International Journal of Informatics and Communication Technology (IJ-ICT)
Vol. 15, No. 2, June 2026, pp. 850~860
ISSN: 2252-8776, DOI: 10.11591/ijict.v15i2.pp850-860

Mobile device application design for ThingSpeak interface using flutter

Moehammad Sauqy Ihza Zuliandra[1], Tigor Hamonangan Nasution[1], Ainul Hizriadi[2]
[1]Department of Electrical Engineering, Faculty of Engineering, Universitas Sumatera Utara, Medan, Indonesia
[2]Department of Technology Information, Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Medan, Indonesia

Article Info

Article history:

Received Nov 1, 2024
Revised Nov 11, 2025
Accepted Dec 14, 2025

Keywords:

Flutter
Internet of things
Mobile apps
Monitoring systems
ThingSpeak

ABSTRACT

The rapid development of internet of things (IoT) is prompting many people to design applications, particularly for monitoring applications based on mobile apps. This includes research designs to monitor electrical parameters from PV and the development of health monitoring applications. Previous research required a separate application to scan each IoT device. In this research, a mobile app-based IoT monitoring system was built using flutter. With this, people no longer need to design separate mobile apps for various IoT devices. This application utilizes the flutter framework, while the cloud component uses ThingSpeak. These research results show that data from multiple IoT devices can be transferred to the user's mobile app. This application enables the monitoring of various IoT devices through a single mobile app, thereby enhancing the efficiency of IoT device design and management.

Corresponding Author:

Tigor Hamonangan Nasution
Department of Electrical Engineering, Faculty of Engineering, Universitas Sumatera Utara
Medan, Indonesia
Email: tigor.nasution@usu.ac.id

1. INTRODUCTION

The rapid development of the internet of things (IoT) has led to increased innovation in areas such as sensors, cyber-physical systems, and cloud computing [1]-[3]. One key trend is the rise of mobile-based IoT monitoring applications, which allow users to track device performance in real-time through smartphone interfaces [4]-[6]. To support this, platforms like ThingSpeak have emerged as cloud-based solutions for IoT data collection, storage, visualization, and analysis, offering features such as RESTful application programming interfaces (APIs) and real-time data streaming [7]-[10].

Despite these advancements, most existing mobile IoT applications are developed for specific domains or devices; for example, as noted by Winasis *et al.* [11] designed a photovoltaic system monitoring tool, while Jankovec *et al.* [12] focused on IoT-based patient monitoring. While these systems demonstrate the potential of mobile IoT integration, they are limited in scope and require users to operate different applications for each device or use case, reducing efficiency and scalability.

This study addresses that gap by developing a general-purpose mobile IoT monitoring application using flutter, integrated with ThingSpeak. The proposed solution allows users to add multiple IoT devices via application programming interface uniform resource locators (API URLs), eliminating the need for separate apps. Furthermore, the cross-platform capability of flutter (Android, iOS, web, and desktop) enhances accessibility. By simplifying the integration process and centralizing monitoring across devices, this research aims to increase efficiency and usability in real-time IoT monitoring systems.

2. METHOD

2.1. System overview

This mobile application-based IoT monitoring system is designed to make it easier for the general public to monitor various IoT data, such as temperature and humidity, using a smartphone. This data can be monitored via the smartphone application. The general description of the system is shown in Figure 1.

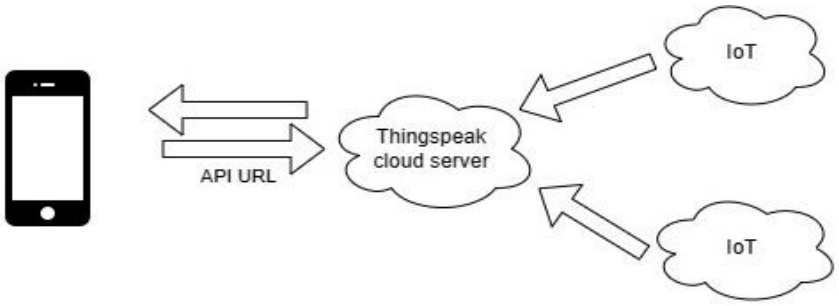

Figure 1. Monitoring system overview

So that IoT data can be monitored via a smartphone, the IoT system sends the data to the ThingSpeak cloud so that it can be read via the smartphone application. To read the data, the system requires an API URL obtained from the ThingSpeak cloud, which enables access to the data sent from the IoT device via smartphones. The data obtained is in JavaScript object notation (JSON) format. The JSON format is a text-based file format generally used in data exchange between servers and clients [13]-[15]. JSON is also compatible with many programming languages, especially flutter [16]-[18].

Flutter is a platform that creates multiplatform applications with a single code base [19], [20]. The resulting application can be used on Android, including iOS, web, and desktop platforms. Mobile flutter is a platform widely used by developers to create mobile applications using attractive designs. Flutter has two essential components, namely:

- SKD is essential because it contains tools for creating applications that run on various platforms.
- Meanwhile, the UI framework is a UI component for customizing applications according to needs.

