESP32 microcontroller and a NEO-6M GPS module, integrated with an Android application named Motocurity via a Bluetooth connection. The system detects vehicle displacement based on changes in GPS coordinates, which are then converted into meters. When movement exceeds a predefined threshold, a buzzer sounds and a notification is sent to the Motocurity Android application. The system was developed using the Waterfall method, which comprises requirements analysis, system design, implementation, and deployment. Testing was conducted using black-box testing and usability testing. The black-box test results show that all features function properly, while the usability test involving 23 respondents yielded an average score of 82.8, which falls into the "Good" category, indicating that the system is easy to use. Therefore, the system is ready for implementation and can be further developed with longer-range connectivity.

Keywords: *Internet of Things, Vehicle Security, ESP32, Android Application, Usability*

1. Introduction

The motorcycle is the most commonly used mode of transportation in Indonesia [1]. The growing number of these vehicles directly contributes to traffic congestion in many areas, particularly around schools and centers of public activity. Beyond traffic problems, the increase in motorcycle ownership has been followed by a rise in theft cases [2]. Several manufacturers have equipped their motorcycles with built-in security systems such as alarms and handlebar locks; however, these technologies can still be easily bypassed by criminals [3]. The main weakness of current two-wheeled vehicle security systems is their low responsiveness and the absence of a detection mechanism that identifies suspicious movement in real time and immediately notifies the owner. Therefore, an additional security system that is smarter and more responsive is needed. One promising approach is the application of Internet of Things (IoT) technology, which allows vehicles to be monitored and controlled in real time through smart devices such as smartphones.

Technology is advancing very rapidly, especially since the emergence of Industry 4.0 [4], in which almost all systems are integrated with the internet. The internet no longer serves merely as a medium for exchanging information; it is also used for monitoring, automation, and remote control systems [5]. This technology enables physical devices such as motor vehicles to connect with smartphone-based applications, so that control and monitoring can be performed in real time [6]. In this context, microcontrollers such as the ESP32, GPS sensors, and Bluetooth or Wi-Fi connectivity are the main components for building IoT-based vehicle security systems [7]. Android offers advantages in leading-edge technologies, for example artificial intelligence (AI) combined with the *Internet of Things* (IoT). Building on this, many features can be developed and added to improve the security of two-wheeled vehicles, and these features can also make the system easier for its users to operate [8]. In addition, Android is *open-source*, so its features are continuously updated and can be further developed according to user needs [9].

In a previous study [2], an IoT-based motor vehicle security system was developed using an ESP32 microcontroller equipped with a fingerprint sensor and a Bluetooth connection,

allowing users to turn the alarm on and off at close range. However, that system did not integrate location-based tracking or the calculation of vehicle displacement using GPS.

This study addresses that limitation by developing a security system that not only detects movement but also dynamically calculates changes in the vehicle's position based on GPS coordinate data. The system combines an ESP32, a u-blox NEO-6M GPS module, and an Android application that serves as a real-time monitoring and alarm control tool. Vehicle displacement is calculated from the difference between coordinates converted into meters, and a notification is issued when the displacement exceeds a predefined threshold.

Accordingly, the main contribution of this study is the integration of GPS-based displacement tracking into an IoT-based security system for two-wheeled vehicles, an approach that has not been comprehensively addressed in previous studies.

2. Research Method

This IoT system was developed using the Waterfall method. The Waterfall method is a sequential development model in which each phase proceeds in order, like the flow of a waterfall [11]. Each phase must be completed before moving on to the next to maintain consistency [8]. This SDLC method was chosen because it is cost-effective and time-efficient for designing small-scale systems. The Waterfall method used in this study consists of four phases: requirements analysis, design, implementation, and deployment and maintenance. The details of the method are shown in Figure 1.

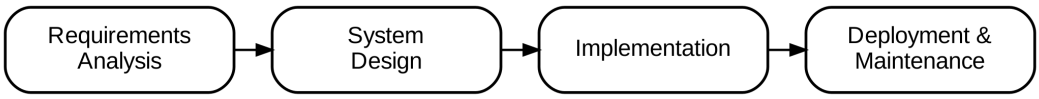


Figure 1. Waterfall Method (adapted from Wahid, 2020 [13])

2.1 Requirements Analysis

Requirements analysis is the first phase of the *Waterfall* method. It begins by identifying the requirements of the system to be implemented as an Android application and an IoT device [14]. The first step of the requirements analysis was to observe the security of motor vehicles that still rely on conventional keys, which are easy to break. Consequently, additional security that is responsive and works in real time is required.

