International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Swarm of Mobile Robots for Security Surveillance Based on Android Smartphone and Firebase

Afdhal Kurniawan[1], Alexander A. S. Gunawan[2], Boby Hartanto[3], Aditya Mili[4], Widodo Budiharto[5]

Submitted: 06/05/2023 **Revised**: 13/07/2023 **Accepted**: 07/08/2023

Abstract: A swarm of mobile robots is an approach used to coordinate large numbers of mobile robots for accomplishing specific tasks. This study utilizes swarm robots in a security surveillance scenario. The research involves the development of an affordable mobile robot based on an IOIO board. To enable its integration into swarm robot systems, the robot must be enhanced with communication capabilities, and an algorithm must be implemented to coordinate multiple robots. The robots should be able to communicate with each other both locally and online. An OpenWrt-based router is employed as the local network communication backbone. Firebase, a cloud based IoT database service, is also utilized as a platform for online communication. The design of the swarm robot system allows a user to monitor and assign tasks to the system remotely. Once connected to a local network, the swarm robots can operate and coordinate autonomously. The mobile robot should have a camera capable of identifying people and detecting motion, making it suitable for security surveillance applications. The experiments successfully demonstrated the swarm robot system's effectiveness in performing security surveillance tasks.

Keywords: *Android, Firebase, Microcontroller, Swarm Robot*

1. Introduction

Robotic research is rapidly expanding and increasingly being incorporated into technological developments. One significant advancement in the field of robotics is the swarm robot system. The swarm robot system operates on the concept of multiple robots working together to achieve a given task, aiming to find the optimal solution. However, due to its potentially high development cost, swarm robot intelligence research often poses challenges. Nevertheless, there are opportunities to leverage the usability of Android smartphones as a powerful platform for robotics [1], [2]. Android smartphones stand out for their capability to perform high-level computation tasks and sensor-based data collection, making them an ideal choice for a robotics platform. Swarm robot intelligence focuses on how robots communicate, exchange information, and compute it to determine their next step toward achieving the primary goal. In this study, an experiment is conducted using an IOIO microcontroller board and an Android smartphone as the central robotic controller [3], [4]. The choice of IOIO in this research is driven by its affordability and utilization of a high-level Java API (Application Programming Interface). The Java API enables researchers to develop Android applications using the primary language for Android application development [5], [6]. Communication among the robots in this research is established through an OpenWrt-based router, which creates a local network for facilitating communication within the immediate vicinity. OpenWrt, a Linux firmware for embedded devices like routers, offers a fully customizable and modifiable filesystem [7]. The study aims to allow users to receive real-time images captured by the robots for monitoring purposes and to assign tasks to the robots. Existing literature introduces surveillance detection robots such as Pioneer-AT and Pioneer2-AT [8]. However, these robots lack a built-in sensor in their microprocessor, leading to higher development costs and slower computational capabilities than an Android-based robot. Each Android device, functioning as a robot controller, captures real-time images of its surroundings using the built-in camera and directly transmits them to a real-time database [9]. The research employs the Firebase real-time database as a platform for data storage. When data is sent to Firebase, it notifies the client, and the captured image is displayed on the user's screen. Similarly, user-assigned tasks are initially sent to Firebase, and the information is synchronized with the robots in the field. Firebase enables robots to respond promptly to changes in the database by providing simultaneous access to the same information, which determines their next step toward reaching the primary goal. As Firebase is a cloud-hosted real-time database, the robots must establish an Internet connection to retrieve information from the database [10].

[1]*Computer Science Department, School of Computer Science, Bina Nusantara University, Jakarta 11530, Indonesia*
Email: afdhal.kurniawan@binus.ac.id
[2]*Computer Science Department, School of Computer Science, Bina Nusantara University, Jakarta 11530, Indonesia*
Email: aagung@binus.edu
[3]*Computer Science Department, School of Computer Science, Bina Nusantara University, Jakarta 11530, Indonesia*
Email: boby.hartanto@binus.ac.id
[4]*Computer Science Department, School of Computer Science, Bina Nusantara University, Jakarta 11530, Indonesia*
Email: aditya.mili@binus.ac.id
[5]*Professor, Computer Science Department, School of Computer Science, Bina Nusantara University, Jakarta 11530, Indonesia*
Email: wbudiharto@binus.edu

