Performance Analysis of Android-Based Smart Agriculture Monitor and Control Applications

Helmy[1], Fenny Rahmasari[2], Arif Nursyahid[3], Thomas Agung Setyawan[4], Ari Sriyanto Nugroho[5]

***Abstract*—The ever-evolving digital era leads to an industrial revolution in the internet of things (IoT)-based smart agriculture and smart farm. Of many uses is the use of an Android-based app that monitors and controls parameters in the cultivation process in this digital era. An unstable internet connection can interfere with the monitoring process. For this reason, a system integration into a single app running even in an offline condition is needed; therefore, the user can monitor and control the Android-based smart agriculture app in two modes, namely online and offline. A performance analysis is also necessary to know the app's reliability in sending and receiving data. This system integration used two modes of operation, i.e. online and offline, wherein the online mode, the app will communicate with the server when connected with the internet using representational state transfer application programming interface (REST API). Meanwhile, the app will communicate directly with the system through a local access point in the offline mode. This app interacts with the system with the MQTT protocol where the app acts as an MQTT client. The performance analysis was conducted in the black box test, load activity test, and app performance test from the Android profiler. The acquired test from the app functionality test (black box) showed that the user could monitor and control the smart agriculture in online and offline mode through the app. The average load time for all the activities was 3.507 seconds with a network bandwidth of 4.54 Mbps. At the same time, the average load time in a network bandwidth of 35.35 Mbps was 1.4 seconds. The system performance test indicated the app was relatively light as the CPU usage for the app was 31%, with a memory usage of 453.8 MB.**

***Keywords*—Android Application, Performance Analysis, IoT, Smart Agriculture.**

I. Introduction

The agricultural sector is the most promising sector due to population growth, tourism, infrastructure, e-commerce, and retail sectors [1]. According to the December 2019 edition of a socioeconomic data report from Statistics Indonesia, there was a rise in four farmer exchange rates by 0.35% from 104.6 to 104.10 [2].

Due to the development of the digital era, Agriculture 4.0 was born and the smart farming industry was improving. This proliferation will help modify and guide the current agriculture system to support the development and ensure food security implemented with the internet of things (IoT) [3]. IoT integration in the agricultural sector can help farmers tending their agricultural land without manually using a particular device, so they can reduce the time needed to do the same task [4]. Monitoring and controlling can be done remotely from a website or an Android-based app [5]. This research used Android-based apps due to the significant number of smartphone users, especially Android users [6], [7].

Research on monitoring and controlling Android-based smart agriculture has already been conducted. One of those research is the performance of the Android app on monitoring and controlling urban farming in the 4G network [8]. The performance measurement from an Android app as a monitor and controller of an urban farming system was done by measuring the response time through the 4G network. The estimated response time was around 4 - 9 seconds [9]. The app testing was done on the 4G network to measure the app’s performance in controlling urban farming in several locations. Other research has also been conducted to detect the growth of plants by using AGRO-TECH to control several sensors. The notification systems were short message services (SMS) containing weekly reports and pop-ups [10]. Next, research designed a mobile app to monitor the hydroponic monitor. This research was intended to monitor the weather and nutrition level change since the plants’ transpiration and absorption employ the IoT. This system used a light-dependent resistor (LDR) sensor, temperature, and humidity sensor to monitor the weather; a pH sensor and electrical conductivity (EC) sensor to measure the water quality; and an ultrasonic sensor to measure the water height. The app used was MIT App Inventor, an online platform designed to develop mobile apps [11].

Based on the previous research, none has produced an Android-based monitoring and controlling smart agriculture system with online (connected to the internet) and offline (no internet connection) mode along with the performance analysis. The online mode of this monitoring and controlling system can help users to conduct remote monitoring and controlling. In contrast, the offline mode can help solve the internet or other issues. This paper is arranged into four parts. Part I is the introduction, part II will discuss the design. Part III will explain the result and analysis and part IV presents the conclusion.

II. Methodology

The methodology used for this research was the waterfall methodology. Waterfall methodology is a classic methodology where the system is linear—the output of the previous step is

[1,3,4,5] Politeknik Negeri Semarang, Jl. Prof. H. Soedarto, S.H., Tembalang, Semarang 50275 (tlp: 024 7473417, 7499585; fax: 024 7472396; e-mail: [1]helmy@polines.ac.id, [3]arif.nursyahid@polines.ac.id, [4]thomas@polines.ac.id, [5]ari.sriyanto@polines.ac.id)

[2] Electrical Engineering, Politeknik Negeri Semarang, Jl. Prof. H. Soedarto, S.H., Tembalang, Semarang 50275 (tlp: 024 7473417, 7499585; fax: 024 7472396; e-mail: [2]fenyyrahma@gmail.com)

[Received: 23 December 2021, Revised: 15 January 2022]