The next step was to analyze the hardware and software required to develop the system. The main components are as follows:

1. An ESP32, which serves as the main microcontroller. The ESP32 is a powerful and affordable platform for developing IoT devices [15].
2. A u-blox NEO-6M GPS module, which detects movement and acquires coordinate data from satellites. Integrating the u-blox NEO-6M GPS is essential for improving precision and efficiency in navigation [16].
3. A buzzer, which sounds a warning when suspicious movement occurs.
4. An 18650 battery, which serves as a 5 V power source.
5. An Android-based application, which serves as the user's monitoring and control tool.

The third step was to determine the communication infrastructure requirements. The current system uses Google Firebase as a lightweight, real-time backend to manage data [17], and operates locally using a Bluetooth connection between the ESP32 and the *smartphone*. In the future, the system will be developed using a SIM800L module so that it can communicate over long distances without depending on a local connection [18]. The main features to be implemented include motion detection, measurement of the GPS coordinates converted into meters, a warning alarm via the buzzer and notifications in the Motocurity Android application, and short-range control via Bluetooth.

2.2 System Design

The results of the requirements analysis in the previous phase serve as the basis for the design to be implemented. The design uses the *Unified Modeling Language* (UML) approach. UML is commonly used in system development to visualize and document a system, and it provides several types of diagrams [19]. In this study, the design is described using the following diagrams.

a. Use Case Diagram

The use case diagram depicts the relationships between the actors and the system, as well as the features of the IoT-based motor vehicle security system. The system has two actors: the User and the ESP32. The User can register and log in, connect the device to the application via Bluetooth, turn the alarm on and off, and receive notifications when the ESP32 detects suspicious movement. In that case, the ESP32, which processes the data from the GPS module, sends an alert to the user's device and activates the alarm.

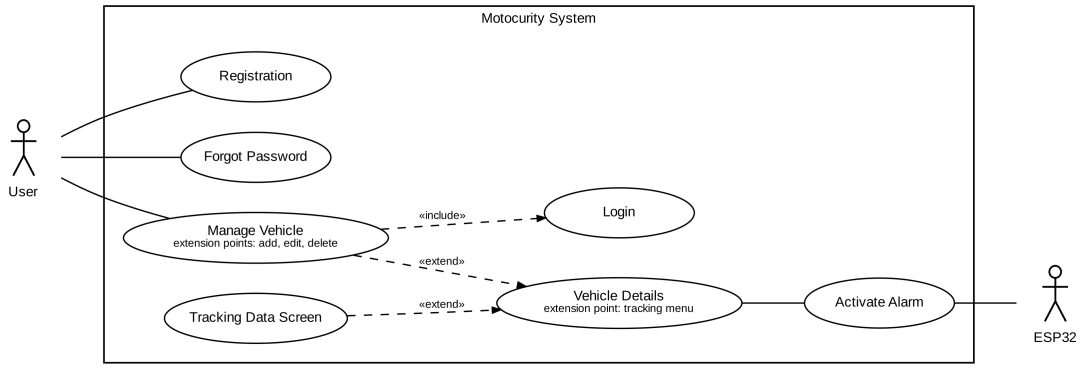


Figure 2. Use Case Diagram of the IoT Security System for Two-Wheeled Vehicles

b. Activity Diagram

The activity diagram visualizes the system workflow from the perspectives of both the user and the system (ESP32). It helps developers understand systematically how the process runs from start to finish. In this system, the main flow begins when the User opens the application and connects it to the ESP32 via Bluetooth. The ESP32 then sends a signal to the GPS module to obtain coordinate data received from satellites. If suspicious movement occurs, the ESP32 sends an alert to the application and the User receives a notification.

c. Tree Data Structure

A tree data structure is used to describe the hierarchical relationships among data organized in a tree-like structure [20]. This structure makes it easier to group the data of each component systematically.

In general, the system works as follows. The user opens the Android application and connects it to the ESP32 via Bluetooth. The ESP32 then activates the GPS module to read the vehicle's location coordinates. These coordinates are compared with the initial position; if the change exceeds the threshold (e.g., > 5 meters), the ESP32 activates the buzzer and sends a notification to the user's application. All user data are managed through Firebase. The system currently operates locally, but it can be extended to long-range communication using a GSM module such as the SIM800L.

2.3 Implementation

Implementation is the phase in which the designed system is realized into a working product according to its functions [14]. The motor vehicle security system was built using several technologies.