2.2. Research procedures

The research procedure follows a systematic development process involving the design, implementation, and evaluation of a mobile-based IoT monitoring system. The project began with a requirement analysis to identify the limitations of existing IoT applications, particularly the need for a unified platform capable of monitoring multiple devices without developing separate applications. Based on these findings, the application was designed using the flutter SDK (version 3.10), enabling cross-platform deployment. The UI/UX design adhered to material design principles and incorporated key features, including onboarding, dashboard, channel input, and data visualization screens. The backend logic was implemented in Dart, where a custom API service module was developed to fetch, parse, and present JSON-formatted data from ThingSpeak based on user-provided API URLs.

To ensure system functionality and replicability, testing was conducted on various Android smartphones and emulators using unit tests, integration tests, and manual testing procedures. IoT data simulation was carried out using ESP8266 microcontrollers [21], [22] connected to DHT11 sensors [23], which transmitted environmental data (e.g., temperature and humidity) to the ThingSpeak platform. The mobile app then retrieved and visualized the data in both numerical and graphical formats. The overall system architecture consists of three layers: the IoT device layer, the ThingSpeak cloud layer, and the mobile application layer. This layered approach provides a scalable and replicable structure, supporting real-time monitoring with an average memory usage of 259 MB, making the system suitable for practical deployment and further research extensions.

2.3. Use case diagram modeling

In the use case diagram modeling shown in Figure 2, there are actors, and in the monitoring system, there are display data, add channels, and delete channels. In this case, the actor in this diagram is the user who displays data, adds channels, and deletes channels. A use case diagram is a diagram that describes the relationship between actors and the system [24]-[26]. Use case diagrams can describe an interaction between one or more actors and the system to be created [27], [28]. Use case diagrams can be used to identify system

functions and illustrate an actor's interaction with the system. This component then explains the communication between actors and the existing system. In this way, use cases can be presented in a simple sequence and will be easily understood by consumers. The primary benefit of the use case is to facilitate communication between domain experts and end users, thereby ensuring an accurate understanding of the system's requirements and needs.

In the data display, there is data displayed in the form of values along with a description of the data, and the data is also displayed in the form of a line graph. Adding a channel is a user activity that involves creating a new channel with a display in the form of a search bar. In the search bar, enter the API URL or URL API obtained from the ThingSpeak server. This API URL enables data transfer from the ThingSpeak cloud to the user's smartphone. Delete channel functions to enable users to remove stored data or any unwanted content.

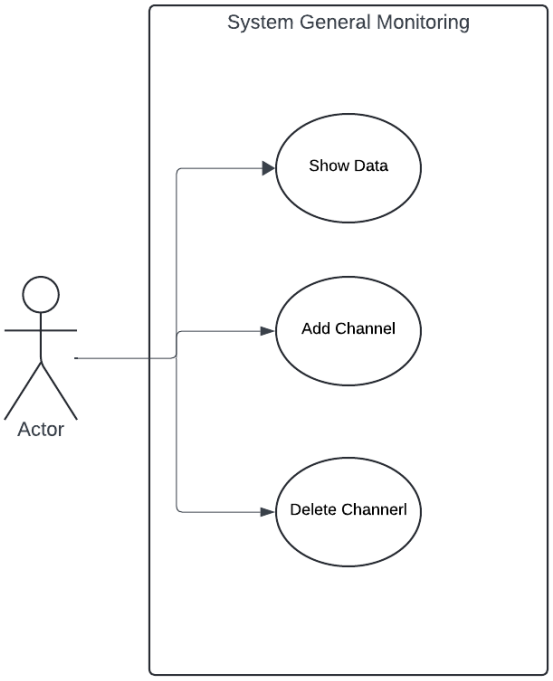

Figure 2. Use case diagram

2.4. Activity diagram

An activity diagram is a diagram that depicts the flow of user activities when opening an application to monitor IoT data. An activity diagram is a design of the activity flow or workflow in a system that will be run [29]-[31]. Activity diagrams are also used to define or group the display flow of the system. Activity diagrams have components with specific shapes that are connected by arrows. The arrow points to the sequence of activities that occur from start to finish. The activity diagram for the added channel monitoring system can be shown in Figure 3.

In the diagram, there are users, systems, and databases. In the user, the actor can open the application, which is the initial stage in the diagram. After the actor opens the application, the system processes the actor's request and then displays the loading interface. After that, the system will display a dashboard. The dashboard displays IoT device data. IoT device data displays are stored in the ThingSpeak cloud database. Activity logs are also stored in this database, and data will be entered when the IoT device is turned on. In the user actor, you can add channels by creating a new one. The system will then receive commands from the user and display the input menu page. The input menu page contains a bar for entering the data API URL, allowing the data to be saved. After that, the system displays a loading interface and then presents the data to the user, allowing the actor to view the data in the form of values, descriptions, and graphs.

