International Journal of Research in Engineering, Science and Management
Volume-3, Issue-3, March-2020
www.ijresm.com | ISSN (Online): 2581-5792

Portable Customised Android OS for Different Computing Devices

Siddhesh Tawade[1], Shrunkhal Shringarpure[2], Swapnil Waghmare[3]
[1,2]*Student, Department of Computer Engineering, Pillai HOC College of Engineering and Technology, Kharghar, India*
[3]*Assistant Professor, Department of Computer Engineering, Pillai HOC College of Engineering and Technology, Kharghar, India*

***Abstract*: Android is a software stack that includes operating system, middle ware, applications for the development of devices. Android has evolved greatly and user experience in addition to consumer level efficiency along with integration of android powered devices also expanded. Because of its promising features and characteristics like open source nature, rich user interface, consistent app API's. Android is being integrated and ported to various computing devices this includes smart phones, tablet, and google pixel book. One major advantage of using android framework beyond the mobile devices is the android applications can talk to the functionality of all these devices powered by android and developers need not to write several applications for different computing systems. In this module we will develop customizable AOSP then it can be transported into new devices like smart phones and tablet once all these has been done android step forward into new platform can be achieved. We are also using a generic system image (GSI) which is a system image with adjusted configurations for android devices. It's considered a pure android implementation with unmodified Android Open Source Project (AOSP) code that any android device running Android 8.1 or higher can run successfully. GSIs are used for running VTS and CTS-on-GSI tests.**

***Keywords*: Software stack, Market share, Android migration, AOSP, Step forward.**

1. Introduction

Android is an open source operating system for mobile devices and a corresponding open source project led by Google. This site and the Android Open Source Project (AOSP) repository offer the information and source code needed to create custom variants of the Android OS, port devices and accessories to the Android platform, and ensure devices meet the compatibility requirements that keep the Android ecosystem a healthy and stable environment for millions of users. Android provides complete software platform and framework rather than just operating system for mobile devices and now-a-days it also creates possibilities of using in a much wider range of devices. Practically the android architecture consists of application framework on top of Linux based kernel and it is java based along with free licensing this facilitates its rapid deployment in many domains, allowing adopters to add additional proprietary value in the android source. Android based smart phones Surpassed Apple iPhone shipments in various places of the world, clearly showing how Android is open for innovation. Any smartphone manufacturer which sells an Android smartphone is using the Android Open Source Project. And to be honest, virtually anyone making a smartphone today that isn't an iPhone is leveraging the AOSP code. This includes Samsung, LG, HTC, Huawei, Xiaomi, ZTE, Honor, OnePlus, and many others. So it doesn't matter if it's Samsung's version of Android (which is called Samsung Experience), or Xiaomi's version of Android (which is called MIUI), Huawei's version of Android (which is called EMUI), HTC's version of Android (which is called Sense UI) or even Google's version of Android. All of these different versions of Android, feel, and perform drastically different but they are all based upon the same set of code that is known as AOSP. As I mentioned, Google maintains the codebase for Android and then releases those changes once a year to the Android Open Source Project repository. This allows all of these smartphone OEMs to start with a clean base of code and then add on their changes onto it. This could be but is not limited to changing how it looks, adding/removing features, and in some case actually changing the fundamental way that software components interact with each other. Companies such as Samsung, LG, HTC, Huawei, Xiaomi and others don't have any advantages over you when it comes to what they do with the original code of the Android Open Source Project. Now, Google does work closely with some of these companies. This is done to get those major Android version updates pushed out to their customers faster. But once the AOSP source code drops you can make your own version of Android if you would like. And that is actually what a lot of custom ROMs are based off of. If you've dabbled in the custom ROM scene for any length of time, then you have most likely heard of Lineage OS. This custom ROM was previously known as Cyanogen Mod and it's actually what the majority of the other custom ROMs are based on. The developers and maintainers of Lineage OS use AOSP as the base of their custom ROM. These developers then add in features to the software that they want to see and that is why the custom ROM scene is as popular as it is among the Android enthusiast crowd. Custom ROMs like Lineage OS have their own set of features

built into them and that list usually grows bigger with ROMs which are based on it. For example, Resurrection Remix is based on Lineage OS but the developers want even more features. So they add those extra features into the operating system and some are very happy with this. Others will say these types of custom ROMs.