First, the Android application was developed in the Kotlin programming language with the Jetpack Compose framework. Kotlin is a language developed by JetBrains that can *compile* code for Android [21]. It was used to build the mobile application that serves as the main user interface. The application allows users to log in and register, connect to the ESP32 via Bluetooth, turn the alarm on and off, and receive notifications when suspicious movement occurs.

Second, the ESP32 acts as the brain of the hardware security system. The ESP32 was programmed using the Arduino IDE to control key components such as the NEO-6M GPS module, which obtains coordinates from satellites, and the buzzer, which produces a sound when suspicious movement occurs.

To power the hardware, the system uses a TP4056 module connected to an 18650 battery. The module not only supplies power from the battery to the ESP32 but also allows the battery to be recharged through a USB Type-C connector, so the device can be reused without replacing the battery.

Communication in this system relies on Bluetooth to connect the Android application and the ESP32. Through Bluetooth, the application can send commands such as enabling or disabling the alarm and can receive real-time notifications from the ESP32 when suspicious movement occurs.

In addition, Firebase is used as the system backend to store user data. Finally, the project was managed using GitHub as the version control system to facilitate development.

2.4 Deployment and Maintenance

Deployment and maintenance is the final phase of the *Waterfall* method for the IoT-based motor vehicle security system. This phase ensures that the system functions optimally and meets user needs.

Several important steps were carried out during deployment. First, the hardware was assembled: the ESP32, NEO-6M GPS module, buzzer, battery, and TP4056 were permanently mounted on a PCB. Next, the Android application was compiled and distributed as an APK, allowing users to install it and control the ESP32 via Bluetooth. Communication was established by *pairing* the ESP32 with the Android application. After testing, the device was declared ready for use.

System maintenance is performed to ensure that its functions and controls remain stable. For the hardware, maintenance includes checking the battery level, cable connections, and cleanliness of the modules. For the software, necessary fixes are applied. The system is designed to allow further development in the future, including the addition of features such as integration with a SIM800L module for long-range motorcycle tracking and Google Maps for visualizing the vehicle's location.

3. Results and Discussion

This section presents the results of the development of the IoT-based motor vehicle security system. The results are divided into two parts: the hardware side and the software (Android) side.

3.1 Hardware Results

a. ESP32

In this system, the ESP32 successfully served as the brain of the IoT device. It was able to process the data sent by the GPS module and forward it to the Android application. The ESP32 ran stably on the 5 V supply provided by the battery.

b. NEO-6M GPS Module

On average, the GPS module took about 5 minutes to acquire satellite signals in an open environment when the device was first powered on (cold start), depending on weather conditions and physical location. The main drawback of this module is its limited accuracy of ±5 meters, which may trigger false alarms when the vehicle has not moved but the coordinate data fluctuate (GPS drift).

c. Buzzer

The buzzer worked very well. Its loudness depends on the power it receives; the buzzer receives 5 V from the ESP32, and its sound is sufficiently loud to serve as an alarm.

d. TP4056 Module and 18650 Battery

The module worked as intended. The battery supplied power to the ESP32 through the TP4056 module without interruption, and it could be recharged through the same module. While the battery is charging, the red LED on the TP4056 lights up, and when the battery is fully charged, the blue LED lights up.

One of the main challenges in implementing this system is the limited GPS accuracy, especially in non-open environments (indoors or surrounded by buildings). The u-blox NEO-6M GPS module has an error margin of ±5 meters, which can trigger a *false alarm* when the vehicle has not moved but the signal fluctuates. To address this, the system applies a minimum movement threshold (e.g., 5 meters) before activating the alarm. In addition, the system reads location changes only at specific time intervals to avoid excessive notifications.

3.2 Software Results (Android Application)

3.2.1 User Interface

This section presents the user interface of the Android application that was developed.

a. Login and Registration Pages

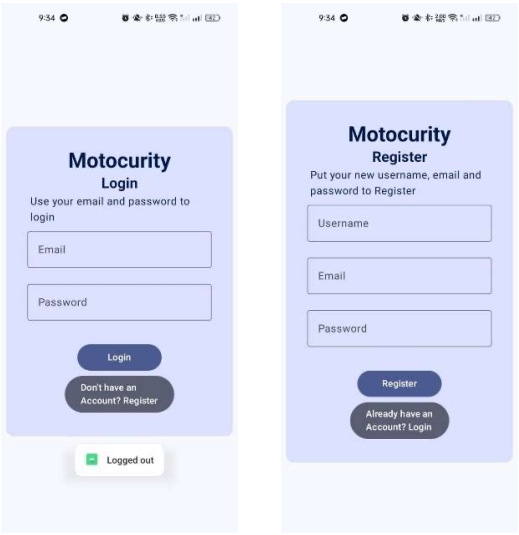


Figure 3. Login and Registration Pages of Motocurity

These pages allow users to register for or sign in to the Motocurity application. Users who already have an account can sign in through the Login page, while users who do not yet have an account can register on the Registration page.