The activity diagram of the channel deletion monitoring system contains users and flutter. Actors can also delete channels within the user; then flutter will delete the channel. After that, flutter redirects to the dashboard, and the diagram is displayed in Figure 4.

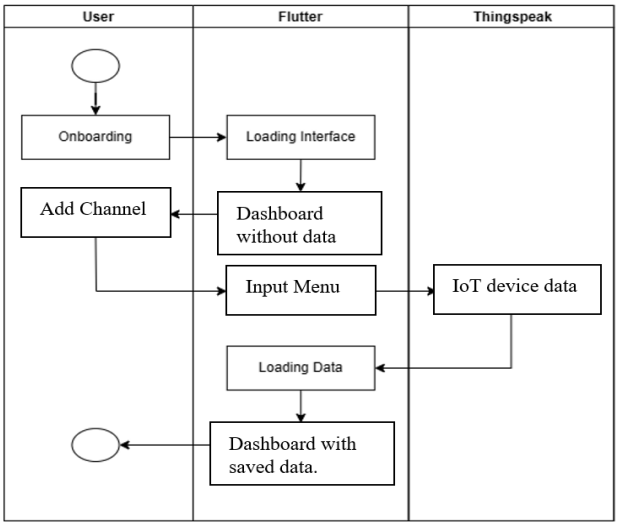

Figure 3. Add channel monitoring system activity diagram

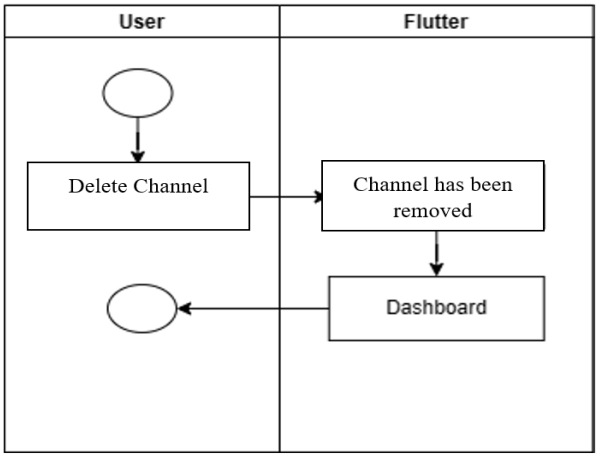

Figure 4. Channel delete monitoring system activity diagram

2.5. System design

The application development began by designing the system's ability to retrieve data from the ThingSpeak cloud using API integration. A service class was created in Dart to fetch data via a URL, where each API URL corresponds to an IoT device channel. The data retrieved from ThingSpeak is in JSON format and was initially unstructured. To handle this, custom classes (allnamachannel and allchannel) were developed to map and structure the data, allowing the application to correctly display each device's name and corresponding values on the smartphone interface.

The onboarding page was designed as the first user interface shown when the application is opened. It features a background using a green container, accompanied by text and image widgets. The title text, styled in Roboto font with white color and size 40, displays "FINAL PROJECT," while the image logo uses a 100x100 mm PNG asset. A "Start" button built with a TextButton widget is placed below, styled with rounded corners and a green background. When pressed, it navigates the user to the dashboard screen.

The dashboard contains an AppBar titled "Channel" in a green background and Roboto font. Below it, a list of IoT channels is displayed using boxes styled as pressable TextButton widgets. Each box contains the device name aligned to the left and a trash icon that allows users to delete a channel. The list adapts dynamically based on stored data. Additionally, a floating action button (+) is positioned at the bottom right, enabling users to add new channels.

Upon pressing the add button, users are directed to the "Add New Channel" page. This page has an AppBar labeled "Add New Channel" with a light green background. Below it is a TextField where users input the ThingSpeak API URL. The text field includes input borders and a label ("URL") styled in Roboto

font. When the user enters a valid API and presses the "Save" button (an ElevatedButton), the system retrieves and parses the JSON data, classifies it, and updates the dashboard by displaying the new channel.

Once added, users can press a channel from the dashboard to view IoT data. This leads to the value display page, which shows sensor readings such as temperature or humidity in a styled layout using TextButton widgets. Each value item displays the sensor name and current reading in Roboto font, accompanied by an arrow icon indicating that the data can be viewed further as a graph. The AppBar at the top of this page displays the channel name and is styled with a green background.

The graphical view page is accessible by pressing a value item. It contains an AppBar labeled "Graphics" and displays a line chart built using sfCartesianChart. The x-axis represents timestamps from the ThingSpeak data, and the y-axis shows the corresponding sensor values. This visual representation enhances user understanding by providing real-time trends of IoT data directly on their mobile device.