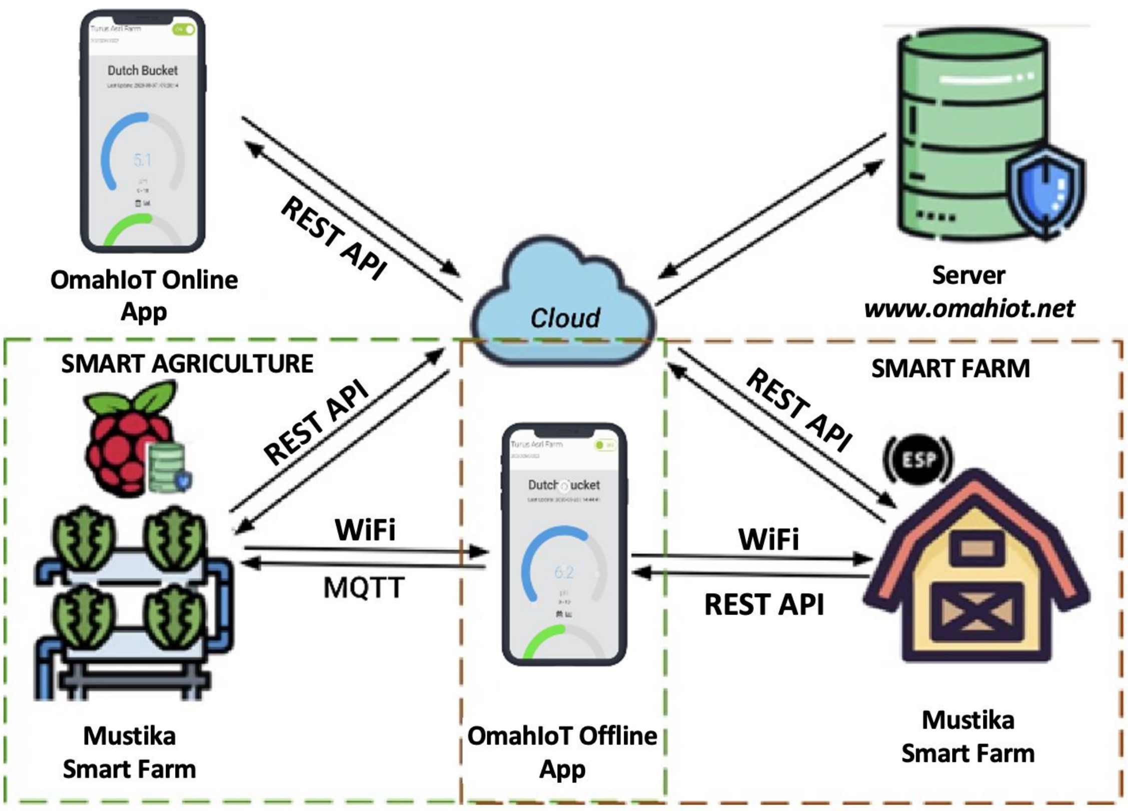


Fig. 1 System integration diagram.

used as input for the next step. The design and testing of this research began with knowing the user’s needs and continued with the planning, modeling, construction, and deployment stage. After the software was deployed, the software would still be provided with an ongoing maintenance if needed.

Tool and material, either in the form of software or hardware, were cataloged in the planning stage. In addition, work plan and schedule were made at this stage. The modeling phase of this study contained the entire design of the system, including the system integration diagram shown in Fig. 1, the design of the software model for the OmahIoT application, the OmahIoT application use-case diagram, the OmahIoT application subsystem use-case, the design of the database in the form of an entity-relationship diagram (ERD), and the user interface (UI) design of the OmahIoT Android application. The construction stage was started by working on the OmahIoT Android application with Android Studio IDE software using three stages, namely the execution stage, testing stage, and debugging stage.

The testing stage contained data retrieval that was conducted along with the system-wide testing. On the application side, testing was carried out with the black box method; the data was taken in the form of incorrect or missing functions, interface errors, errors in data structures or external database access, performance errors, initializations, and termination errors.

III. Design

The integration of smart agriculture monitoring, control systems, and an Android-based smart farm is shown in Fig. 1. The Android-based system in this study utilized two modes, namely online and offline mode. Online mode indicates that android applications will interact with the system through the server when connected to the internet using the representational state transfer application programming interface (REST API) method. The server used was called *www.omahiot.net*. Offline mode means that Android applications will interact directly with the system through a local device connection (local access point) for smart farms. Smart agriculture interacted with the system through the message queuing telemetry transport (MQTT) protocol where the Android smartphone acts as an MQTT client.

In this application, there were three actors, namely farmers, guests, and admins/developers. In monitoring and controlling, farmers were given two modes, namely online and offline. Both modes could run according to the manager’s internet connection at the site. Guests are allowed to do a monitoring in online mode only. Monitoring data were provided by the webserver, which were then displayed on the Android user. The admin/developer served as a software maintainer and application developer.

The use-case diagram on smart agriculture can be seen in Fig. 2. Actors could monitor plant parameters, set plant parameter thresholds, and received notifications in case of threshold changes.