2. Literature review

When you buy a brand-new Android device, it comes equipped with a "stock ROM," also known as the "stock firmware". The stock ROM is the pre-installed operating system on your phone. The stock ROM has limited functionalities that are that are defined by the phone's manufacturer. If you wish to have extra features added to your device, then you will have to resort to custom ROM. Google has the final say when it comes to what is and is not accepted into the Android Open Source Project repository. Naturally, they just can't add in everything in one release so they usually start with a vision that they have for the next big version of Android. From here, they begin working on adding in a certain number of APIs (usually just one or two) to the AOSP codebase. These APIs are not only available to Google and smartphone OEMs, but they are also open to 3rd-party developers for applications and games that you see in the Play Store. It may seem like most of the new features of the brand new version of Android is based on the newly added API, that is usually not the case. For example, Android 9 Pie added 1 API to AOSP (which is the 28th API available to developers) that adds Indoor positioning with Wi-Fi RTT. Out of the box, your phone or tablet's hardware is clocked at a speed that the manufacturer considers optimal in terms of heat and battery life. On a custom ROM, you'll be able to overclock your hardware to get big performance increases. These changes can really show when you're playing graphics intensive games. Another way a ROM can improve performance is by removing carrier or OEM-installed apps, known as bloatware, which can free up system resources. However, Android 9 Pie got a slew of new features added to the Android Open Source Platform including. . .

- New user interface for the quick settings menu.
- The clock has moved to the left of the notification bar.
- The "dock" now has a semi-transparent background.
- Battery saver no longer shows an orange overlay on the notification and status bars.
- A "screenshot" button has been added to the power options.
- A new "Lockdown" mode which disables biometric authentication once activated.
- Rounded corners across the UI.
- Support for display cutouts.
- Redesigned volume slider.
- Battery percentage now is shown in Always-On Display.

3. Implemented technique

A. System Architecture

1) Linux Kernel

The foundation of the Android platform is the Linux kernel. For example, the Android Runtime (ART) relies on the Linux kernel for underlying functionalities such as threading and low-level memory management. Using a Linux kernel allows Android to take advantage of key security features and allows device manufacturers to develop hardware drivers for a well-known kernel.

2) Hardware Abstraction Layer (HAL)

The hardware abstraction layer (HAL) provides standard interfaces that expose device hardware capabilities to the higher-level Java API framework. The HAL consists of multiple library modules, each of which implements an interface for a specific type of hardware component, such as the camera or bluetooth module. When a framework API makes a call to access device hardware, the Android system loads the library module for that hardware component.

3) Android Runtime

For devices running Android version 5.0 (API level 21) or higher, each app runs in its own process and with its own instance of the Android Runtime (ART). ART is written to run multiple virtual machines on low-memory devices by executing DEX files, a bytecode format designed especially for Android that's optimized for minimal memory footprint. Build toolchains, such as Jack, compile Java sources into DEX bytecode, which can run on the Android platform.

Some of the major features of ART include the following:

- Ahead-of-time (AOT) and just-in-time (JIT) compilation
- Optimized garbage collection (GC)
- On Android 9 (API level 28) and higher, conversion of an app package's Dalvik Executable format (DEX) files to more compact machine code.
- Better debugging support, including a dedicated sampling profiler, detailed diagnostic exceptions and crash reporting, and the ability to set watchpoints to monitor specific fields

Prior to Android version 5.0 (API level 21), Dalvik was the Android runtime. If your app runs well on ART, then it should work on Dalvik as well, but the reverse may not be true.

Android also includes a set of core runtime libraries that provide most of the functionality of the Java programming language, including some Java 8 language features, that the Java API framework uses.

4) System Apps

Android comes with a set of core apps for email, SMS messaging, calendars, internet browsing, contacts, and more. Apps included with the platform have no special status among the apps the user chooses to install. So a third-party app can become the user's default web browser, SMS messenger, or even the default keyboard (some exceptions apply, such as the system's Settings app). The system apps function both as apps

for users and to provide key capabilities that developers can access from their own app. For example, if your app would like to deliver an SMS message, you don't need to build that functionality yourself—you can instead invoke whichever SMS app is already installed to deliver a message to the recipient you specify.