3. RESULTS AND DISCUSSION

This section presents the test results of designing an IoT monitoring system using flutter, beginning with the onboarding page and dashboard test results. Next, test the "Add New Channel" page to add new channels. Next, verify the IoT data page's results to confirm whether the data is stored on the dashboard page. The IoT data page is divided into two sections: the value IoT data page and the graph IoT data page. The IoT data page, which contains IoT data from ThingSpeak in JSON format, displays data in the form of values. In contrast, the graph IoT data page presents data in a line graph. After showing the IoT data page test results, proceed to display the smartphone specification test results to determine if the smartphone being tested can run this application. Next, test the application's size and memory usage to determine its size and the memory it consumes.

3.1. Onboarding page and dashboard test results

In testing, opening the application begins with installing the application on the user's smartphone. After the installation is complete, open the application on your smartphone. After that, the initial interface page will appear. If the user presses the "Start" button, they will be taken to the dashboard page, as shown in Figure 5.

3.2. Add new channel

From the dashboard page, the user selects "Add New Channel" by pressing the plus icon (+) button located in the bottom corner of the dashboard. After that, the user will be directed to the Add New Channel page. Users must fill in the Channel ID from the ThingSpeak cloud API. After that, the user can press the "save" button. The "Add New Channel" page is shown in Figure 6.

3.3. IoT data page

After adding a new channel, when the user enters the channel ID on the Add New Channel page, the channel name obtained from ThingSpeak will appear on the smartphone. The IoT data display, presented in the form of channel names, is shown in Figure 7.

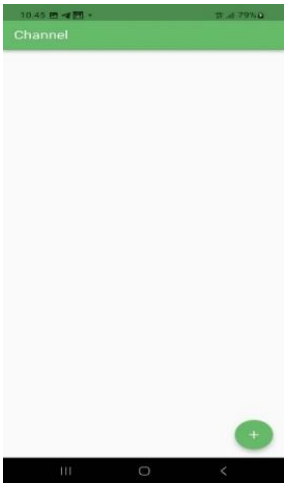

Figure 5. Dashboard page

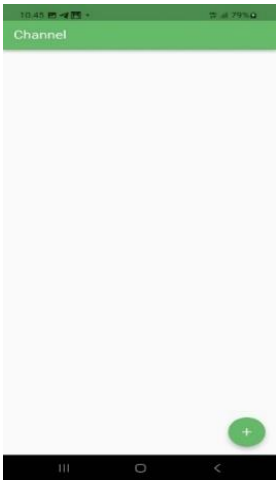

Figure 6. Add a new channel page

Figure 7. IoT data page

3.4. Values page

After the user presses the channel name bar on the IoT data page, they will be directed to a page containing values from the IoT device, such as temperature and humidity. After that, the data on ThingSpeak is adjusted to match the data on the smartphone application. The IoT value data page on ThingSpeak is shown in Figure 8, and the IoT value data page on the smartphone is shown in Figure 9. If the value display bar is pressed, it will be directed to the data display as a graph.

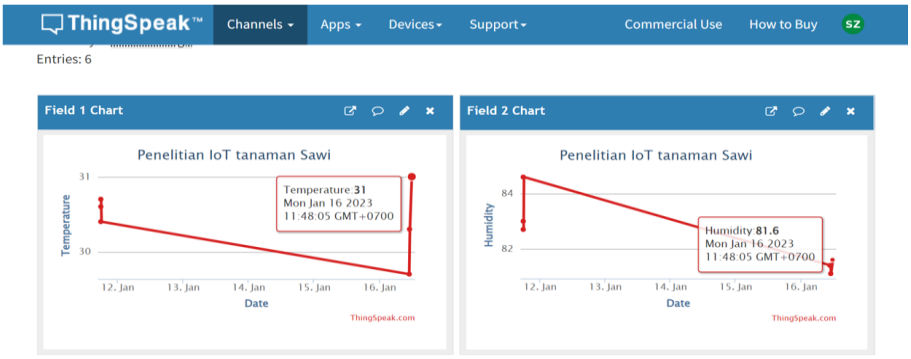

Figure 8. Value IoT data page on ThingSpeak

Figure 9. IoT value data page on Smartphone

3.5. Pages in the form of graphics

Apart from pages in the form of values, IoT data pages are displayed in graphical form; these pages are obtained when the user presses the value bar contained in the IoT data display. The graph on this page is a line graph. After that, the data on ThingSpeak is adjusted to match the data on the smartphone application. The graphical IoT data page on ThingSpeak is shown in Figure 10. and the graphical IoT data page on the smartphone is shown in Figure 11.