2. Robotic Mechanics

One of the primary areas of focus in the study of robot mechanisms is kinematics, which pertains to the description of motion without considering factors such as mass, gravity, or external forces. Kinematics encompasses the mathematical representation of an object or group of objects regarding position, velocity, and acceleration. These parameters are expressed within a coordinate system defined by axes. In this study, a mobile robot serves as the fundamental framework for the robot's mechanism. The robot is equipped with a caster wheel and two actuators directly attached to the wheel, significantly contributing to its ability to change position[11]. Each actuator operates independently, exhibiting its velocity and acceleration. The caster wheel on a vertical axis governs the robot's direction, determining its trajectory. However, the mobile robot has unrestricted movement within its environment [12]. Since the robot's position cannot be measured directly, it must be integrated over time. By modelling the robot as a rigid object and performing kinematic transformations in a horizontal plane, we can calculate and determine its position within a coordinate system. This kinematic model involves two dimensions for the robot's position on the horizontal plane and one dimension for its orientation along the vertical axis, which is orthogonal to the horizontal plane [13], [14]. The robot's position can be specified by establishing the relationship between the plane's global reference frame and the robot's local reference frame, as shown in

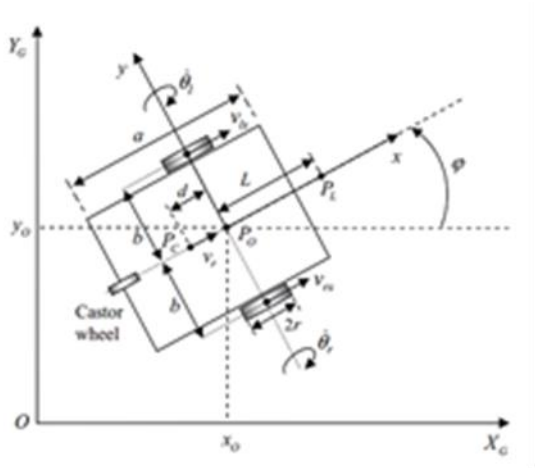

Fig. 1. Mobile robot coordinate frame [15]

3. Mobile Robot Design

Two issues must be addressed when discussing robot implementation: hardware and software design. Building robot hardware involves constructing the fundamental skeleton or robotic chassis that aligns with its main function and serves as the primary attachment point for all tools required for the robot to perform its functions flawlessly. Each tool paired with the robot has a specific role that contributes to the successful execution of the robot's actions. Before each tool becomes functional and fulfils its role, it is necessary to develop robot software that controls the robot's actions and determines how each tool collaboratively performs its function to achieve the primary goal.

3.1. Hardware Design

Constructing an Android mobile robot involves several key components, including a microcontroller, motor driver, and Android smartphone. These components are essential for establishing the core foundation of a mobile robot. The required components are detailed in Table 1.

Table 1. Component Mobile Robot

Component	Total
IOIO microcontroller board	1
Dual Motor Driver TB6612FNG	1
2WD (wheel-drive) chassis	1
DC Motor + Roda	2
Battery 5V	1
Bluetooth Dongle	1
Ball Caster	1
Android phone + holder	1

The wiring scheme of the Android mobile robot is depicted in Figure 2, as shown above. The Bluetooth dongle is connected to facilitate the connection and can be paired with an Android smartphone using the Bluetooth communication channel. A power bank provides electricity resources for the IOIO microcontroller board and DC motors. The required electric voltage is 5V with a current of 2 Amperes. Based on the previously specified wiring scheme, the final construction of the mobile robot used in this research is illustrated in Figure 3. An Android smartphone is the host controller for the IOIO board and its peripherals. The Android smartphone functions as a connector, receiving commands from another Android smartphone and translating them into inputs for interfaces understandable by the IOIO board. Additionally, the smartphone utilizes PWM (Pulse Width Modulation) streamed using the Java API provided by the IOIO microcontroller board. Furthermore, Android smartphones can serve as cameras, capturing and preprocessing images.