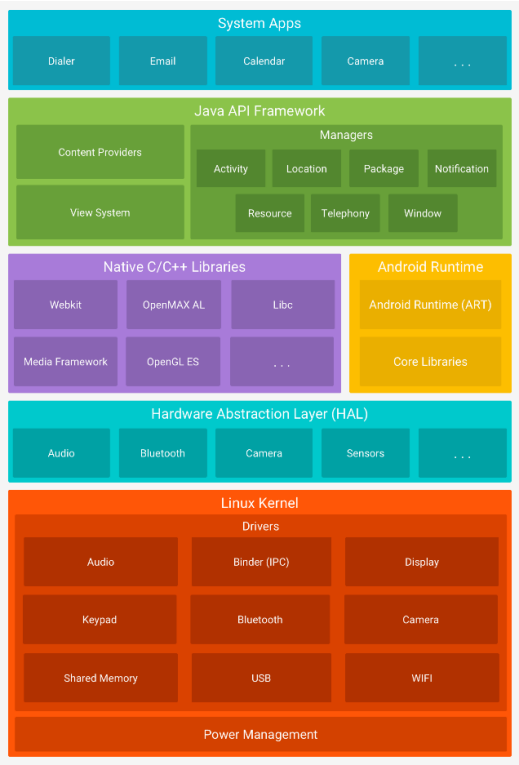

Fig. 1. System Architecture

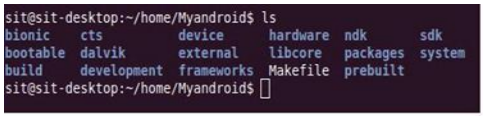

Fig. 2. Sub-folder in AOSP folders

Historically Android has been widely deployed in mobile phones such as smart phones and tablets. Now the android is grown-up with more patches and powerful versions of android has come up with support for new kind of things, as well as more strong multimedia processing, android is escalating beyond its origins and started migrating [11] into a wider array of applications and products. As a result, an increasing number of developers are educating them self on android, with advances in android and kernel development we got curiosity to explore some of its benefits and challenging issues as well as new possibilities it presents they are,

- For migrating android system to the many embedded devices we have to be aware of the differences between the Linux and android.
- The Linux kernel driver policy for android is to avoid modules and it runs on a static hardware configuration unlike Linux distributions that can run on a most hardware as possible.
- We need to ensure that the kernel is appropriate with the target hardware and with android, and thus ensure suitability between android and the hardware [7].
- Linux kernel for an android needs to be compiled according to the hardware of the target device in which we want to build android system [8].
- Besides the kernel and drivers, the GUI of android needs to be adaptable with the target platform.
- Accordingly, the changes made in the android kernel and codes for the hardware emulator do not work well with actual hardware.
- Creating patches so that the Linux kernel is attuned for android and android intern is tailored for the mobile phone or other devices [9].
- Once you have the Linux kernel ported, it is a fairly easy to get the rest of the android system in place and then you can take advantage of a enormous number of the available android apps that are existing in the community.
- Some functionality like Ethernet, AM/FM radio is not supported by default in android stack.
- Modularity is not included in stack designing so that packages and package features can't be changes easily.
- Android stack has a strong dependency on hardware of a device. It is evident that the largest dependency is on the touch screen as user interacts via the touch screen. A majority of the embedded systems still does not contain any display [12].

4. Result and Analysis

A. Chat Bubbles

Fig. 3. Chat Bubble Feature

Chat bubbles first appeared back with Android Q beta versions, but Google decided to remove them in the stable release for Android 10. With Android R's first dev beta, they have made a reappearance with the same concept. Basically,

with chat bubbles, messaging apps can show conversations in floating bubbles, similar to Facebook Messenger's Chat Heads feature that a lot of people like to use. Theoretically, chat bubbles can be used by any messaging app, but as of right now, only the Google Messages and Gmail apps are using the feature. You can open a chat bubble for a conversation by simply long-pressing on a message notification and tapping on 'Show in chat bubble'. You can then reply to conversations in a floating window similar to Messenger's implementation, and it even has support for Smart Reply. Following Figure shows chat bubble feature.

B. Native Screen Recorder

Another feature that first popped up last year with Android Q only to later be removed before the stable release of Android 10 screen recorder is back this time, with an improved UI. In Android 11, you can access the built-in screen recorder functionality from the Quick Settings panel. Tapping on this starts a 3 second timer before the screen starts being recorded. There's also a handy notification to stop, pause, or cancel the screen record. Following Figure shows Screen Recorder feature.

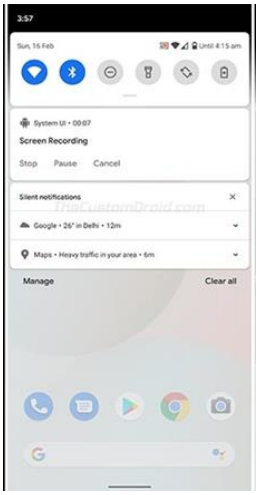

Fig. 4. Native Screen Recorder Feature

C. Dark Mode

You must have seen the myriad of leaks about the system-wide dark mode coming in tow with Android Q and it's true. However, the first beta build doesn't carry an option to enable the same, instead, it has even removed the dark theme option which we have available in Android P. Following Figure shows Dark Mode feature.