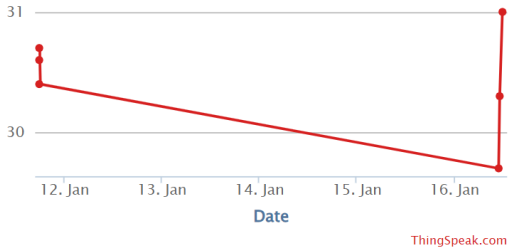

Figure 10. IoT graph data page on ThingSpeak

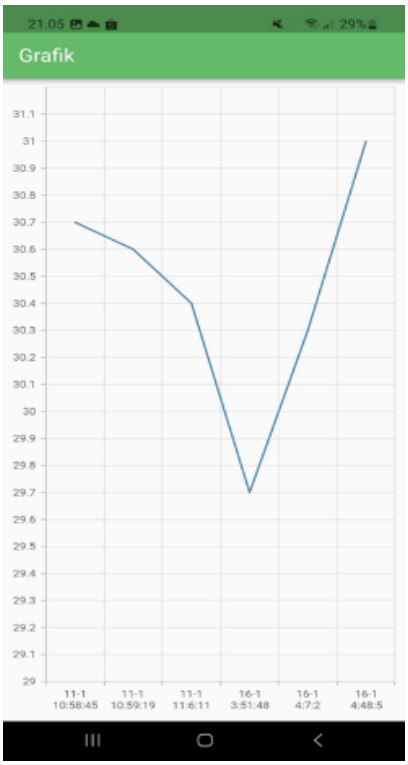

Figure 11. IoT graph data page on a smartphone

3.6. Testing smartphone specifications

To test this application, several smartphones with different specifications are needed to demonstrate its success. The list of smartphones used, along with their specifications, is shown in Table 1.

Table 1. Smartphones are tested, and their specifications are evaluated

Brand	OS	Specification	Result
Samsung Galaxy A52	Android 13	Screen: Super AMOLED 6.5-inch (1080 x 2400 piksel), Dimension: 159.9 x 75.1 x 8.4 mm, Chipset: Qualcomm Snapdragon 720G, GPU: Adreno 618 CPU: Octa-core (2 x 2.3 GHz Kryo 465 Gold & 6 x 1.8 GHz Kryo 465 Silver)	Succeed
Xiaomi Redmi 9T	Android 12	Screen: IPS 6.53 inch (1080 x 2340 pixels), Dimension: 162.3 x 77.3 x 9.6 mm, Chipset: Qualcomm Snapdragon 662, GPU: Adreno 610 CPU: Octa-core (4 x 2.0 GHz Kryo 260 Gold & 4 x 1.8 GHz Kryo 260 Silver)	Succeed
Xiaomi Poco M3	Android 12	Screen: IPS 6.53 inch (1080 x 2340 pixels), Dimension: 162.3 x 77.3 x 9.6 mm, Chipset: Qualcomm Snapdragon 662, GPU: Adreno 610 CPU: Octa-core (4 x 2.0 GHz Kryo 260 Gold & 4 x 1.8 GHzKryo 260 Silver)	Succeed
Oppo Reno 4	Android 12	Screen: OLED 6.4 inch (1080 x 2400 pixels), Dimension:160.3 x 73.9 x 7.7 mm, Chipset: Qualcomm Snapdragon 720G, GPU: Adreno 618 CPU: Octa-core (2 x 2.3 GHz Kryo 465 Gold & 6 x 1.8 GHz Kryo 465 Silver)	Succeed

3.7. Testing size and memory usage

In this test, the application size and memory usage were evaluated to determine system efficiency and resource consumption. One of the Android devices was used as the benchmark for this evaluation. The application size is presented in Figure 12, while memory consumption was measured using SauceLabs under various usage scenarios. The test results are summarized in Table 2.

Figure 12. Display of mobile memory usage

Table 2 shows the memory usage under different conditions, based on the number of IoT channels being monitored and the volume of data within each channel. When the application runs without any channels, it uses 236 MB of memory. As channels are added and populated with data, memory usage increases accordingly, peaking at 278 MB when two channels are present with two data points each. Interestingly, even as the number of channels increases up to eight, memory usage stabilizes in the range of 252–256 MB, indicating efficient memory management and scalability. The average memory consumption across all tested scenarios is 259 MB, which falls within an acceptable range for flutter-based mobile applications, supporting the system's practicality for everyday use.

Table 2. Memory usage in applications

Testing condition	Memory used (MB)
When the channel is empty.	236
When there is one channel, it contains 8 data points.	271
When there are two channels, each channel contains two pieces of data.	278
When there are three channels, each channel contains 8 data points.	273
When there are four channels, each channel contains 3 data points.	255
When there are five channels, each channel contains 3 data points.	254
When there are six channels, each channel contains 3 data points.	256
When there are seven channels, each channel contains 6 data points.	252
When there are eight channels, each channel contains 8 data points.	252

3.8. Functional testing results

The system testing carried out is validation testing to test whether the specifications for the functional requirements of the smartphone application can be met. The system testing results are shown in Table 3. Based on the test results in Table 3, the system built has met testing standards. Thus, functionally, the designed system produces test results as expected.

