Proceedings of the 5th International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/120/2025.18914

Inter-mode Development: Android Application Transition from Mobile to IoT

Yuzhe Gao[1,a,*]

[1]*Northeastern University at Qinhuangdao, Qinhuangdao, Hebei Province, China*
a. 202219033@stu.neuq.edu.cn
**corresponding author*

Abstract: The development of IoT applications based on the Android operating system is on the rise. However, the process presents greater challenges compared to traditional mobile application development, primarily due to the need to support and ensure compatibility with hardware across diverse scenarios. This complexity demands increased manpower and specialized expertise, making the promotion and reuse of Android-based IoT application development more difficult. To address these challenges, this study first reviews the existing research on IoT application development, with a particular focus on analyzing the scenarios, scope, and functional tasks associated with Android-based IoT applications, thereby identifying the current bottlenecks in the field. Secondly, it examines the typical development process, analyzing interaction methods within specific hardware scenarios to extract key coding patterns and development approaches. Furthermore, by comparing these findings with the development paradigms of Android mobile applications, the work proposes an inter-mode development paradigm for the critical development task involved in IoT application. A case study is then used to thoroughly illustrate the transformation process from mobile to IoT applications. This paper aims to provide effective strategies for reducing IoT development costs and offers recommendations for simplifying the transfer process between different devices.

Keywords: Android, IoT, Mobile app, Software development.

1. Introduction

In recent years, the Internet of Things (IoT) has emerged as a promising technology in both industry and academia, driven by the concept of connecting all devices and objects through the internet, often referred to as the “Internet of Everything” [1]. This paradigm shift toward ubiquitous connectivity allows smart devices, sensors, and systems to interact and share data, creating a seamless network of interconnected devices. IoT applications span various domains, including smart homes, wearables, healthcare, automotive systems, and industrial automation, fundamentally transforming how devices operate and communicate.

In the context of IoT application development, Android has become a significant platform due to its flexibility, widespread adoption, and open-source nature. Android IoT development supports various scenarios, including wearables, automotive systems, smart home devices, and televisions, offering a versatile foundation for building IoT applications [2]. Meanwhile, the demand for IoT application development tasks based on the Android operating system continues to rise [3]. However,

© 2025 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/120/2025.18914

in practical development, the necessity for hardware support and ensuring compatibility across diverse scenarios significantly increases the complexity compared to traditional mobile application development. This heightened complexity requires more manpower and specialized expertise, making the promotion and reuse of Android-based IoT application software development challenging [4].

To address these challenges, this study first reviews the existing work in IoT application development, identifying the current research bottlenecks. It then analyzes the interaction methods within specific hardware scenarios, extracting distinctive code patterns and development solutions. Finally, by comparing the development paradigms of Android mobile applications, it identifies key technologies essential for IoT application development tasks. This comprehensive analysis provides valuable insights into the transition from mobile to IoT applications, effectively aiding in reducing development costs and alleviating manpower demands.