D. Gesture Control

You had all been expecting the back button to vanish and Google completely relying on the pill in Android Q, but the Beta 1 build doesn't bring the same in tow. The navigation gestures have been retained in their current form, the same as Android 9 Pie – except for one minor addition that makes switching between apps easier. Google seems to have taken cues from iPhone X's gestures, which are super good and fluid, and adding a new gesture to switch between apps. You can still use Android Pie's right flick on the pill to jump between recent apps but it's janky and slow. This new arc-like gesture on the other hand, where you slightly pull up on the pill and swipe to the right, is snappier because of fewer animations. Following Figure shows Gesture Control feature.

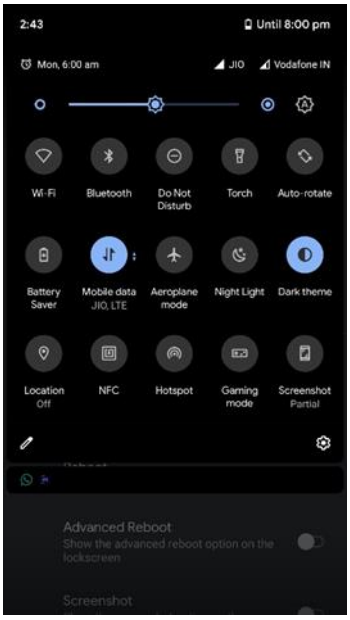

Fig. 5. Dark Mode Feature

Fig. 6. Gesture Navigation Feature

E. Theming Option

It now comes with a dedicated theming section that allows you to change the accent color (pick between blue, black, green & purple), the font (currently just two options), and the icon shape, which was possible in Android P as well. Following Figure shows theming option feature.

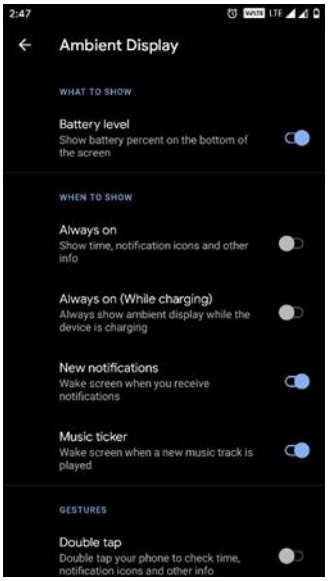

Fig. 7. Theming Option Feature

Following table shows the comparison between existing technique and implemented technique:

Table 1
Comparison between existing technique and implemented technique

Features	Existing Technique	Implemented Technique
Chat Bubbles	No	Yes
Screen Recorder	Yes	Yes
Dark Mode	No	Yes
Gesture Control	No	Yes
Theming Option	No	Yes

5. Conclusion

In this paper we look at challenges and benefits of android migration into new embedded devices and we presented general procedures to port android to ARM and non-ARM boards including partial view about How to customize Android source for preparing it to port onto new devices. The success of android was only possible due to easy portability of Linux to various hardware platforms but android is a watered-down, limited Linux distribution [4] that works for one specific purpose and Linux kernel inside was always unaltered we need to apply some patches to add features that we want for our new device [11]. The customizing ability of AOSP and also as an embedded android with Linux kernel it provides new possibilities [10] for porting into many more devices. Creating androidism in the kernel for different target hardware's we can definitely make android to support for new kind of devices other than mobiles [5].

References

[1] Marko Gargenta, “Learning android,” first edition O’Reilly publications, 2011.
[2] Karim Yaghmour, “Embedded Android,” first edition O’Reilly publications, 2013.
[3] Official google android website developers.android.com
[4] Al-Rayes, Hadeel Tariq. Studying Main Differences between Android &Linux Operating Systems. In. Proc: International Journal of Electrical & Computer Sciences
[5] Colin Walls, Mentor Graphics Android development for embedded systems beyond mobiles.
[6] Marko Gargenta founder and CEO at Marakana Learn about Android internals and NDK.
[7] Darren Etheridge, Android Multimedia Engineering Manager, DSP and ARM Texas Instruments Developing Android applications for ARM Cortex-A8 cores.
[8] Soumya Kanti Datta, “Mobile Communication Department EURECOM Sophia Antipolis,” France Android Stack Integration in Embedded Systems.
[9] Abhyudai Shanker, “Android porting concepts,” in. Proc. Electronics Computer Technology (ICECT), 2011 3rd International Conference, vol. 5, 2011.
[10] Loc Perneel, et al., “Can Android be used for real-time purpose,” International conference on computer systems and industrial informatics, pp. 1-6, December 2012.
[11] Igor Kalkov, Aachen, Dominik Franke, “A real-time extension to the Android platform,’ Proceedings of the 10th International Workshop on Java Technologies for Real-time and Embedded Systems publication, pp. 105-114, 2012.