ISSN 22773061

INVESTIGATION, FORMULATION AND DEVELOPMENT OF AN OPEN GUI FOR THE TOUCHSCREEN SMARTPHONE

Daniel Sooknanan, Dr. Ajay Joshi
Department of Electrical and Computer Engineering,
University of the West Indies, St. Augustine, Trinidad and Tobago
daniel.sooknanan@my.uwi.edu
Department of Electrical and Computer Engineering,
University of the West Indies, St. Augustine, Trinidad and Tobago
ajay.joshi@sta.uwi.edu

ABSTRACT

The use of touchscreens in handheld mobile devices, including mobile phones, PDA's, media players and tablet PC's, has rapidly increased in recent times. One of the most important aspects of these devices is the software which comprises the Graphical User Interface (GUI). This paper is centered on designing and implementing an open source GUI for a touchscreen smartphone, with the overall intent being to verify that the use of open source development tools can reduce the cost of production and by extension the cost of smartphones on the market. The methodology adopted to achieve successful completion of this research included comprehensive background research into existing GUI design theory and mobile usability studies, and applying these findings to the design of an open source GUI, within the constraints of an Embedded Linux target platform. The major outcomes of this study include the successful formulation and design of a hierarchical, touchscreen GUI suitable for a smartphone, as well as successful development and target-specific implementation of this GUI on an Embedded Linux, ARM-based platform, functioning as a hypothetical smartphone. After successful implementation of the GUI, it can be said that the adoption of an open source philosophy in the design of a smartphone GUI can reduce overall cost.

Indexing terms/Keywords

Graphical User Interface (GUI); Open Source; Human Machine Interaction (HMI); Embedded Linux; Qt; ARM.

Academic Discipline And Sub-Disciplines

Computer Engineering, Software Engineering

SUBJECT CLASSIFICATION

Mobile Computing and Applications, Open Source Tools

Council for Innovative Research
Peer Review Research Publishing System
Journal: INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY
Vol 10, No.8
editor@cirworld.com
www.cirworld.com, member.cirworld.com

1. INTRODUCTION

With the proliferation of touchscreen smartphones in the mainstream market today, one of the key aspects which differentiate these devices and separate the sheep from the goats is the graphical user interface (GUI). One of the main factors which can influence a user's choice of smartphone OS is the GUI. As the primary means of human-machine interaction in touchscreen smartphones, the GUI (being interacted with via the touchscreen itself) should boast such qualities as usability, overall flexibility, innovation, visual appeal and uniqueness.

The issue with this however rests in the associated costs of developing such a GUI. Therefore, the question to ask oneself is: can the use of open source tools in the development of an open GUI decrease the overall cost of touchscreen smartphones? This research is essentially concerned with the investigation, formulation and development of an open graphical user interface for the touchscreen smartphone. To support this hypothesis of the correlation between open source tools and cost, the actual design and development of a smartphone GUI was carried out using open source development tools, where all the code and tools used are open and where the final version of the software will be available as open source software.This was used as the basis for comparison to existing open source smartphone platforms.

Factors which influence the cost of smartphones in the high-end market include: the vast feature sets coupled with performance (defined by speed and power efficiency), the actual tools used for production and the bill of materials to achieve said performance/features[27].

In deducing the role that the open source philosophy has to play in terms of pricing, Android smartphones were considered. The Android operating system (OS), which is the market-leading smartphone OS today, is the only major, widespread open source OS, which by extension implements an open GUI [7]. However, it is no surprise that Android smartphones are at the upper echelon of the price range (attributed to the bill of materials to achieve cutting-edge performance). This therefore emphasizes the fact that both software and hardware contribute to cost.

With this being said, in addition to the use of open source development tools, a low-cost implementation is achieved using an Embedded Linux platform as the target platform. This inherently reduces development costs since Embedded Linux is licence free [30]. Furthermore, the target platform contains an ARM processor, which effectively reduces power consumption and enhances the level of integration and dependability [26].

Perhaps the greatest benefit of the open GUI implemented here is the possibility of expansion and customization by the user. While the traditional smartphone is fixed at the factory, with the open source GUI developed here, all code is available; therefore, users can customize their own smartphone GUIs and Linux operating systems/kernels, which greatly improves flexibility [26]. While Android may potentially allow this, this product differs in that it is low cost, yet powerful and stable.