2. Background

2.1. IoT Application in Android System

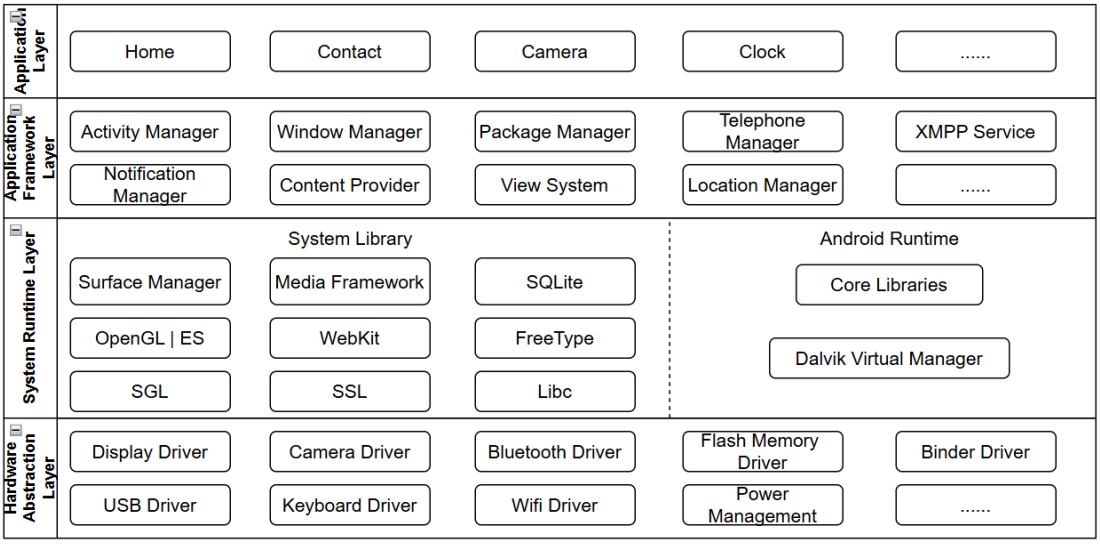

Figure 1: Android System Structure.

Android is an open-source mobile operating system developed by Google, built on the Linux kernel [1]. It offers a flexible and customizable platform for developers to create applications (apps) using programming languages such as Java, Kotlin, and C++. As depicted in Fig 1, Android's architecture consists of multiple layers: the Application Layer, Application Framework Layer, System Runtime Layer, and Hardware Abstraction Layer.

The Application Layer is the topmost layer comprising end-user applications, such as Home, Contacts, Camera, and Clock. The Application Framework Layer provides essential APIs and system services to developers, enabling the development of diverse applications. The System Runtime Layer includes system libraries and the Android Runtime, with the libraries offering fundamental functionalities to applications. The Hardware Abstraction Layer acts as an interface to the hardware, including drivers for components like displays, cameras, Bluetooth, and USB. This layer ensures developers can interact with hardware through a unified API, simplifying the development process.

Android IoT applications, such as those developed using Android Things, are also built upon this architecture. These IoT applications generally operate on embedded devices and interface with components like sensors, cameras, Bluetooth, and Wi-Fi via the Hardware Abstraction Layer.

DOI: 10.54254/2755-2721/120/2025.18914

Android's pre-existing drivers and services enable developers to create applications for smart home devices, industrial automation systems, and other IoT solutions. By abstracting the complexities of hardware communication, Android provides a robust and reliable platform for IoT development, facilitating seamless integration of diverse IoT devices.

2.2. Android IoT Application Development Study

Research on IoT spans a wide range of areas. Table 1 summarizes some of the main fields, including data security, energy consumption, hardware interoperability, and application development. These research efforts are significant for addressing challenges such as scalability, efficiency, and user privacy.

Table 1: Research in IoT Application Development.

Year	Field	Content
2020	IoT security	Resample raw bytecodes to deep networks for malware detection [5].
2017	IoT security	Propose a methodological approach for secure IoT application development [6].
2022	Hardware Interoperability	Describe various interconnections of components for home automation [7].
2021	Hardware Interoperability	Investigate the effects on IoT devices when different energy harvesters are connected [8].
2018	Application Development	Use a vehicle monitoring method to check car conditions [9].
2022	Application Development	Examine the main software development models used in IoT [10].
2021	Energy Efficiency	Propose a metaheuristic optimization approach for energy-efficient IoT architecture [11].
2021	Energy Efficiency	Use the Whale Optimization Algorithm (WOA) with Simulated Annealing (SA) to optimize energy consumption [12].

However, there remains a lack of effective solutions for migrating development from mobile to IoT platforms. In this context, we present the first attempt to identify the differences between mobile application development and IoT application development and to explore a migration pattern for transitioning from mobile to IoT applications.

3. Inter-mode Development Based on IoT and Mobile

From prior studies, it is evident that IoT development scenarios based on Android have permeated various aspects of daily life. However, numerous challenges persist in the development phase, highlighting the need for a general development scheme. This work is inspired by these challenges. In this paper, we examine the associations and differences between Android IoT development and mobile application development, and propose an inter-mode development paradigm grounded in the development relationship. The effectiveness of this paradigm is demonstrated through a case study.