The advanced application design applied splitting into subsystems method. Table I describes the use-case subsystem of the OmahIoT application. The subsystem consists of

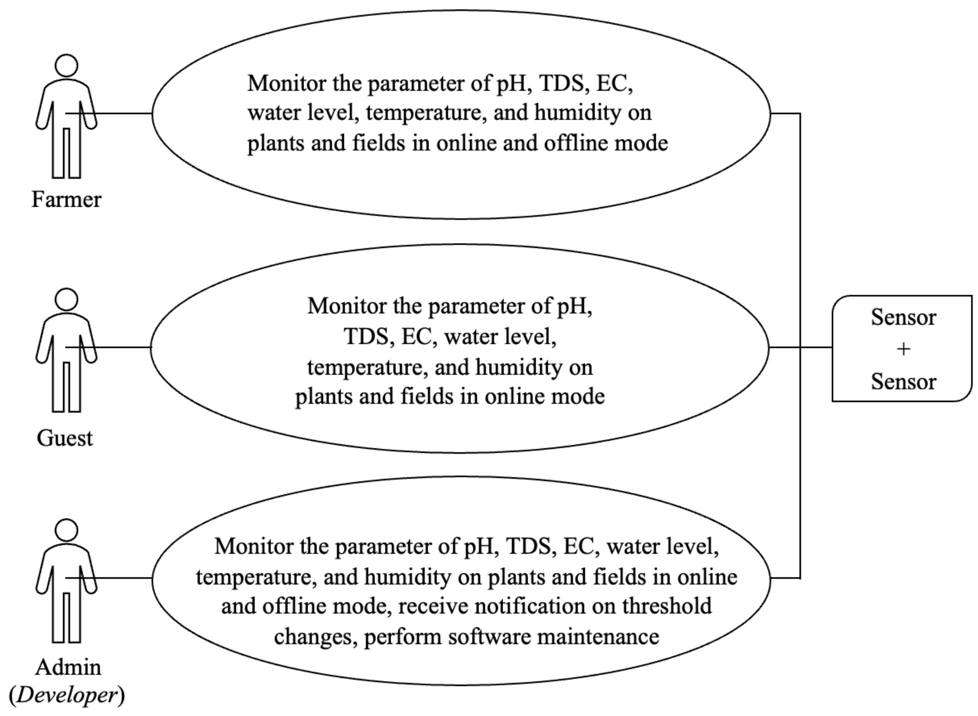

Monitor the parameter of pH, TDS, EC, water level, temperature, and humidity on plants and fields in online and offline mode
Farmer
Monitor the parameter of pH, TDS, EC, water level, temperature, and humidity on plants and fields in online mode
Guest
Sensor + Sensor
Monitor the parameter of pH, TDS, EC, water level, temperature, and humidity on plants and fields in online and offline mode, receive notification on threshold changes, perform software maintenance
Admin (*Developer*)

Fig. 2 Smart agriculture use-case diagram.

TABLE I
USE-CASE SUBSYSTEM APPLICATION

Subsystem	User/Actor	Description
Management user account	Farmer & Guest	The user can register themself and update or view details. The user can view and edit other user details and delete them. Farmer users must have a serial number for farmer-specific application features.
Management farmer monitoring	Farmer	Farmer can monitor their field both online and offline and the monitoring display will be displayed according to the farmer's username.
Management guest	Guest	Guest can see the latest info about the field.
Management farmer controlling	Farmer	Farmer can control the thresholds dan will be updated to the local and central database.

management user account, management monitoring farmer, management guest, and management controlling farmer.

A. *Design on the Application Functionality Testing (Black Box Testing)*

This stage contained data retrieval conducted along with the system-wide testing. In the application side, testing was carried out using the black box method. The data that would be taken was in the form of incorrect or missing functions, interface errors, errors in data structures or external database access, performance errors, and initialization and termination errors.

B. *Design on the Load Testing Activity*

Application load testing is the time the data moves from one activity to another. In addition, the analysis of load time on monitoring activities and load time controlling activities were carried out. The primary purpose was to test whether the application could handle requests from users, such as maintaining response time to user requests. Some applications can experience problems, namely long load time. It is a condition where an application takes more than one minute to lad. If possible, the load time should be under 10 seconds. The time needed to load in an area with a bad connection is less than 5 seconds. Otherwise, the load time should be less than 3 seconds.

C. *Design on the Application Performance Analysis*

App performance analysis was done by utilizing the Profiler feature in the Android Studio IDE. The more responsive a mobile application, the better it will be. The more intelligent the application recognizes all forms of actions that the user wants, the more value will be added to the quality of the application.