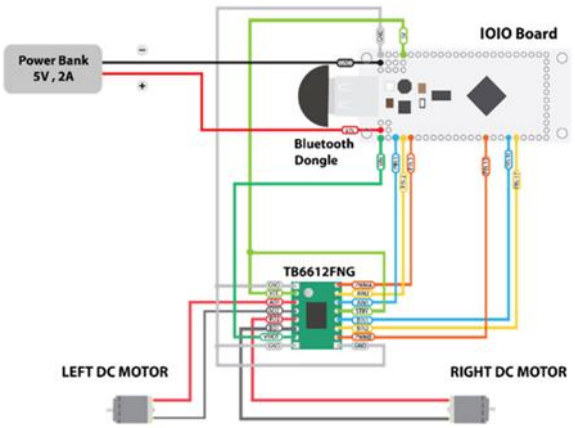

Fig. 2. Wiring scheme of mobile robot

Fig. 3. Side view of Android mobile robot

3.2. Software Design

The software engineering method employed for this project is Rapid Application Development (RAD). The application is developed on the Android platform, requiring knowledge of the XML data format and the Java programming language. Java is the primary language in the Android SDK for developing Android-based applications. XML is utilized for designing the application's user interface, with each component referenced by an XML tag in the Android application. Java serves as the application's interface controller, responsible for providing data to the application interface and performing computations in response to interface events like button clicks or data changes. The Android architecture follows a basic Model-View-Controller (MVC) design pattern by separating the view file (containing the user interface design) from the controller, ensuring clean and readable code, and avoiding mixing of user interface and controller logic, which can be common in pure Java GUI programming. In Java, components are referenced by their ID, which is generated each time a component is placed on the application interface. The application is built using the Android SDK, and for faster development, Google provides an Integrated Development Environment (IDE) called Android Studio. Android Studio offers specific features for coding Android applications, including Android programming code completion, SDK manager for managing installed SDKs, and the Firebase package utilized in this research. Additionally, it enables the emulation of Android devices with various firmware or hardware specifications, such as screen dimensions, memory, CPU type, and storage capacity. After compiling the application can be directly run on virtual devices emulated using the Android Debug Bridge (ADB), as illustrated in Figure 4.

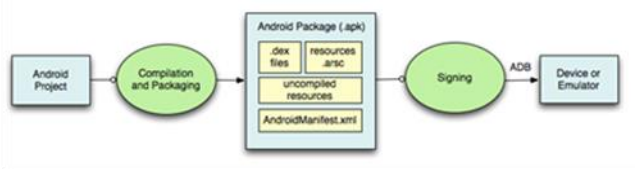

Fig. 4. Android build process

Android SDK will compile the provided code and build an application package based on its minimum SDK version configured with an apk extension, which can be installed on Android smartphones. There are various versions of SDK, and not all Android smartphones support all versions of SDK. The Android SDK version keeps developing along with the Android version, with improvements for better performance. The newest version of the Android SDK is SDK 26, which targets Android Oreo (version 8) as the platform. This research targets applications with a minimum SDK version of 23 that can run on a minimum Android version of 6.0 with the codename "Marshmallow."

3.3. Firebase

Firebase offers a real-time database and backend as a service. The Firebase real-time database provides APIs (application programming interfaces) that facilitate synchronization between clients and the data stored on the Firebase cloud. The company provides client libraries for seamless integration with Android, iOS, JavaScript, Java, Objective-C, Swift, and Node.js applications. Data stored in Firebase can also be accessed through its REST API, which can be bound to various JavaScript frameworks. The REST API leverages the Server-Sent Events protocol, an API for establishing HTTP connections to receive server push notifications. Data security is enforced through server-enforced security rules provided by Firebase. These security rules enable applications to authenticate users solely using client-side code. While client-side authentication is typically vulnerable and susceptible to hacking, Firebase offers robust data protection. The security rules provided by Firebase enable applications to authenticate and authorize users, controlling access to specific data and preventing unauthorized access. Firebase provides an authentication service called FirebaseAuth, allowing users to store credentials using an OAuth token obtained from third-party