Overall, the work presented here seeks to implement a low-cost, usable GUI that can be integrated with an existing smartphone platform to effectively penetrate the open source smartphone market. The bulk of the research rests in the formulation and planning of a hierarchical menu interface for a hypothetical device, and consequently implementing, testing and verifying this system on a 32-bit ARM-based device. Finally, upon completion, the tested version of the software will be released and available as open source software.

2. GUIs FOR TOUCHSCREEN SMARTPHONES

Before delving further into the actual design, an overview of GUI design theory is given. First off, touchscreen GUIs are dictated by the device's operating system. Currently existing, popular and market leading GUIs include those associated with: Google's Android OS, Apple's iOS, Microsoft's Windows Phone and RIM's Blackberry OS [7].

2.1. Gui Design And Layout

The mobile realm has several constraints which UI designers must take into account when designing and developing GUIs. In particular, the form factor is of great importance since mobile screens are relatively small. As such, a designer should only include critical functions and content of an application, and these should be laid out strategically in the available screen space [24].

Since users are expected to interact with the GUI via fingers or styli, it is recommended practice to design an interface for use with both fingers and styli – this enhances overall usability and flexibility of a system [12].User interface controls (e.g. icons, push buttons) should be of adequate size to capture fingertip/stylus actions [28]. In terms of icons used to launch applications from the smartphone menu, the ISO/IEC in[11] states that for touchscreen devices accessible by stylus pen or finger, all icon graphics should be displayed with a resolution of 32 × 32 px or higher.

Another point of interest to keep in mind is the areas where fingers typically come to rest on the device, in particular the thumbs. These are known as easy reach regions and can be considered areas of high activity. A user's thumb can effectively sweep the entire screen, but some regions require extension of the thumb. This should be kept in mind when considering button layout – frequently used buttons and primary controls should be placed at the bottom of the screen for easy tapping. Additionally, this allows for lack of obstruction of vision by the fingers, known as occultation [6].

2.2. Input Methods – Human Machine Interaction (HMI) and the GUI

The touchscreen interface offers a new dimension of human-machine interaction by enabling a different input mechanism. With the absence of a physical keypad, human-device interface problems can arise [16]. There must be a means of navigating content. This leads to the concept of various gestures to enhance interactivity and usability [24].

Some of the core touch gestures as outlined in [25] which can be implemented on both single-touch and multi-touch surfaces as a means of HMI include: tap, double tap, drag, flick/swipe and press. In general, the gestures used during design should depend on the type of application being developed. Within the touchscreen interface environment, the speed and ease of human interaction is heightened; therefore, the interface's responsiveness should be of utmost importance. Also, in terms of HMI, an ambidextrous design must be considered. In delivering the same unique experience to all users, designers should consider vertically symmetrical design, which can further simplify the interface and enhance flexibility.

2.2.1. Keyboard/Keypad

In a truly touchscreen smartphone, there is no physical keypad; as such, there must be the inclusion of a virtual/software keyboard to compensate for textual and numerical data input. Typically, touchscreens are not well suited for data entry [16]. On the single touch touchscreen, data input is sequential, which can effectively slow down the typing process.

Reference [16] advises that a virtual QWERTY keyboard should be provided for textual input. The de facto standard position for displaying the keyboard is at the bottom of the screen. Also, the size and position of keys affect accuracy and input speed [15]. When designing the QWERTY keyboard, ensure that the activation area for each interaction element is as large as the corresponding visual representation to prevent user mishits [16].

2.3. Review of Open Source GUIs

In keeping with one of the central themes of this paper, the effect of the use of open source development tools and an Embedded Linux target platform on the overall cost of production of a smartphone is investigated. When reviewing the graphical user interface of a smartphone, an important factor to note is the particular source model of the OS, i.e. whether it is closed source or open source. Widespread open source smartphone GUIs include those associated with Android OS, Nokia's Symbian OS and Nokia's Maemo OS.

Upcoming open source operating systems expected to be released in 2013 include Canonical's Ubuntu Touch, Mozilla's Firefox OS, Tizen OS, OpenWebOS and Sailfish OS [23], [29]. All of the aforementioned operating systems, with the exception of Symbian, are Linux-based and are therefore designed around an open source philosophy. In order to obtain a better understanding of open source software GUIs in general, consider the following synopsis of open software conditions.