Table 3. Functional testing results

Test cases	Test procedure	Expected results	Test results
OnBoarding	Press the application button on the smartphone menu.	The system can display an application launch screen and a Start button.	Succeed
Dashboard	Press the Start button on the onboarding page.	The system can display the main page of the channel list that will be saved.	Succeed
Add new Channel	Press the + icon button on the dashboard page.	The system adds new channels by inputting the API.	Succeed
IoT data is in the form of values	Open one of the channel lists stored on the dashboard.	The system can display IoT data in the form of a value.	Succeed
IoT data is typically presented in graphical form	Open one of the values contained in the data as a value.	The system can display IoT data in graphical or line chart formats.	Succeed

3.9. Discussion

Previous studies in IoT monitoring typically focus on single-purpose applications limited to specific devices or domains, such as photovoltaic systems [11] or health monitoring [12], often requiring separate apps for each device type. This study addresses that gap by developing a flexible, cross-platform mobile application using flutter, which enables users to monitor multiple IoT devices through a single interface by dynamically integrating ThingSpeak API channels. Experimental results show that the application performs reliably on various Android smartphones, retrieving real-time data accurately and displaying it in numerical and graphical formats that align with ThingSpeak. Functional and performance testing confirmed stable operation with an average memory usage of 259 MB, indicating its efficiency and compatibility across devices.

Compared to previous research, this system offers broader functionality, cross-platform support, and dynamic multi-device management within a single application. However, current limitations include its exclusive reliance on the ThingSpeak platform, lack of multi-user features, and absence of real-time alert notifications. Future research could expand the system's interoperability by integrating additional IoT cloud platforms (e.g., AWS IoT, Google Cloud), implementing push notifications, and incorporating AI for predictive analytics. In conclusion, the proposed system represents a significant step toward general-purpose, real-time IoT monitoring, offering a replicable and scalable solution that addresses key limitations in existing works while laying the groundwork for more intelligent and responsive IoT applications.

4. CONCLUSION

This research successfully developed and tested an IoT monitoring system using flutter and ThingSpeak, enabling users to monitor multiple IoT devices within a single mobile application. The system demonstrated accurate real-time data retrieval and visualization in both numeric and graphical formats, and functioned reliably across various Android devices with efficient memory usage averaging 259 MB. Key features, including onboarding, dashboard access, channel management, and data display, worked as intended, confirming the system's overall effectiveness.

Compared to prior research focused on single-device or domain-specific monitoring, this study presents a more flexible and scalable solution. With flutter's cross-platform capabilities and ThingSpeak integration, the application offers wide accessibility and adaptability. Future improvements may include support for other IoT cloud platforms (e.g., AWS, Google Cloud), AI-driven analytics, real-time alerts, and enhanced user interfaces. These enhancements would allow the system to evolve into a more intelligent, responsive, and user-friendly tool for broader IoT monitoring applications.

FUNDING INFORMATION

This research was funded by the Ministry of Research of the Republic of Indonesia through a Basic Research Grant with contract number 12/E1/KP-PTNBH/2021.

AUTHOR CONTRIBUTIONS STATEMENT

Moehammad Sauqy Ihza Zuliandra developed the application and conducted the experiments. Tigor Hamonangan Nasution supervised the research and revised the manuscript. Ainul Hizriadi managed the cloud integration and data processing aspects. All authors reviewed and approved the final manuscript.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Moehammad Sauqy Ihza Zuliandra	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓			
Tigor Hamonangan Nasution	✓	✓		✓	✓			✓		✓	✓	✓	✓	✓
Ainul Hizriadi							✓			✓		✓	✓	✓

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

The authors declare that there is no conflict of interest regarding the publication of this paper.

DATA AVAILABILITY

The data supporting the findings of this study are available from the corresponding author upon reasonable request.

REFERENCE

[1] K. N. Swaroop, K. Chandu, R. Gorrepotu, and S. Deb, "A health monitoring system for vital signs using IoT," *Internet of Things (Netherlands)*, vol. 5, 2019, doi: 10.1016/j.iot.2019.01.004.

[2] A. Hamied, A. Mellit, M. A. Zoulid, and R. Birouk, "IoT-based experimental prototype for monitoring of photovoltaic arrays," *Proceedings of the 2018 International Conference on Applied Smart Systems, ICASS 2018*, Jul. 2018, doi: 10.1109/ICASS.2018.8652014.

[3] M. Bhatia, "IoT-inspired framework for athlete performance assessment in smart sport industry," *IEEE Internet Things Journal*, vol. 8, no. 12, pp. 9523–9530, Jun. 2021, doi: 10.1109/JIOT.2020.3012440.

[4] A. I. Siam *et al.*, "Portable and real-time iot-based healthcare monitoring system for daily medical applications," *IEEE Transactions on Computational Social Systems*, vol. 10, no. 4, pp. 1629–1641, Aug. 2023, doi: 10.1109/TCSS.2022.3207562.