platforms like Facebook, Twitter, Google, or GitHub. User credentials entered in the application are forwarded to the Firebase backend service for verification. The backend service responds with an authentication status, which is then relayed to the client. Firebase rules are written in a JSON format with a JavaScript-like syntax, encompassing four main types that govern data access and storage according to the defined rules. The rule specified above mandates that data written to "/robots/Status_Active" must be a number between 0 and 1. To determine whether a write operation is allowed or not, validating rule types are used. The significant distinction between validating rule types and write and read rule types is that validating rule types do not cascade. Cascade means that the rule will be applied to all children within the rule's scope. Since validating rules do not cascade, accurate evaluation of all validating rules within the relevant scope is required to allow a write operation. Firebase stores its data in JSON format. Every piece of data sent to Firebase is serialized into JSON format before storage. Consequently, retrieving data from Firebase needs to be deserialized before it can be utilized. Firebase supports storing nested data without any level limitation. Data retrieval in Firebase involves attaching an asynchronous listener to a data reference. By attaching a listener, any changes to the information in the database will instantly trigger related events in all connected clients listening to that data. This event-based triggering mechanism eliminates the need for constant polling or implementing pull-to-refresh controls in applications. The robot's status is used to validate whether an Android smartphone acting as the controller can send tasks and retrieve captured images. Only an Android device with the controller role and an "Active" robot status can send tasks and retrieve image data from a robot in the field. Additionally, each mobile robot frequently updates its direction. The data structure used and stored in Firebase's real-time database is specified in Figure 5.

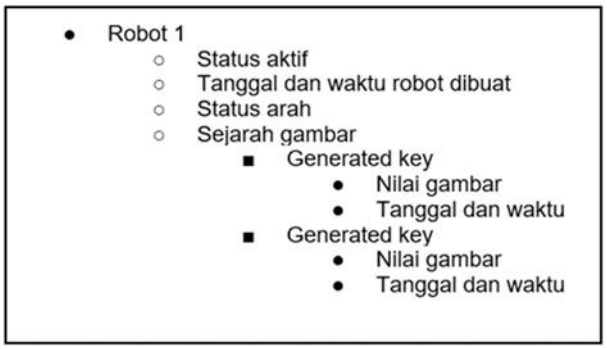

Fig. 5. Firebase data structure used and stored Firebase data structure used and stored.

Android smartphone sends a command to the IOIO board to control the motors and execute a differential drive on the robot, changing its direction, the Android smartphone updates its direction in Firebase. This update informs all clients connected to the Firebase database that the robot's direction is being changed. As both the status and direction are updated in Firebase, whenever an Android smartphone with the controller role submits a task, it is directly stored in the cloud database. This update triggers notifications to all robots, informing them about the new task. The overall process is depicted in Figure 6.

Fig. 6. Submitting Tasks to Robots

After a task is given, all robots will communicate together and work cooperatively to find the best solution for solving the given task. All robot communication is performed separately between the Android Camera and Android Controller. Robots on the field will communicate with each other using the local area network, which is built using OpenWrt. OpenWrt will serve as a communication medium and a server, as detailed in the following subsection.

3.4. Other Recommendations

OpenWrt is a Linux-based distributed system commonly embedded in devices like routers. One of the key advantages of OpenWrt is its open-source nature, allowing for complete modification of the filesystem to suit device requirements. Configuration of the filesystem can be accomplished using the provided command-line interface or web interface. LuCI is a web interface specifically designed to simplify the configuration of the OpenWrt system, providing developers with a free, clean, extensible, and user-friendly solution without the need for manual command-line coding. LuCI adopts a Model-View-Controller (MVC) web framework and is implemented in the Lua programming language. The interface of LuCI is showcased in Figure 7. LuCI's interface is structured into distinct parts, including models and views, and utilizes object-oriented libraries and templating. This design approach ensures enhanced performance, a reduced installation size, faster runtimes, and improved system maintainability.

4. Experiment Result

In order to evaluate the developed swarm robot, experiments were conducted to measure the time it takes for image data to be sent from the Android device operating in client mode, received, and stored in Firebase's real-time database. Additionally, the experiment measures the duration for all

servers to retrieve the new image data. The results of the firebase experiment are presented in Table 2.