DOI: 10.54254/2755-2721/120/2025.18914

3.1. Transition from Mobile to IoT

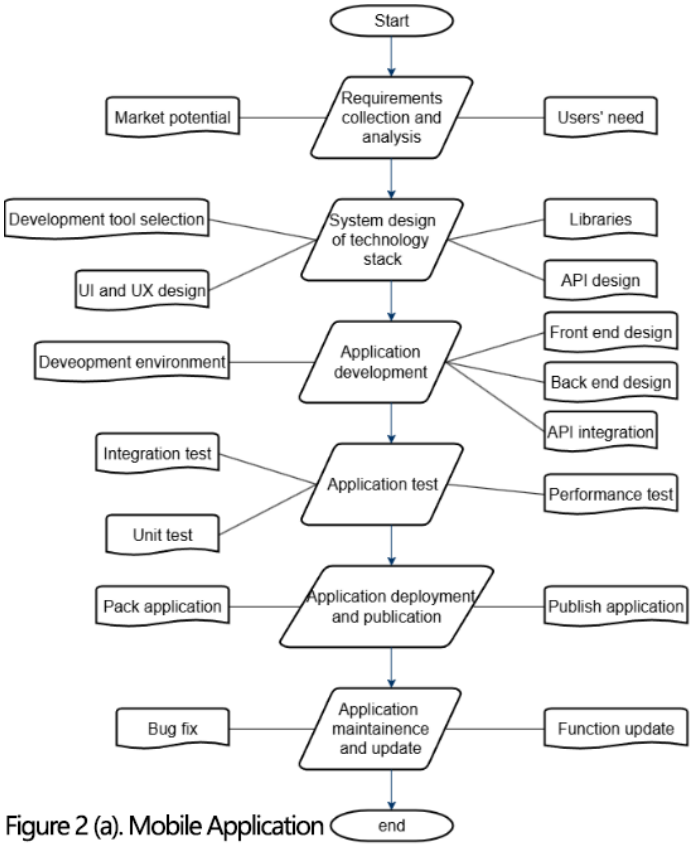

Figure 2 (a). Mobile Application

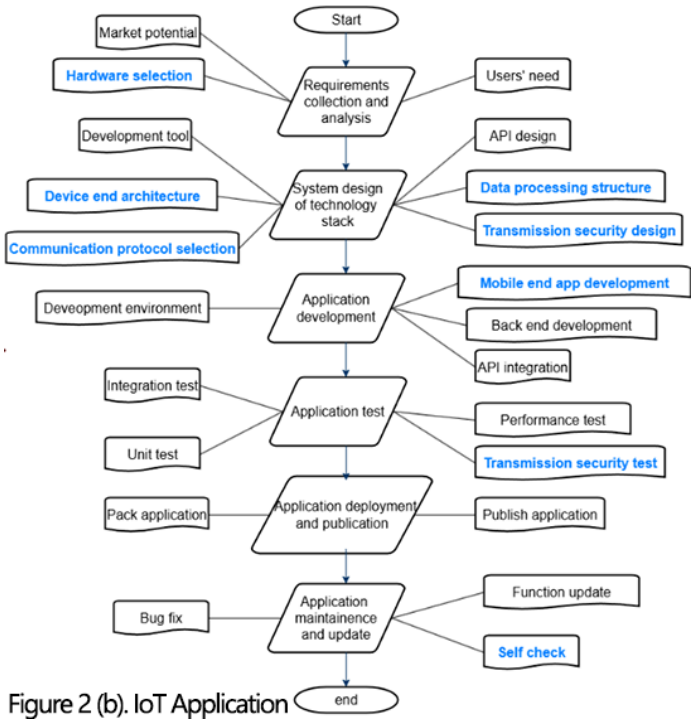

Figure 2 (b). IoT Application

Figure 2: Development Processes for Mobile and IoT Applications.