[5] N. M. Abdulkareem, S. R. M. Zeebaree, M. A. M. Sadeeq, D. M. Ahmed, A. S. Sami, and R. R. Zebari, "IoT and cloud computing issues, challenges and opportunities: a review," *Qubahan Academic Journal*, vol. 1, no. 2, pp. 1–7, Mar. 2021, doi: 10.48161/QAJ.V1N2A36.
[6] S. V. S. R. Raju, B. Dappuri, P. R. K. Varma, M. Yachamaneni, D. M. G. Verghese, and M. K. Mishra, "Design and implementation of smart hydroponics farming using IoT-based AI controller with mobile application system," *Journal Nanomater*, vol. 2022, no. 1, p. 4435591, Jan. 2022, doi: 10.1155/2022/4435591.
[7] T. H. Nasution, M. A. Muchtar, S. Seniman, and I. Siregar, "Monitoring temperature and humidity of server room using Lattepanda and ThingSpeak," *Journal of Physics: Conference Series*, vol. 1235, p. 012068, Jun. 2019, doi: 10.1088/1742-6596/1235/1/012068.
[8] N. Sindhwani, R. Anand, R. Vashisth, S. Chauhan, V. Talukdar, and D. Dhabliya, "Thingspeak-based environmental monitoring system using IoT," *PDGC 2022 - 2022 7th International Conference on Parallel, Distributed and Grid Computing*, pp. 675–680, 2022, doi: 10.1109/PDGC56933.2022.10053167.
[9] G. Chandrasekaran, N. S. Kumar, A. Chokkalingam, V. Gowrishankar, N. Priyadarshi, and B. Khan, "IoT enabled smart solar water heater system using real time ThingSpeak IoT platform," *IET Renewable Power Generation*, 2023, doi: 10.1049/RPG2.12760.
[10] T. H. Nasution, A. Hizriadi, K. Tanjung, and F. Nurmayadi, "Design of indoor air quality monitoring systems," *2020 4th International Conference on Electrical, Telecommunication and Computer Engineering, ELTICOM 2020 - Proceedings*, pp. 238–241, Sep. 2020, doi: 10.1109/ELTICOM50775.2020.9230511.
[11] W. Winasis, A. W. W. Nugraha, I. Rosyadi, and F. S. T. Nugroho, "Design of a photovoltaic system monitoring system based on the internet of things (IoT)," *Jurnal Nasional Teknik Elektro dan Teknologi Informasi (JNTETI)*, vol. 5, no. 4, pp. 328–333, 2016, doi: 10.22146/jnteti.v5i4.281.
[12] M. Jankovec, K. Brecl, M. Bokalič, M. Pirc, and M. Topič, "Monitoring solar irradiance and PV module performance in mobile applications," *Solar Energy Materials and Solar Cells*, vol. 277, p. 113101, Oct. 2024, doi: 10.1016/J.SOLMAT.2024.113101.
[13] C. O. Truică, E. S. Apostol, J. Darmont, and T. B. Pedersen, "The forgotten document-oriented database management systems: an overview and benchmark of native XML DODBMSes in comparison with JSON DODBMSes," *Big Data Research*, vol. 25, p. 100205, Jul. 2021, doi: 10.1016/J.BDR.2021.100205.
[14] B. Srisungsittisunti, J. Duangkaew, S. Mekruksavanich, N. Chaikaew, and P. Rojanavasu, "Enhancing data retrieval efficiency in large-scale javascript object notation datasets by using indexing techniques," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 13, no. 2, pp. 2342–2353, Jun. 2024, doi: 10.11591/ijai.v13.i2.pp2342-2353.
[15] P. Brogan *et al.*, "Representing synchrophasor data using JSON," 2021 *32nd Irish Signals and Systems Conference*, ISSC 2021, Jun. 2021, doi: 10.1109/ISSC52156.2021.9467868.
[16] S. Y. Ameen and D. Y. Mohammed, "Developing cross-platform library using flutter," *European Journal of Engineering and Technology Research*, vol. 7, no. 2, pp. 18–21, Mar. 2022, doi: 10.24018/EJENG.2022.7.2.2740.
[17] O. M. A. AL-atraqchi, "A proposed model for build a secure RESTful API to connect between server side and mobile application using laravel framework with flutter toolkits," *Cihan University-Erbil Scientific Journal*, vol. 6, no. 2, pp. 28–35, Aug. 2022, doi: 10.24086/CUESJ.V6N2Y2022.PP28-35.
[18] S. Stender and H. Åkesson, "Cross-platform framework comparison : flutter & react native," 2020, Accessed: Jun. 16, 2024. [Online]. Available: https://urn.kb.se/resolve?urn=urn:nbn:se:bth-19749
[19] P. Tyagi, "Pragmatic flutter : building cross-platform mobile apps for android, iOS, web & desktop," *Pragmatic Flutter*, Aug. 2021, doi: 10.1201/9781003104636.
[20] N. Kuzmin, K. Ignatiev, and D. Grafov, "Experience of developing a mobile application using flutter," *Lecture Notes in Electrical Engineering*, vol. 621, pp. 571–575, 2020, doi: 10.1007/978-981-15-1465-4_56.
[21] A. Shrivastava, S. J. S. Prasad, A. R. Yeruva, P. Mani, P. Nagpal, and A. Chaturvedi, "IoT based RFID attendance monitoring system of students using arduino ESP8266 & adafruit.io on defined area," *Cybernetics and Systems*, vol. 56, no. 1, pp. 21–32, 2025, doi: 10.1080/01969722.2023.2166243.
[22] V. N. Sulistyawan, N. A. Salim, F. G. Abas, and N. Aulia, "Parking tracking system using ultrasonic sensor HC-SR04 and NODEMCU ESP8266 based IoT," *IOP Conference Series: Earth and Environmental Science*, vol. 1203, no. 1, p. 012028, Jun. 2023, doi: 10.1088/1755-1315/1203/1/012028.
[23] G. N. Mori, P. R. Swaminarayan, and R. Panchal, "Knowledge representation of sensor dataset with iot collaboration of semantic web and iot: storage of temperature and humidity details," *Recent Patents on Engineering*, vol. 19, no. 2, Feb. 2025, doi: 10.2174/0118722121242190230921070510.
[24] M. Imtiaz, M. Id, M. Azam, S. Id, R. Ayaz, and A. Id, "Extraction of use case diagram elements using natural language processing and network science," *PLoS One*, vol. 18, no. 6, p. e0287502, 2023, doi: 10.1371/JOURNAL.PONE.0287502.
[25] R. Fauzan, D. Siahaan, S. Rochimah, and E. Triandini, "Use case diagram similarity measurement: a new approach," *Proceedings of 2019 International Conference on Information and Communication Technology and Systems, ICTS 2019*, pp. 3–7, Jul. 2019, doi: 10.1109/ICTS.2019.8850978.
[26] R. AL-Msie'deen *et al.*, "Detecting commonality and variability in use-case diagram variants," *Journal of Theoretical and Applied Information Technology*, vol. 28, p. 2022, Mar. 2022, Accessed: Jun. 16, 2024. [Online]. Available: https://arxiv.org/abs/2203.00312v1
[27] R. K. Sahoo, M. Derbali, H. Jerbi, D. van Thang, P. P. Kumar, and S. Sahoo, "Test case generation from UML-diagrams using genetic algorithm," *Computers, Materials and Continua*, vol. 67, no. 2, pp. 2321–2336, Jan. 2021, doi: 10.32604/CMC.2021.013014.
[28] B. Alturas, "Connection between UML use case diagrams and UML class diagrams: a matrix proposal," *International Journal of Computer Applications in Technology*, vol. 72, no. 3, pp. 161–168, 2023, doi: 10.1504/IJCAT.2023.133294.
[29] R. N. Kulkarni and C. K. Srinivasa, "Abstraction of activity diagram from sequence diagram," *Lecture Notes in Networks and Systems*, vol. 446, pp. 145–156, 2022, doi: 10.1007/978-981-19-1559-8_15.
[30] T. Ahmad, J. Iqbal, A. Ashraf, D. Truscan, and I. Porres, "Model-based testing using UML activity diagrams: a systematic mapping study," *Computer Science Review*, vol. 33, pp. 98–112, Aug. 2019, doi: 10.1016/J.COSREV.2019.07.001.
[31] S. Al-Fedaghi, "Validation: conceptual versus activity diagram approaches," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 6, pp. 287–297, Jun. 2021, doi: 10.14569/IJACSA.2021.0120632.