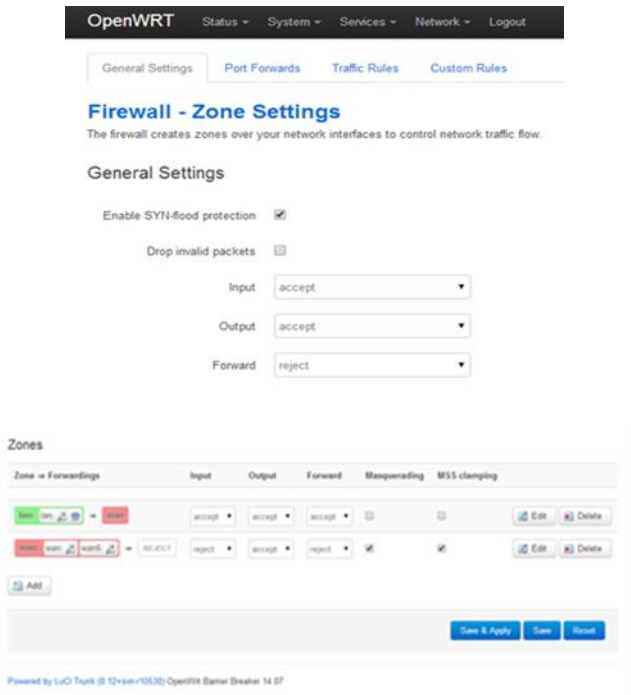

Fig. 7. LuCI interface

Table 2. Result of Firebase Experiment

No	Send Time	Receive Time
1	16:15:37.601	16:15:40.734
2	16:19:38.065	16:19:39.942
3	16:20:34.361	16:20:38.029
4	16:21:25.503	16:21:28.308
5	16:22:08.826	16:22:13.199

In the server mode, the Android device can send commands to the Android device operating in client mode, thereby controlling the robot's movement. The robot can perform four main movement types: forward, backwards, left, and right. The results of the movement experiment are presented in Table 3. Considering its precision, the robot's speed can be calculated by dividing the distance between the start and end points by the time it takes to reach the goal area. Qualitative experiments have been conducted with the robot to evaluate its performance in executing tasks sent by the Android server. These experiments assess the robot's ability to complete assigned tasks and achieve desired objectives successfully. The evaluation criteria include factors such as task completion rate, accuracy of movement, responsiveness to commands, and overall efficiency in executing the assigned tasks. The qualitative experiments provide valuable insights into the robot's performance, highlighting its capabilities and limitations when performing tasks instructed by the Android server. The results of these experiments contribute to the understanding of the robot's functionality and assist in identifying areas for improvement in task execution, responsiveness, and overall performance.

Table 3. Result of Movement Experiment

Movement	Average Speed	Average Precision
Straight forward	37.5 cm/s	±11 cm
Straight backwards	35.29 cm/s	±10 cm
Left	1240 / s	00
Right	1380 / s	00

5. Conclusion

In this experiment, an affordable robot based on the IOIO microcontroller was successfully constructed, enabling communication over the Internet using the Firebase Real-Time Database. The results of the experiments provide valuable insights and lead to the following conclusions. The first experiment revealed a short delay between when the image data is sent and when it is displayed on the server. To enhance this aspect future improvements could involve resizing the image data to reduce the duration of data transfer. The second experiment demonstrated that rotational movement was more precise than linear. To enhance the precision of linear movement, the attachment of an encoder to monitor the robot's wheel movement can be implemented. The average speed of the robot in the linear movement was approximately 36.4 cm/s, while the rotational movement had a speed of approximately 131 degrees per second. These findings provide valuable insights into the performance and capabilities of the developed robot. The experiment highlights potential areas of improvement and paves the way for further research and enhancements in robot design, communication, and movement precision.

References

[1] N. Oros and J. L. Krichmar, “Smartphone Based Robotics: Powerful, Flexible and Inexpensive Robots for Hobbyists, Educators, Students and Researchers,” Irvine, USA, 2013. [Online]. Available: https://pdfs.semanticscholar.org/1e4f/371b9509dac7b

649a473a0b95c39f0bfd63a.pdf

[2] S. Wilson et al., “Pheeno, A Versatile Swarm Robotic Research and Education Platform,” IEEE Robot Autom Lett, vol. 1, pp. 884–891, 2016.