DOI: 10.54254/2755-2721/120/2025.18914

To explore the migration of Android mobile applications to IoT devices, this paper abstracts the development processes of both into six stages. While these models share similar steps, they also reveal distinct differences:

1) Requirements Analysis Phase: Mobile application developers focus on market research and understanding user needs. For IoT development, this phase also involves selecting suitable hardware to ensure the application functions correctly and meets service quality expectations. 2) System Design Phase: Developers combine user requirements to establish a technological framework. Since IoT devices frequently interact with nearby devices, secure data transmission is critical. Security mechanisms, such as RSA encryption, are essential. 3) Development Phase: Developers implement functional modules. Unlike mobile devices, many IoT devices lack display screens. Consequently, mobile or web applications become critical interfaces, enabling users to monitor device statuses, perform operations, or view data analysis results via smartphones or browsers. 4) Testing Phase: Testing ensures proper functionality, security, and efficiency. IoT devices require efficient power management, making battery life and power consumption optimization key focus areas during testing. 5) Maintenance and Updating Phase: Some IoT devices lack network connectivity for updates and log uploads, necessitating manual firmware and software updates. When errors occur, devices must detect and store logs automatically, providing detailed error feedback for manual inspection when necessary.

3.2. Analysis of the Inter-mode Development

Section 3.1 outlines the differences between mobile development and IoT development. This section delves into the distinctions between TV and mobile development from a practical application development perspective, focusing on user interfaces, layouts, input methods, notifications, and information display, as shown in Table 2.

Table 2: Development Analysis Between TV and Mobile.

Development Process	Corresponding Representative API	
UI Design	TV	androidx.tv.material3
	Mobile	androidx.compose.material3 androidx.compose.ui
Data Processing	TV	TVInputManager
	Mobile	androidx.compose.foundation.gestures
System Interaction	TV	Leaningback.GuidedStepFragment TvStorageManager
	Mobile	NotificationManager StorageManager

User Interface and Layout: Considering the differences between Android mobile and IoT development, Compose provides two library modules: androidx.tv.material3 for TV development and androidx.compose.material3 for mobile development, each tailored to their respective UI requirements. For TVs, components are optimized for large screens and remote control operations. Buttons, icons, and text are larger, with more spacing and a clearer focus state to enhance visibility and usability. For the mobile version, components are designed for touch-based devices, featuring compact sizes and spacing suitable for finger interactions.

Data Processing Structure: In Android mobile application development, input primarily comes from the soft keyboard and touchscreen. The androidx.compose.foundation.gestures module in Jetpack Compose provides various functions and components to implement and detect gestures,

DOI: 10.54254/2755-2721/120/2025.18914

enabling developers to incorporate complex touch and gesture interactions easily. For Android TV application development, input is mainly via remote controls, and the TVInputManager API manages remote control events. TVInputManager coordinates interactions between applications and the selected TV input.

System Interaction: For Android mobile application development, mobile devices often emphasize timely communication and urgent notifications. Mobile apps use the NotificationManager API to create and manage notifications, which can appear in the status bar, lock screen, or other locations. For storage, mobile apps rely on the StorageManager API to manage file storage efficiently. In contrast, Android TV application development primarily involves navigation through remote controls. Notifications are generally displayed within the app, navigated using directional buttons, resulting in relatively slower interactions. Due to the limited performance of TVs, the Leaningback library's GuidedStepFragment is used to display information or guide users. For storage management, the TvStorageManager API is employed to address TVs' limited storage capabilities.

3.3. Case Study

To better illustrate the inter-mode development discussed the previous sections. This section provides a real-world example to illustrate these issues more precisely.

We first developed an information flow app for mobile devices. As shown in Fig 3 (a), the app consists of multiple cards, each containing an image at the top, a text description in the middle, and "Mark as favorite" and "Share with others" buttons at the bottom. The app uses the Material3 library to construct the user interface and performs well on mobile devices.

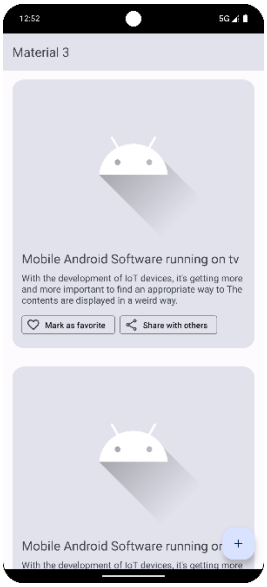

Figure 3 (a). Mobile Running Result

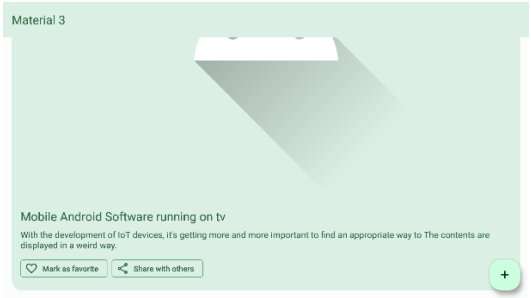

Figure 3 (b). TV Running Result

Figure 3: Application Running on Mobile and TV.