BIOGRAPHIES OF AUTHORS

Moehammad Sauqy Ihza Zuliandra is an undergraduate at the Department of Electrical Engineering, Universitas Sumatera Utara. He actively helps several lecturers with research in the field of Computer Engineering. He can be contacted at email: ihzazuliandra@students.usu.ac.id.

Tigor Hamonangan Nasution is a lecturer in electrical engineering at the Universitas Sumatera Utara. He has been a lecturer since 2014. He is also a senior member of IEEE. He completed his doctor's degree in Computer Science in 2025 at Universitas Sumatera Utara. He is actively engaged in research in the field of computer engineering, particularly in the areas of embedded systems and artificial intelligence. He can be contacted at email: tigor.nasution@usu.ac.id.

Ainul Hizriadi is a lecturer at the Universitas Sumatera, Faculty of Computer Science and Information Technology, precisely in the S1 Information Technology Study Program. Not only as a lecturer, he is also trusted to join the USU Information System Center team as the implementation of network system management and arrangement, including internet bandwidth, internet networks, intranet networks, and hotspot access in the USU environment, including the provision of network facilities, including hardware and bandwidth, which are the responsibility of USU. This division is also responsible for arranging and maintaining servers used by designers, as well as storing and managing data and information in several strategic work units. His research field is related to the internet of things. He can be contacted at email: ainul.hizriadi@usu.ac.id.