What is open source software? Essentially, open source software is software distributed with a license that allows access to its source code, free redistribution, the creation of derived works, and unrestricted use [1]. This is achieved via the use of open source licenses which ensure freedom to everyone [19]. The Open Source Initiative in [18] states that the definition of open source is not necessarily limited to access to source code but must also comply with several criteria, including integrity of the author's source code, no discrimination against persons, groups or fields of endeavor, the license not restricting other software and the license being technology neutral.

3. METHODOLOGY

This section of the paper seeks to show the application of the aforementioned information to the formulation of a GUI for the hypothetical smartphone. It also seeks to assess the feasibility of adopting an open source philosophy in the development of the GUI as it impacts on the cost of development. In doing this, the platforms used for development will be introduced and justified accordingly.

3.1. Justification of the Hardware Platform

With the vast feature set that smartphones are now capable of handling, there must be some part of the system that can process and carry out these functions accordingly. This is where the processor of a smartphone comes into play. Inside a smartphone, processors actually refer to the System-on-a-Chip (SoC), which is essentially a combination chipset that incorporates all critical components of a device in a relatively small area [22].

Of particular importance however, is the processor core inside the SoC. According to [5], [14], nearly every smartphone on the market contains a CPU based on a processor core design from ARM Limited. ARM processors are the most popular processors in embedded architecture today, due to their unique combination of features, which allows for minimized energy consumption, high code density, a small core size and power efficiency[9].

Since the majority of smartphones today are equipped with ARM cores, it was desired to utilize a platform containing an ARM processor, thus minimizing energy consumption and increasing the level of integration in the mass market. As mentioned previously, an Embedded Linux platform was preferred due to the reduced developmental costs of a free license and the flexibility in development. The use of Embedded Linux also purports the concept of the open source philosophy, as will soon be explained.

In selecting an appropriate development platform that met the requirements stated previously, it was decided that the FriendlyARM Mini2440 development board would be used to emulate the smartphone platform. The Mini2440, shown in Figure 1, is an Embedded Linux Single Board Computer (SBC) based on the Samsung S3C2440 microprocessor [13].This microprocessor was developed as per the ARM920T processor architecture, which is a member of the ARM9TDMI processor family, adopting the ARMv4 architecture version [13], [20], [21].

The Mini2440 features a 3.5 inch, 4-wire TFT LCD resistive touchscreen interface with a 320×240px resolution. In terms of memory, the device uses both SDRAM memory and Flash memory [13].The memory module is interesting to note since the 64M NAND Flash serves as the hard disk, i.e. it is used for the storage of system boot, OS Kernel, file system,

graphical user interface and application programs [26].

Figure 1: FriendlyARM Mini2440 Development Board

In terms of interfaces used in the programming process, the Mini2440 is equipped with a USB Host port, a USB slave B port, and an RS-232 DB-9 serial port. The Mini2440 also has six user buttons and four user LEDs as a means of HMI [13].

3.2. Justification of Software Platform

When considering viable software platforms that could be used to design a rich, yet open graphical user interface, the choices were narrowed down by taking into account the constraints set by the development board, that is, which platforms were compatible with the Mini2440. Being an Embedded Linux device, the board is preinstalled with a Linux kernel – Linux 2.6.29. Embedded Linux implies the adoption of the Linux distribution in an embedded device. Since no specific kernel is written for embedded devices, the Linux kernel source code (available online via open source repositories) can be built on various platforms [8]. The GUI will be developed using this kernel.

In selecting an application framework that is compatible with Embedded Linux and which is in itself open and will give rise to open software, it was decided that Qt would be used to develop the GUI for the touchscreen smartphone. Qt is a comprehensive C++, cross-platform application development framework for creating cross-platform GUI applications [4]. Qt uses C++ language constructs and syntax with added macros for rich GUI functionality. Available under dual licensing, Qt can be used to develop both commercial applications and open source programs. The open source version of Qt is available under the GNU General Public License (GPL) – in particular, the GNU Lesser General Public License (LGPL) v2.1 [4]. Using the open source version of Qt reiterates the objective of using open tools.