IV. RESULT AND DISCUSSION

A. *Application Functionality Test Result (Black Box Testing)*

Application-side testing (black box testing) tests the application runtime according to the scenario. This test focuses on functionality and output. The testing is aimed at the

TABLE II
APPLICATION FUNCTIONALITY TEST RESULT

No.	Test Activity	Expected Outcome	Status
1	Intro: Splash Welcome	Splash: showing the logo and slogan of OmahIoT for 3-4 seconds. Welcome: showed info concerning OmahIoT in a carousel	Splash (success) Welcome (success)
2	User authentication on: Login Register Profile	Login: user could log in with the registered email and password. Register: user could register an account by filling in their name, email, password, and phone number. Profile: user could change their data and name, email, address, phone number, and profile picture.	Login (success) Register (success) Profile (success)
3	Dashboard menu: Smart agriculture	Showing the initial menu consisting of Monitoring, Controlling, Profile, and Apps Guide. The content for each menu would be different based on each user.	Dashboard smart agriculture (success)
4	Monitoring menu: Offline Monitoring Online Monitoring	Offline mode: by connecting the device to the nearest local WiFi to the system. Showing the sensor value in real-time through the edge according to the username. Online mode: by connecting the device to a regular WiFi. Showing the sensor value with a gauge chart using WebView.	Offline Monitoring (success) Online Monitoring (success)
5	Controlling menu: Offline Controlling Online Controlling	Setting several threshold parameters of sensors from the app. The communication method used for offline mode was local WiFi + MQTT protocol to a nearby device. The data in the latest offline mode was sent to the central database on online mode.	Controlling Smart Agriculture (success)
6	Notification: Smart Agriculture	On smart agriculture, a notification would pop up during controlling. A notification would notify the user that the threshold data has been sent to the device.	Smart Agriculture Notification (success)
7	App Guide	Displaying a number of information and instructions based on the user role.	App Guide (success)

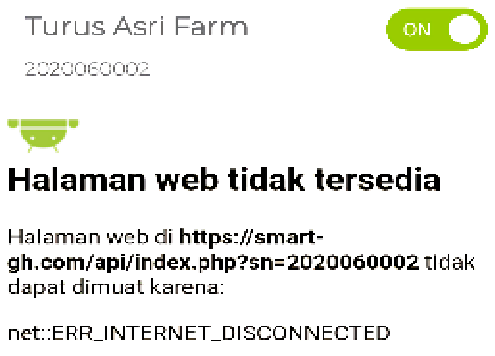

Turus Asri Farm
2020060002
ON
Halaman web tidak tersedia
Halaman web di https://smart-gh.com/api/index.php?sn=2020060002 tidak dapat dimuat karena:
net::ERR_INTERNET_DISCONNECTED

Fig. 3 Online mode disconnected.

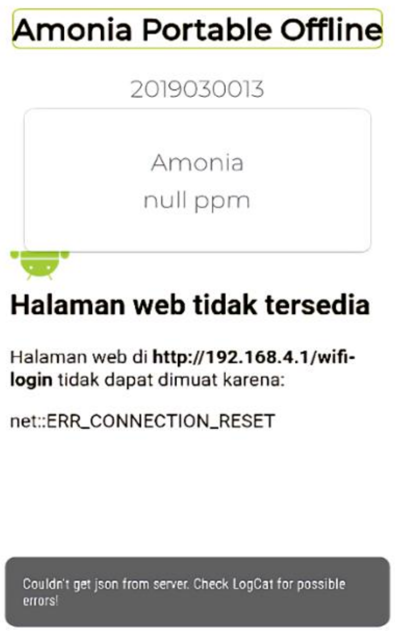

Amonia Portable Offline
2019030013
Amonia
null ppm
Halaman web tidak tersedia
Halaman web di http://192.168.4.1/wifi-login tidak dapat dimuat karena:
net::ERR_CONNECTION_RESET
Couldn't get json from server. Check LogCat for possible errors!

Fig. 4 Offline mode disconnected.

application's design following the modeling and response if there are bugs in the app. The tool used to test this functionality was an Android smartphone as well as an emulator with an actual device in the form of an Android smartphone. The API used to develop the Android OmahIoT application was a minimum of SDK 29. As for the Android version, it should be Jelly Bean at the least. However, during the testing, Android version 10 was used. Table II presents the results of the application functionality test.

This Android application is made to monitor smart agriculture online and offline. If connected to the internet, the Android app has the task of reading sensor data in the form of pH, total dissolved solids (TDS), light, greenhouse temperature, humidity, water tank volume, and nutrient temperature, along with reading sensor data in the form of chicken coop temperature, ammonia levels, humidity, fan status, cooler status, and heater status. The sensor data is sent to the central database first, then the Android app receives it via the cloud with WebView. The mode of online operation relies heavily on an internet connection. Fig. 3 displays the online mode failures due to a lost internet connection.