b. Home Page

The home page of the Motocurity application (Figure 4) displays the list of motorcycles that the user has added through the Add Motorcycle page, which opens when the user taps the

"+" button in the lower-right corner. The user profile icon in the upper-right corner opens the user profile page.

c. User Profile Page

The user profile page (Figure 5) allows users to set their profile picture and edit their name.



Figure 4. Home Page

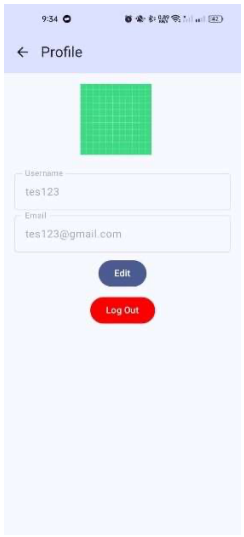


Figure 5. User Profile Page

d. Motorcycle Details Page

When the user taps one of the cards on the home page, the user is directed to the motorcycle details page (Figure 6). On this page, the user can enable or disable the alarm on the motorcycle and view its last recorded coordinates. The user can also edit or delete the motorcycle from this page.

e. Add Motorcycle Page

On the Add Motorcycle page (Figure 7), users can add a motorcycle to the list displayed on the home page of the application.



Figure 6. Motorcycle Details Page

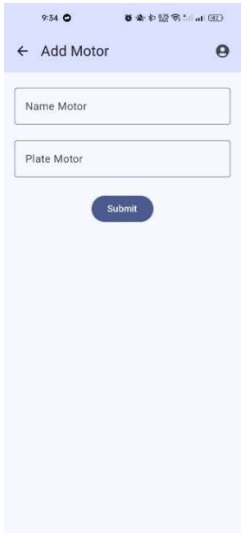


Figure 7. Add Motorcycle Page

f. Edit Motorcycle Page

On the Edit Motorcycle page (Figure 8), users can modify the motorcycle data that have been saved.



Figure 8. Edit Motorcycle Page

3.3 System Testing

System testing was conducted to ensure that the features designed for the Motocurity system function as intended. Two testing approaches were used: black-box testing, a method for testing a system without examining its internal details [22], and *usability testing*, which measures how satisfied users are when using the system [23].

3.3.1 Black-Box Testing

Black-box testing was performed by observing the system's output for given inputs. The test results are presented in Table 1.

Table 1. Black-Box Testing Results

No	Test Scenario	Expected Result	Status
1	ESP32 is powered on	The device turns on and is ready for use	Passed
2	The Android application connects to the ESP32	The connection status changes to "Connected"	Passed
3	The vehicle is moved	The buzzer sounds and a notification is sent to the application	Passed
4	The alarm is activated	The buzzer sounds when the threshold is exceeded	Passed
5	The motorcycle's location changes	The displacement distance from the initial point is sent	Passed
6	A notification appears in the application	The user receives a warning message	Passed

3.3.2 Usability Testing

Usability testing was used to assess how comfortable and easy it is for users to use the Motocurity system, based on the assessment criteria presented in Table 2. A total of 23 respondents completed a questionnaire consisting of 15 statements with five answer options on

a *Likert scale* ranging from "Strongly disagree" to "Strongly agree" [24]. The respondents consisted of university students and local residents.

Table 2. Usability Testing Assessment Criteria

Category	Average Score	Interpretation
Very Poor	0–50	The system is difficult to use
Poor	51–65	The system is usable but still has many obstacles
Fair	66–75	The system is reasonably comfortable to use
Good	76–85	The system is easy to use and stable
Excellent	86–100	The system is highly effective and provides an optimal experience

Each response was converted using the Likert weights shown in Table 3. The weighted scores for each statement were then summed and expressed as a percentage of the maximum possible score (number of respondents × 5). Based on the 15 statements answered by each respondent, the usability test yielded an average score of **82.8**, which falls into the **"Good"** category according to the interpretation scale in Table 2, indicating that the system is easy to use.