4. DESIGN

Applying the necessary design theory, and given the constraints of the Mini2440, the GUI was formulated and developed using Qt. For development with the Mini2440, a Linux-based Operating System is required. The Linux distribution chosen was Canonical's Ubuntu which reiterates the notion of using open tools.

4.1. Planning of the Hierarchical Menu System

In designing the GUI for the touchscreen smartphone, it was necessary to devise a hierarchical menu system. This can be visualized as different applications in the system existing at different levels or nodes in a "tree" structure, with applications in the upper layers calling applications in the lower layers. The level of a particular menu interface is determined by its appearance in the GUI and the manner in which it is launched. When the GUI is engaged, the first screen the user is greeted with will be at the highest level in the hierarchy. Subsequent applications which can be called from this screen will then be at lower levels in the hierarchy, and so forth.

Formulation of this hierarchy allows for a more structured approach to design. It was decided that the home screen of the GUI would be the root node. Within this home screen there would be docked applications and an icon to launch the main menu of the GUI. These are therefore children of the homescreen.

ISSN 22773061

From this approach, it may be said that the main menu and the docked applications all exist at the second level in the hierarchy. However, due to the nature of the main menu and its ability to launch these home screen applications as well, the applications are also children of the main menu. Therefore, the main menu inhabits the second level and the docked applications exist at the third level in the tree. The third level in the hierarchy will also consist of other native applications that can be launched from the main menu, as well as sub-menus.

Finally at the fourth level in the hierarchy are applications which can be launched from the sub-menus. In this particular implementation of the GUI, there are two sub-menus, Games and Multimedia. The hierarchy of the GUI is illustrated in Figure 2.

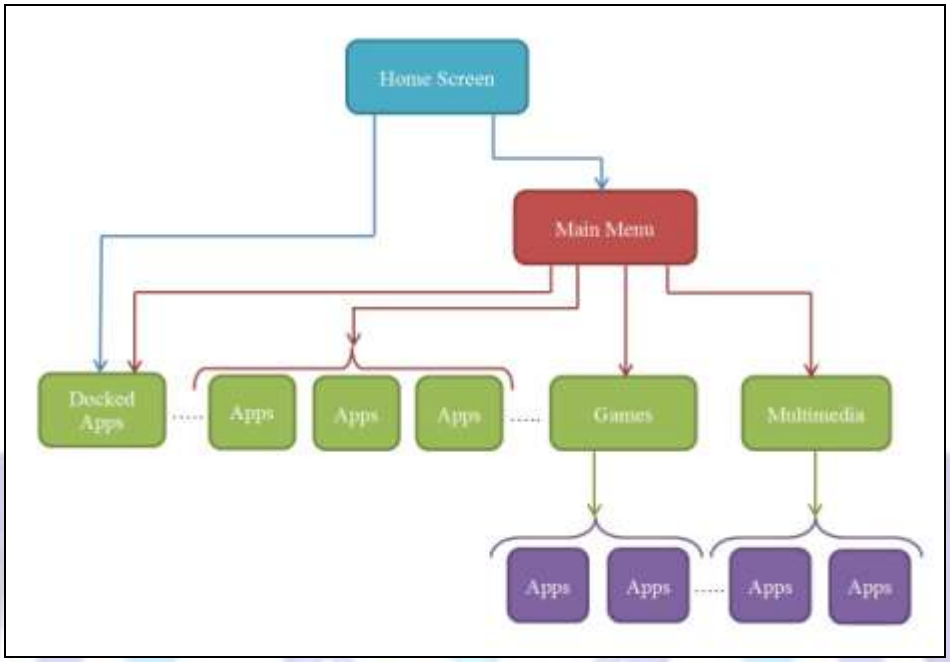

Figure 2: Touchscreen GUI Hierarchy

4.2. GUI Layout and Design

During the development of the GUI and corresponding applications, the constraints of the Mini2440 must be taken into account, in particular, the screen resolution (240 × 320 px) and the suitability to stylus input. One important point to note is that the Mini2440 contains a resistive touchscreen, which implies that only single-touch gestures can be used during design. Also, no physical buttons were used in this design, therefore, navigation must be facilitated on-screen.