Monitoring in offline mode relies heavily on the device’s local WiFi access point (AP). When the connection to the

TABLE III
ACTIVITY LOAD TEST

No.	Bandwidth (Mbps)	Object	Load Time (s)
1	4.54 (bad)	Splash screen	3.000
		Welcome activity	5.053
		Login activity	3.553
		Turus Asri dashboard	8.123
		Mustika dashboard	4.159
		Turus Asri online monitoring	6.306
		Turus Asri offline monitoring	1.453
		Mustika offline monitoring	7.458
		Turus Asri controlling	0.300
		Mustika controlling	0.200
		Turus Asri profile	1.637
		Mustika profile	3.747
		Turus Asri logout	0.237
		Mustika logout	1.790
Average load time for each activity			3.507
2	35.15 (good)	Splash screen	3.000
		Welcome activity	2.847
		Login activity	1.127
		Turus Asri dashboard	1.420
		Mustika dashboard	0.358
		Turus Asri online monitoring	1.702
		Mustika online monitoring	3.620
		Turus Asri offline monitoring	1.410
		Mustika offline monitoring	4.320
		Turus Asri controlling	0.020
		Mustika controlling	0.030
		Turus Asri profile	0.388
		Mustika profile	0.504
		Turus Asri logout	0.060
		Mustika logout	0.205
Average load time for each activity			1.400

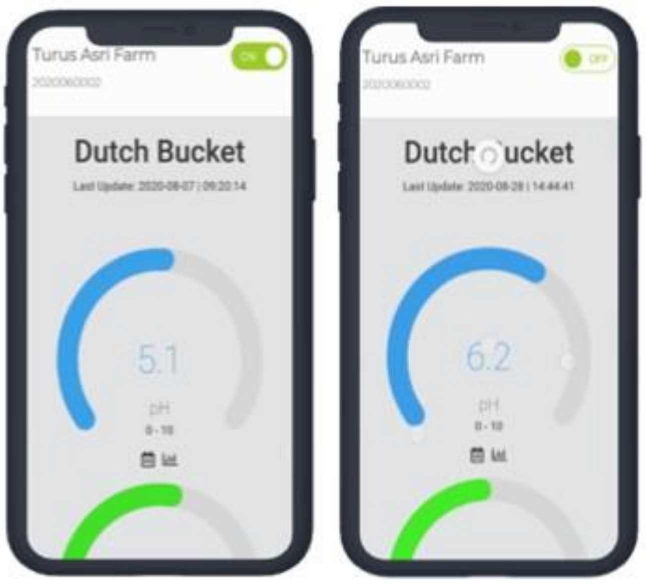
access point (AP) is disconnected, the offline monitoring will fail to receive data. Indications of offline monitoring failure due to a disconnected AP connection are shown in Fig. 4.

The displays of offline and online monitoring are shown in Fig. 5. The controlling interface in this Android application uses offline mode or through the AP connection. Fig. 6 shows the controlling smart agriculture on the Android app.

Controlling for smart agriculture uses MQTT protocol while the Android application acts as a client and AP Raspberry Pi connection. The manager only needs to enter the threshold value data in the form of TDS and pH values that are wanted according to the writing format in the hint form. Controlling is successful if the user gets a "connected" message when entering the controlling activity and receives message indicates the data has been published after entering the threshold value correctly and then pressing the publish button.

B. *Activity Load Test Result*

Activity load test is a performance test in which the system response time is measured in various load conditions. On



(a) (b)

Fig. 5 Monitoring display, (a) offline, (b) online.

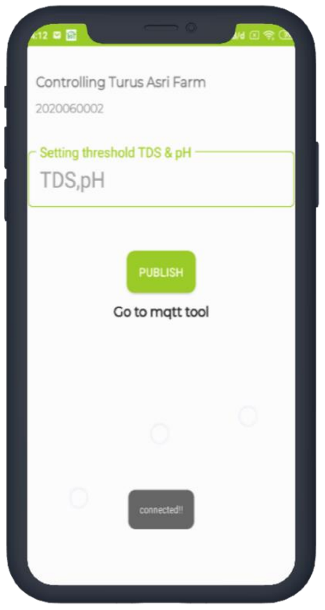


Fig. 6 Controlling smart agriculture.

average, the targeted load time per activity is less than 5 seconds on a bad network condition, while on a good network condition is less than 3 seconds. Table III presents the results of the activity load test.

Table III shows that in a poor network bandwidth of 4.54 Mbps, on average, the load time is 3.507 seconds. The load time is 1.4 seconds on a good network bandwidth, meaning that the targeted load time is fulfilled. Activity with the worst load time is from the login activity to the dashboard menu of Turus Asri, which takes 8.123 seconds in bad network bandwidth. At the same time, the activity with the best load time is in the controlling activity of Mustika, which takes 0.2 seconds in a bad network bandwidth. On a good bandwidth condition, the highest load time is in the offline monitoring of Mustika, which