[3] S. Monk, Making Android Accessories with IOIO, 1st ed. Sebastopol, CA 95472: O’Reilly Media, Inc., 2012.

[4] S. Gobel, R. Jubeh, S.-L. Raesch, and A. Zundorf, “Using the Android Platform to control Robots,” RiE 2011, 2011, [Online]. Available: http://www.innoc.at/fileadmin/user_upload/_temp_/RiE/Proceedings/65.pdf

[5] E. B. B. Gyebi, M. Hanheide, and G. Cielniak, “Affordable mobile robotic platforms for teaching computer science at African universities,” 2015.

[6] F. Arvin, J. Espinosa, B. Bird, A. West, S. Watson, and B. Lennox, “Mona: an Affordable Open-Source Mobile Robot for Education and Research,” J Intell Robot Syst, vol. 94, no. 3–4, pp. 761–775, Jun. 2019, doi: 10.1007/s10846-018-0866-9.

[7] C. E. Palazzi, M. Brunati, and M. Roccetti, “An OpenWRT solution for future wireless homes,” in 2010 IEEE International Conference on Multimedia and Expo, IEEE, Jul. 2010, pp. 1701–1706. doi: 10.1109/ICME.2010.5583223.

[8] J. Azeta et al., “An Android Based Mobile Robot for Monitoring and Surveillance,” Procedia Manuf, vol. 35, pp. 1129–1134, 2019, doi: 10.1016/j.promfg.2019.06.066.

[9] Y. Yan, S. Cosgrove, E. Blantont, S. Y. Ko, and L. Ziarek, “Real-Time Sensing on Android,” in Proceedings of the 12th International Workshop on Java Technologies for Real-time and Embedded Systems, New York, NY, USA: ACM, Oct. 2014, pp. 67–75. doi: 10.1145/2661020.2661026.

[10] L. Moroney, “The Firebase Realtime Database,” in The Definitive Guide to Firebase, Berkeley, CA: Apress, 2017, pp. 51–71. doi: 10.1007/978-1-4842-2943-9_3.

[11] R. Raj and A. Kos, “A Comprehensive Study of Mobile Robot: History, Developments, Applications, and Future Research Perspectives,” Applied Sciences, vol. 12, no. 14, p. 6951, Jul. 2022, doi: 10.3390/app12146951.

[12] R. Siegwart and I. R. Nourbakhsh, Introduction to Autonomous Mobile Robots. USA: Bradford Company, 2004.

[13] C.-L. Shih and L.-C. Lin, “Trajectory Planning and Tracking Control of a Differential-Drive Mobile Robot in a Picture Drawing Application,” Robotics, vol. 6, no. 3, p. 17, Aug. 2017, doi: 10.3390/robotics6030017.

[14] G. Dudek and M. Jenkin, Computational Principles of Mobile Robotics. Cambridge University Press, 2010. doi: 10.1017/CBO9780511780929.

[15] K. Shojaei, A. M. Shahri, A. Tarakameh, and B. Tabibian, “Adaptive trajectory tracking control of a differential drive wheeled mobile robot,” Robotica, vol. 29, no. 3, pp. 391–402, 2011, doi: 10.1017/S0263574710000202.

[16] Dr. Sandip Kadam. (2014). An Experimental Analysis on performance of Content Management Tools in an Organization. International Journal of New Practices in Management and Engineering, 3(02), 01 - 07. Retrieved from http://ijnpme.org/index.php/IJNPME/article/view/27

[17] Sheeba, T. B. ., Hemanth, S. V. ., Devaraj, V. ., Arularasan, A. N. ., & Gopianand, M. . (2023). Digital Hash Data Encryption for IoT Financial Transactions using Blockchain Security in the Cloud. International Journal on Recent and Innovation Trends in Computing and Communication, 11(4s), 129–134. https://doi.org/10.17762/ijritcc.v11i4s.6316