Due to the lack of an accelerometer, the GUI will be developed solely in the portrait orientation, as automatic screen reorientation is not supported. The following sub-sections delineate some important aspect of design.

The outcomes of the following design strategies will all be illustrated in the final results, therefore the blueprints for their design will not be investigated in much detail.

4.2.1. Indicators

One aspect of the GUI which is important in any smartphone is that of phone status indicators. Indicators will populate the top of the screen, as is the standard for most mobile phones. Indicators implemented in the design include: battery indicator, antenna level indicator, Wi-Fi antenna indicator and a digital clock. It should be noted that these indicators, with the exception of the digital clock, are not actual reflections of on-board statistics and are thus implemented as images placed on screen to add authenticity. Due to the form factor of the Mini2440, an area of 240 × 20 px was dedicated to status indicators.

4.2.2. Home Screen

The home screen of a smartphone is the first interface a user is greeted with when the GUI is engaged and is at the topmost level of the hierarchy. The home screen will consist of a background image, a digital clock, phone status indicators and four icons at the bottom of the screen (docked applications and menu launcher) Recall that it was recommended that primary controls be placed at the bottom of the screen; this approach was adopted in placing the home screen icons, and allows for easy reach. Vertical symmetry allows for an ambidextrous design.

4.2.3. Menus

When planning the menu system for the hypothetical touchscreen smartphone, it was decided that the icons would be laid out according to the frequency of use of features, i.e. from most used to least used. This will entail the strategic placing of icons on convenient locations on-screen. Unlike in traditional feature phones, where the purpose of a mobile phone was

primarily for making calls and SMS messaging, with smartphones there has been a paradigm shift, due to the wide range of features with which smartphones are now equipped. The frequency of use of these features was determined by mobile usability studies conducted in [17].

For a GUI to be an efficient means of HMI, the use of icons is imperative. For the menu system implemented, each application was represented by an icon 44 × 44px in size, as per the guidelines stipulated in [2]. Each icon was selected based on the relevance to the application it represented. Within the Qt IDE, in order to obtain a grid formation of the icons so that spacing was even between adjacent icons and between icons and their labels, layouts were implemented in Qt.

Within the hierarchy of the GUI, there are also sub-menus. When the respective icons are selected from the main menu, the Multimedia and Games sub-menus are opened.

4.2.3. Dialling Interface

This interface appears when a user is making a call and is launched by touching the “Phone” icon from either the Home Screen or Main Menu. It consists of a dial pad, containing number “1” through “9”, “0”, “*” and “#”. The numbers on the dial pad also contain associated letters, as outlined by the International Telecommunications Union [10]. Additionally, this interface contains a Call button, an End button and a Backspace icon. The dial pad as a UI entity on its own is part of a sequential task. According to Microsoft Inc.’s Windows Phone UI Design and Interaction Guide, for sequential tasks, the UI element should be larger 9mm, in order to prevent touch errors.

After dialing, when Call is pressed, the user is taken to another screen, the Call Screen, which displays the number previously dialed, and which simulates a phone call. To facilitate HMI during calling, the LEDs of the Mini2440 were used in this application by starting the LED service in Qt when the Call button is pressed and stopping the LED service when the call is ended.

4.2.4. Virtual Keyboard

For the sake of textual input, for example when composing messages, a virtual keyboard had to be included. A standard QWERTY keyboard was designed and placed at the bottom of the screen. The keyboard consisted of numerous push buttons, representing letters of the alphabet, a Spacebar, a Backspace button, a Shift button and an additional number/symbol button which changes the QWERTY keyboard to an array of numbers and symbols. The dual nature of the virtual keyboard allows for textual, numerical and symbolic input. In this implementation, the virtual keyboard was used in the Messaging application.

5. RESULTS

Application of the Methodology and Design principles led to the development of the open GUI for the hypothetical touchscreen smartphone. This section outlines these results in terms of implementation in the debugging environment and on board implementation. The majority of this section will consist of images which illustrate the GUI, thus materializing the blueprints presented in the Design section of this paper.

When the GUI is launched, the Home Screen first appears; it is at the highest level in the hierarchy. A fully functioning digital clock displaying hours and minutes with a blinking colon was implemented as a custom widget – the code for this was obtained online as open source code under the BSD license. In the Qt IDE, the docked application icons were implemented using QPushButtons, and the images of these buttons were set as .png images, thus enhancing authenticity of the icons.