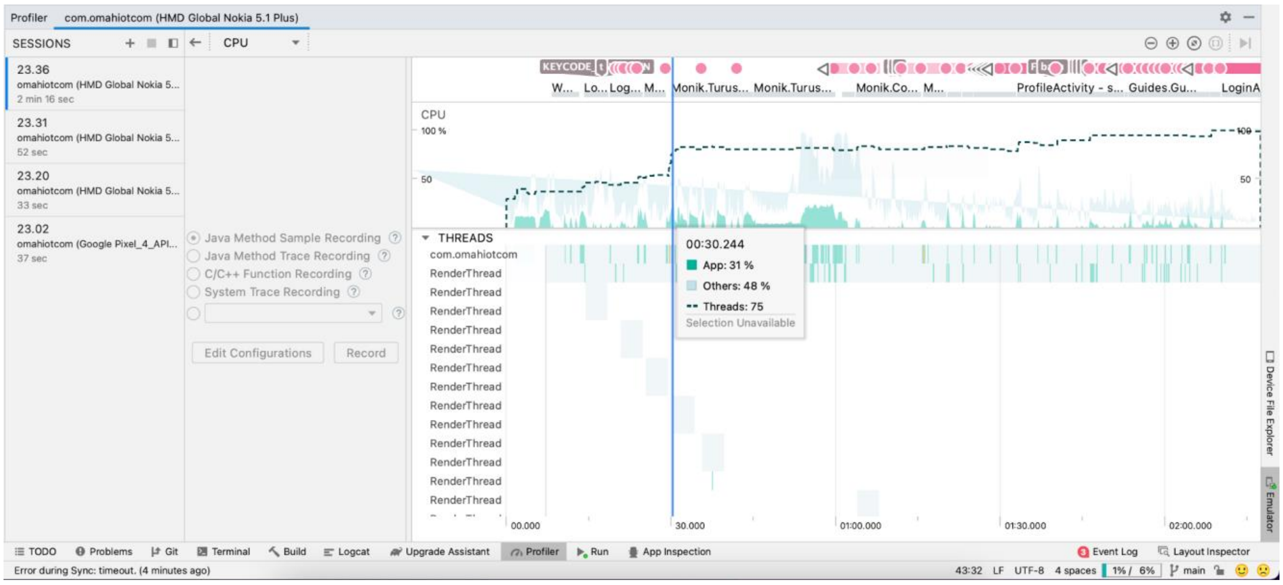


Fig. 7 Profiling results of OmahIoT application CPU.

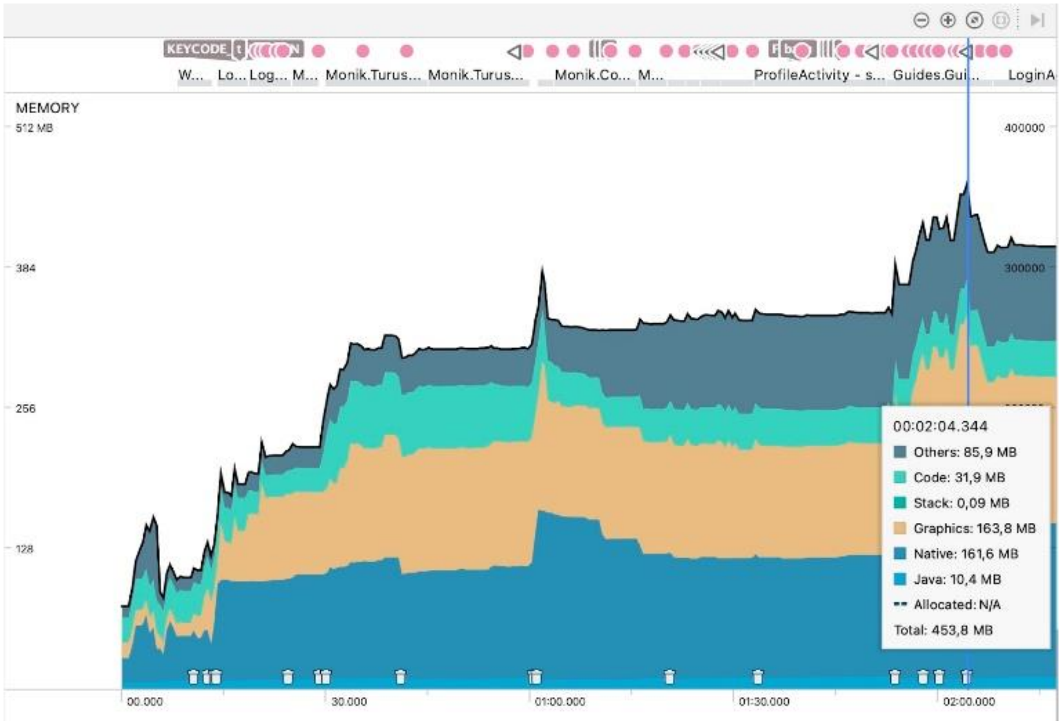


Fig. 8 Profiling memory results of OmahIoT application.

takes 4.32 seconds. The fastest load time in a good bandwidth condition is in the controlling activity of Turus Asri which takes 0.02 seconds.

C. Application Performance Analysis Result

Android Studio’s Profiler provides real-time data to help understand how the application uses CPU resources, memory, and network. Fig. 7 shows the results of OmahIoT application CPU profiling. Optimizing the application’s CPU usage has many advantages, such as providing a faster and smoother user experience, as well as saving the device’s battery life. Fig. 7 also shows that none of the CPU usage reaches 50% when the OmahIoT application runs. The highest CPU usage was when the application displayed the monitoring activity, at 31%. While loading the web in the monitoring activity, users indirectly interacted with the database server which leads to an increase in CPU usage.