To test the feasibility of direct migration, we ran the app on a TV without modifying the code. The result, as shown in Fig. 3(b), reveals several issues. The interface appears abnormal and is difficult for users to operate. The image at the top occupies the entire screen but still cannot be displayed fully. This problem arises because the code includes .fillMaxWidth() and .aspectRatio(3f / 2f) in the image component, causing an uncoordinated layout. Additionally, the "Mark as favorite" and "Share with others" buttons are placed too close together, failing to take advantage of the larger TV screen and making it difficult for users to see them clearly from a distance. This issue occurs because FlowRow

DOI: 10.54254/2755-2721/120/2025.18914

automatically places the buttons, but its placement on TVs is suboptimal. Furthermore, Material3 is specifically designed for mobile devices, and its performance on TVs is unpredictable. The migration experiment demonstrates that, with the current Compose library, interfaces for different devices must be designed individually, as displays cannot be automatically adapted.

4. Conclusion

The growing demand for Android-based IoT application development presents significant challenges, particularly in terms of hardware compatibility and the need for specialized expertise, which hinder promotion and reuse. This paper reviews existing IoT application development efforts, identifying current research bottlenecks. It then analyzes interaction methods in specific hardware scenarios, extracting key code patterns and development approaches. By comparing these with traditional Android mobile application development, the paper highlights critical technologies involved in IoT development. A case study is used to illustrate the transition from mobile to IoT applications, aiming to reduce development costs and manpower requirements.

References

[1] *Fahmideh, M., & Zowghi, D. (2020). An exploration of IoT platform development. Information Systems, 87, 101409.*
[2] *Khan, M. A., Ahmad, I., Nordin, A. N., et al. (2022). Smart android-based home automation system using internet of things (IoT). Sustainability, 14(17), 10717.*
[3] *Gargenta, M. (2011). Learning Android. O'Reilly Media.*
[4] *Khanna, A., & Kaur, S. (2020). Internet of things (IoT), applications and challenges: A comprehensive review. Wireless Personal Communications, 114, 1687–1762.*
[5] *Ren, Z., Wu, H., Ning, Q., et al. (2020). End-to-end malware detection for Android IoT devices using deep learning. Ad Hoc Networks, 101, 102098.*
[6] *Duc, A. N., Jabangwe, R., Paul, P., et al. (2017). Security challenges in IoT development: A software engineering perspective. In Proceedings of the XP2017 Scientific Workshops (pp. 1–5). ACM.*
[7] *Khan, M. A., Ahmad, I., Nordin, A. N., et al. (2022). Smart android-based home automation system using internet of things (IoT). Sustainability, 14(17), 10717.*
[8] *Rana, B., Singh, Y., & Singh, P. K. (2021). A systematic survey on internet of things: Energy efficiency and interoperability perspective. Transactions on Emerging Telecommunications Technologies, 32(8), e4166.*
[9] *Türk, E., & Challenger, M. (2018). An Android-based IoT system for vehicle monitoring and diagnostic. In 2018 26th Signal Processing and Communications Applications Conference (SIU) (pp. 1–4). IEEE.*
[10] *Ismail, S., & Dawoud, D. W. (2022). Software development models for IoT. In 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC) (pp. 0524–0530). IEEE.*
[11] *Zanaj, E., Caso, G., De Nardis, L., et al. (2021). Energy efficiency in short and wide-area IoT technologies—A survey. Technologies, 9(1), 22.*
[12] *Iwendi, C., Maddikunta, P. K. R., Gadekallu, T. R., et al. (2021). A metaheuristic optimization approach for energy efficiency in the IoT networks. Software: Practice and Experience, 51(12), 2558–2571.*