Table 3. Answer Categories and Weights

Answer Category	Weight
Strongly Disagree (SD)	1
Disagree (D)	2
Neutral (N)	3
Agree (A)	4
Strongly Agree (SA)	5

For visualization, the usability test results are presented in Figure 9.

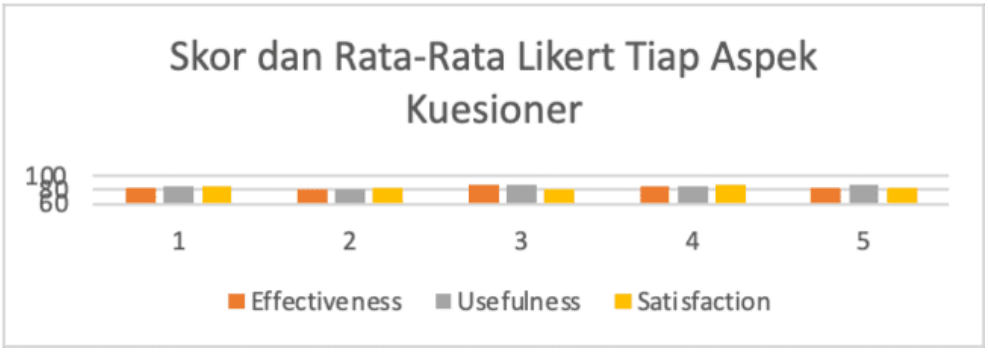


Figure 9. Questionnaire Results

The chart shows that all question aspects obtained an average score above 80, with the highest score in the *Satisfaction* aspect. This indicates that users are satisfied with the system's performance. The *Usefulness* and *Effectiveness* aspects also received good scores, meaning that the system is considered useful and effective in improving motor vehicle security.

4. Conclusion

Based on the results of this study, it can be concluded that an *Internet of Things* (IoT)-based motor vehicle security system using an ESP32 microcontroller and a NEO-6M GPS

module was successfully developed and integrated with an Android application named Motocurity. The system is able to send notifications to the application and detect displacement from the initial point based on the GPS coordinates, which are converted into meters. The buzzer successfully sounds when movement exceeds the predefined threshold.

Black-box testing showed that the features and functions work as designed, including the Bluetooth connection between the Android application and the IoT device, alarm activation, and motion detection. The usability test yielded an average score of 82.8, which falls into the **"Good"** category, indicating that the application is easy to use.

Overall, the IoT-based security system is ready to be implemented. Nevertheless, it can be further developed by adding a SIM card module, since the Bluetooth connection is limited in range, so that the system can be accessed from greater distances.

References

[1] A. Zahidi and H. Wulandari, "Statistik Transportasi Darat 2023," vol. 9, 2024.

[2] F. A. Aryatama and S. Samsugi, "Sistem Keamanan Kendaraan Bermotor Dengan ESP32 Menggunakan Kontrol Android," *SMATIKA Jurnal*, vol. 14, no. 01, pp. 167–181, Jul. 2024, doi: 10.32664/smatika.v14i01.1267.

[3] S. Albaar and M. Dzikrullah Suratin, "Rancang Bangun Sistem Monitoring GPS Tracker pada Kendaraan Sepeda Motor Berbasis IoT Menggunakan Platform Blynk dan Notifikasi Telegram," *Jurnal Teknik*, vol. 17, no. 1, 2024. [Online]. Available: www.jurnal.ummu.ac.id/dintek

[4] M. I. T. B. N. Sumadi, R. Putra, and A. Firmansyah, "Peran Perkembangan Teknologi Pada Profesi Akuntan Dalam Menghadapi Industri 4.0 Dan Society 5.0," *Journal of Law, Administration, and Social Science*, vol. 2, no. 1, pp. 56–68, Mar. 2022, doi: 10.54957/jolas.v2i1.162.

[5] A. Tahar, P. B. Setiadi, S. Rahayu, M. M. Stie, and M. Surabaya, "Strategi Pengembangan Sumber Daya Manusia dalam Menghadapi Era Revolusi Industri 4.0 Menuju Era Society 5.0."

[6] N. S. Lestari *et al.*, "Internet of Things Based Motorcycle Monitoring Application Development," in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Jun. 2021, doi: 10.1088/1742-6596/1933/1/012098.

[7] A. Boy Panroy Manullang *et al.*, "Implementasi NodeMCU ESP8266 dalam Rancang Bangun Sistem Keamanan Sepeda Motor Berbasis IoT," 2021. [Online]. Available: http://e-journal.stmiklombok.ac.id/index.php/jireISSN.2620-6900

[8] I. Agustine Cahyaningtyas and A. Stefanie, "Implementasi ESP32 CAM dan Kodular Berbasis Android untuk Monitoring Smart Garden," 2023.