Note, also, the inclusion of the indicators at the top of the screen as explained previously. These were implemented as pixmap images in QLabel widgets in Qt. Figure 3 illustrates the final results of the GUI Home Screen.

Fig 3: Home Screen Aspect of the GUI

The main menu application consisted mainly of a grid layout of QPushButton icons. As before, the indicators at the top of the screen were implemented using QLabelpixmaps; however, in this screen, the indicators area now displays a fully

ISSN 22773061

functional digital clock which reflects the time of the system clock of the Mini2440. A scroll area was implemented to ensure that all icons were visible. Additionally, the sub-menus were implemented in the same manner. Figures 4 and 5 illustrate these results.Note the inclusion of navigation in the Home icon and the Back buttons.

Figure 4: Main Menu of the GUI

Figure 5: Sub Menus of the GUI

In the Phone application, the dial pad and associated controls were laid out as specified previously. In order to customize the appearance of the buttons, style sheets were used to add colours and gradients. In order to capitalize on screen space, while enhancing usability, the dial pad buttons were set to 80px × 50px. When a number is called, the call screen is raised. This screen is actually the main window of another class, which was instantiated within the dialing interface class. Figure 6 shows this aspect of the GUI.

Figure 6: GUI Implementation of Dial Interface and Calling Interface

In the implementation of the SMS Messaging interface, the focus was on developing a virtual keyboard. In the keyboard, each key was implemented as a QPushButton. To increase input speed, certain basic punctuation marks, such as a period and comma were included. This keyboard allows for upper case letters to be input via the shift button, and as mentioned previously, numbers and symbols also. Figure 7 shows the Messaging application in action.

Figure 7: SMS Messaging Application showing the Default Interface (Left) and Number-Symbol Keypad (Right)

The implementation of the Browser application is shown in Figure 8. Note that due to the lack on Internet connectivity, the Google homepage is only a simulation, and as such, an image was superimposed for the sake of authenticity. Qt has a tab widget, which was used to mimic the tab system of typical browsers today. All other browser controls were implemented as push buttons with images.

Figure 8: Browser Application

5.1. Implementation of the GUI on the Mini2440

It was necessary to conduct on-board testing and implementation of the GUI to determine its operation within the Mini2440 environment, since different platforms execute applications differently, which may lead to unexpected results. Additionally, certain features of the GUI were native only to the Mini2440, for example touch events (as opposed to debugging with mouse clicks), and the use of LEDs. To carry out testing on the Mini2440, the executable files were transplanted to the device via the USB flash drive and the Home Screen application (which can effectively call other applications) was launched.

It is important to note anomalies between the appearance of the GUI on the device, as opposed to within the IDE. The most striking difference in appearance was that of brightness, contrast and colour. While the layout of controls remained fairly constant, in comparison to the debugging results, text was significantly smaller than expected, almost indecipherable.

Applications were tested individually, to gauge the performance of each. If changes were necessary, the programs were modified accordingly before releasing the final version. Integration testing was then conducted to test the interactivity of applications within the GUI. Testing was imperative to determine the performance of the GUI on the device in terms of speed and responsiveness. When the GUI is launched, it is important to note that it is relatively unresponsive. Firstly, in terms of touch events, when a push button is touched, the time taken for the corresponding reaction to occur is notable. Secondly, within the main menu, the reaction time to scrolling is relatively slow.

Most notable, however, is the time taken to launch an application either from the home screen or main menu. From the home screen, it takes roughly 6-7s to launch the Phone application, and approximately 28s to launch the Main Menu. Conversely, when closing an application, the GUI is surprisingly very responsive, since the previous screen is almost instantaneously shown.

6. DISCUSSION

From the results obtained, it can be said that the entire hierarchy of the touchscreen smartphone GUI was successfully implemented, as per the design specifications. The application of GUI design theory from a myriad of notable sources ensured the usability and flexibility of the interface. Most importantly, the adoption of the open source philosophy led to flexibility and cost savings in development, as will soon be outlined.