Fig. 8 shows the results of profiling memory OmahIoT application. This test helps identify memory leaks and memory churn (indicating the number of temporary objects allocation that occurs in a given time) that can cause the application to stutter, stop responding, and even stop working. This component displays real-time graphics of an application’s memory usage, allowing to record heap dumps, force memory

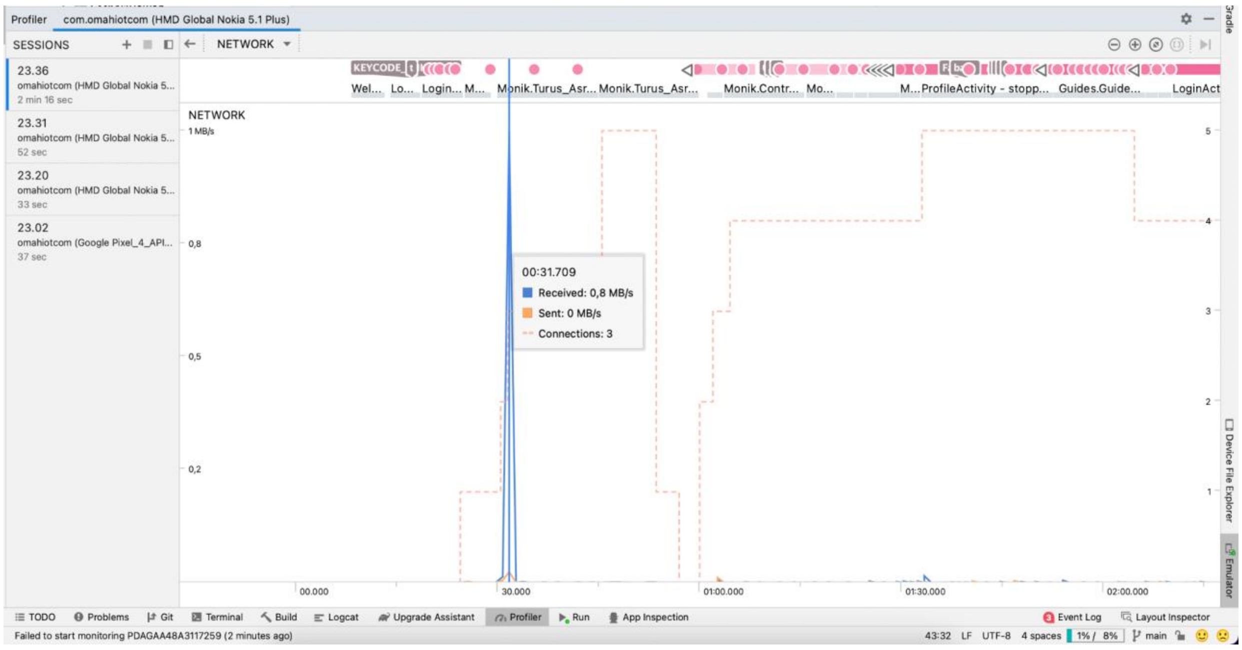

Profiler
com.omahiotcom (HMD Global Nokia 5.1 Plus)
SESSIONS
NETWORK
23.36
omahiotcom (HMD Global Nokia 5...
2 min 16 sec
23.31
omahiotcom (HMD Global Nokia 5...
52 sec
23.20
omahiotcom (HMD Global Nokia 5...
33 sec
23.02
omahiotcom (Google Pixel_4_API...
37 sec
KEYCODE
Wel... Lo... Login... M... Monik.Turus_Asr... Monik.Turus_Asr... Monik.Contr... Mo... M... ProfileActivity - stopp... Guides.Guide... LoginAct
NETWORK
1 MB/s
0,8
0,5
0,2
00:31.709
Received: 0,8 MB/s
Sent: 0 MB/s
Connections: 3
5
4
3
2
1
00.000
30.000
01:00.000
01:30.000
02:00.000
TODO
Problems
Git
Terminal
Build
Logcat
Upgrade Assistant
Profiler
Run
App Inspection
Event Log
Layout Inspector
Failed to start monitoring PDAGAA48A3117259 (2 minutes ago)
43:32 LF UTF-8 4 spaces main
Device File Explorer
Emulator

Fig. 9 Profiling network results of OmahIoT application.

waste cleaning, and track memory allocation. As seen in Fig. 8, the memory spent on OmahIoT applications is 453.8 MB at most when online monitoring of Turus Asri moves to the activity menu dashboard. From the memory details, the allocated objects from Java code were 10.4 MB; memory used by Java stacks and natives in applications based on the number of threads used was 0.09 MB. Memory used by applications for code and resources, such as bytecode dex, compiled or optimized dex code, .so libraries, and fonts were 25 MB and uncategorized memory usage was 85.9 MB.