[9] H. Bakri, "Pengembangan Aplikasi E-Library Berbasis Android di Jurusan Teknik Informatika dan Komputer Universitas Negeri Makassar," 2022.

[10] K. Z. Yonatan, H. S. Utama, and Y. Calvinus, "Aplikasi Kendali ESP32 Menggunakan Sensor Sidik Jari pada Ponsel sebagai Lapisan Keamanan," *AVITEC*, vol. 6, no. 1, p. 61, Feb. 2024, doi: 10.28989/avitec.v6i1.2049.

[11] T. Danardana Lokananta and I. C. Utomo, "Implementation of Cloud Computing on Website-Based Car Rental System at Dampit Trans Solo," 2025.

[12] A. F. Diansyah, M. R. Rahman, R. Handayani, D. D. Nur Cahyo, and E. Utami, "Comparative Analysis of Software Development Lifecycle Methods in Software Development: A Systematic Literature Review," *International Journal of Advances in Data and Information Systems*, vol. 4, no. 2, pp. 97–106, Sep. 2023, doi: 10.25008/ijadis.v4i2.1295.

[13] A. A. Wahid, "Analisis Metode Waterfall Untuk Pengembangan Sistem Informasi," Sekolah Tinggi Manajemen Informatika dan Komputer Sumedang, 2020. [Online]. Available: https://www.researchgate.net/publication/346397070

[14] G. Jimmy Tri Pradana and I. Saifudin, "Rancang Bangun Sistem Penyewaan Rental Mobil di Garuda Car Rental Berbasis Website Menggunakan Metode Waterfall."

[15] D. Hercog, T. Lerher, M. Truntič, and O. Težak, "Design and Implementation of ESP32-Based IoT Devices," *Sensors*, vol. 23, no. 15, p. 6739, Jul. 2023, doi: 10.3390/s23156739.

[16] "Sistem Navigasi Robot Mobil Pada Daratan Rata Menggunakan GPS Ublox Neo M6 V2 Untuk Peningkatan Presisi Dan Efisiensi."

[17] P. Megantoro *et al.*, "Instrumentation system for data acquisition and monitoring of hydroponic farming using ESP32 via Google Firebase," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 27, no. 1, pp. 52–61, Jul. 2022, doi: 10.11591/ijeecs.v27.i1.pp52-61.

[18] F. Nur, A. Wijaya, S. Noertjahjono, and Y. A. Pranoto, "Rancang Bangun Sistem Keamanan pada Sepeda Motor Menggunakan SMS Gateway Berbasis Mikrokontroller," 2020.

[19] H. Koç, A. M. Erdoğan, Y. Barjakly, and S. Peker, "UML Diagrams in Software Engineering Research: A Systematic Literature Review," in *The 7th International Management Information Systems Conference*, Basel, Switzerland: MDPI, Mar. 2021, p. 13, doi: 10.3390/proceedings2021074013.

[20] H. Jurnal *et al.*, "Implementasi Struktur Data Tree pada Web Blog sebagai Media Pembelajaran," *Jurnal Informatika dan Teknologi Komputer*, vol. 2, no. 2, pp. 206–212, Jul. 2022.

[21] K. Wasilewski and W. Zabierowski, "A comparison of Java, Flutter and Kotlin/Native technologies for sensor data-driven applications," *Sensors*, vol. 21, no. 10, May 2021, doi: 10.3390/s21103324.

[22] D. Yulistyanti, T. Y. Akhirina, T. Afrizal, A. Paramita, and N. Farkhatin, "Testing Learning Media for English Learning Applications Using BlackBox Testing Based on Equivalence Partitions," *Scope: Journal of English Language Teaching*, vol. 6, no. 2, p. 73, Apr. 2022, doi: 10.30998/scope.v6i2.12845.

[23] P. Sukmasetya, A. Setiawan, and E. R. Arumi, "Penggunaan Usability Testing sebagai Alat Evaluasi Website KRS Online pada Perguruan Tinggi," *Jurnal Sains dan Teknologi*.

[24] G. S. Mahendra and I. K. A. Asmarajaya, "Evaluation Using Black Box Testing and System Usability Scale in the Kidung Sekar Madya Application," *Sinkron*, vol. 7, no. 4, pp. 2292–2302, Oct. 2022, doi: 10.33395/sinkron.v7i4.11755.