Fig. 9 shows the results of the network profiling application. The network profiler displays a real-time network activity on the timeline, showing the data sent and received, as well as the current number of connections. The results show that in the OmahIoT application three files have been sent or received during the selected activity in the list across the application CPU thread. The three files ran based on the thread used, e.g., file 192.168.4.1 running using thread asynctask (GET method) and taking 31.07 seconds to transmit (delivery and reception), this is the worst result among others. It because the required connection is a local connection, while in testing an internet connection is used.

Based on the network profiling data, the network would only work on activities requiring files as data transmission. The time needed to send and receive responses depends on the thread—e.g., asynctask thread—and the network connection used.

Based on all profiling data, it can be interpreted that the application runs appropriately. Besides that, the application can be said to be relatively light to use since it depends on the internet connection when sending and receiving data.

V. Conclusion

Based on the tests that have been done, Android-based applications can run as designed and can help farmers and ranchers to monitor and control land in one application only. The functionality testing results (Black Box) of Android-based applications can run as per the modeling. This application had an average load time value across activities of 3.507 seconds when the bandwidth was 4.54 Mbps. Meanwhile, the average load time generated when bandwidth conditions of 35.15 Mbps were 1.4 seconds. The value has met the target of the expected load testing results. In terms of performance, this application is quite light, with the largest central processing unit (CPU) usage of 28%, the largest memory usage of 236.7 MB, and the network performance between sending and receiving the longest data files was when using the local network (offline).

Conflicts of Interest

All authors declare that we have no conflict of interest in this research.

Author Contributions

Conceptualization, Helmy and Arif Nursyahid; methodology, Thomas Agung Setyawan; system design, Helmy, Thomas Agung Setyawan and Ari Sriyanto Nugroho; software, Fenny Rahmasari; analysis, Fenny Rahmasari and Ari Sriyanto Nugroho; writing—original draft preparation, Helmy; writing—review and editing, Helmy, Fenny Rahmasari and Ari Sriyanto Nugroho.

References

[1] R. Khairiyakh, I. Irham, and J.H. Mulyo, “Contribution of Agricultural Sector and Sub Sectors on Indonesian Economy,” *Ilmu Pertan. (Agricultural Sci.)*, Vol. 18, No. 3, pp. 150–159, 2015.

[2] “Laporan Bulanan Data Sosial Ekonomi,” Badan Pusat Statistik, 2019.

[3] K.A. Patil and N.R. Kale, “A Model for Smart Agriculture Using IoT,” *2016 Int. Conf. Global Trends Signal Process., Inf. Comput., Commun. (ICGTSPICC)*, 2016, pp. 543–545.

[4] A.P. Atmaja, A. El Hakim, A.P.A. Wibowo, and L.A. Pratama, “Communication Systems of Smart Agriculture Based on Wireless

Sensor Networks in IoT," *J. Robot. Control*, Vol. 2, No. 4, pp. 297–301, Jul. 2021.

[5] E. Hesti and Adewasti, "Aplikasi Android sebagai Pengontrol Jarak Jauh Smarthome dengan Koneksi Jaringan Internet," *J. Surya Energy*, Vol. 2, No. 2, pp. 157–165, Mar. 2018.

[6] S.A. Arnomo and H. Hendra, "Perbandingan Fitur Smartphone, Pemanfaatan dan Tingkat Usability pada Android dan iOS Platforms," *J. Nas. Inform., Teknol. Jaringan*, Vol. 3, No. 2, pp. 184–192, 2019.

[7] A.T Sondha, U. Sa'adah, F.F. Hardiansyah, and M.B.A. Rasyid, "Framework dan Code Generator Pengembangan Aplikasi Android dengan Menerapkan Prinsip Clean Architecture," *J. Nas. Tek. Elektro, Teknol. Inf.*, Vol. 9, No. 4, pp. 327–335, Nov. 2020.

[8] I.Z.T. Dewi, *et al.*, "Smart Farming: Sistem Tanaman Hidroponik Terintegrasi IoT MQTT Panel Berbasis Android," *J. Keteknikan Pertan. Trop., Biosist.*, Vol. 9, No. 1, pp. 71–78, 2021.

[9] R.P. Astutik, "Aplikasi Telegram untuk Sistem Monitoring pada Smart Farming," *J. Teknol., Terap. Bisnis*, Vol. 2, No. 1, pp. 1–6, 2019.

[10] O. Pandithurai, S. Aishwarya, B. Aparna, and K. Kavitha, "Agro-Tech: A Digital Model for Monitoring Soil and Crops Using Internet of Things (IOT)," *2017 3rd Int. Conf. Sci. Technol. Eng. Manage.*, 2017, pp. 342–346.

[11] A.D. Chachadi and G.R. Rajkumar, "Development of Automated Hydroponic System for Smart Agriculture," *Int. Res. J. Eng. Technol.*, Vol. 8, No. 6, pp. 1273–1278, Jun. 2021.