6.1. Effects of GUI Layout on Usability

In analyzing the application of mobile GUI design theory to this research, consider the factor of touch target sizes. To allow for flexibility of the GUI, several aspects of the interface accounted for the use of both styli and fingers. Therefore, if the software is ported to another platform, say one with a capacitive touchscreen, these aspects would still be operable using finger-input.

In the home screen, main menu and sub-menus, all recommended sizes of the icons were taken into consideration, thus preventing user mishits, due to users with “fat-fingers”. For all icons, the visual representations were of the same size as the corresponding push buttons. Usability was also enhanced by adhering to recommendations for the minimum spacing between adjacent icons; this also contributed to the prevention of mishits. In terms of the home screen and menu, the layout of icons allowed for ambidextrous use of the GUI, thus enhancing flexibility.

Conversely, in the Messages application, button mishits are highly probable due to the size of the keys in the virtual keyboard. However, this is only in the event that the finger pad is used since a stylus would grant the user with higher accuracy. This application therefore poses usability issues and inhibits flexibility. The same can be said for the Browser since the browser controls were deemed too small for ideal finger pad touching, thereby leading to button mishits.

The results of the GUI can also be analyzed from the perspective of gestures. Of the numerous gestures used in typical smartphones, due to the resistive touchscreen used, a minimal subset of gestures was considered, from which only one main gesture, the single tap, was used. Consequently, it can be deduced that the single tap is the most important gesture used in touch screens.

To allow users to traverse the hierarchy of the GUI, soft buttons were utilized. Although useful and overcoming the need for physical buttons, due to the size of the display area of these applications, soft buttons occupied useful space and effectively crowded the GUI. The use of soft buttons in a GUI implemented for a device with a small screen size is therefore discouraged. However, if the screen resolution is higher, as is the case in most modern smartphones, this practice may be valid.

6.2. Flexibility of Using Qt and Relation to Existing Qt GUIs

Since Qt is a cross platform GUI application framework, the code written during development can be cross-compiled and implemented on other target platforms, as long as the qmake tool can be compiled for that specific target. This alludes to a sense of universality of the GUI, which enhances the openness of the final released version, since users can modify and redistribute the software.

Implementations of open source GUIs in the mainstream market include Nokia’s Symbian OS and Maemo OS. The GUI designed in this project was not done so to mimic the Symbian OS, however the tools used are the same and the programming principles were similar. Conversely, if compiled for a Symbian target, this GUI can be launched on any Nokia Symbian platform. Currently, Qt is used in the development of apps for BlackBerry’s latest OS, BlackBerry 10 OS. If ported successfully, this GUI can function as an app which can, in theory, run on a BlackBerry 10 device.

6.3. Effect of Open Source Development Tools on Costs Incurred

In order to analyze and validate the credibility of the menu system hierarchy and the applications developed, and to deduce the effects of the use of open source development tools on overall cost of development, the GUI developed here was compared to the existing Android OS v4.0.4 (Ice Cream Sandwich) GUI as implemented on the Samsung Galaxy SIII – the leading smartphone as of the last quarter of 2012 [7]. Android OS was chosen for comparison since it is the leading open source operating system at present.

The first point of interest is the home screen. The home screen of the Android GUI, like the GUI developed, is at the top level of the GUI hierarchy. Similarities between the two home screens include the docked icons at the base of the screen to launch native applications, the digital clock and the status indicators. Differences include the inclusion of the date, a weather widget and additional launch icons in the Android GUI, as illustrated in Figure 9.

ISSN 22773061

Figure 9: Comparison of Home Screen of Android GUI and Developed GUI

The differences in implementation were due to the disparity in the display resolution of each, with the 720×1280 px and 4.8-inch touchscreen of the Samsung Galaxy SIII having more space to lay out widgets and icons. Consequently, the Mini2440 implementation incorporated several similar aspects within the space limitation.

The main menus of both GUIs were also compared. In each GUI, the main menu can be launched from the home screen. Icons are placed in a grid layout, however, in the case of the Android GUI, due to the hardware used for that smartphone, kinetic scrolling is allowed, and as such, the scrolling mechanisms differ. In the Samsung Galaxy SIII, the user swipes the screen to the left or right to access another set of applications, rather than scrolling up and down. This is shown in Figure 10.

Figure 10: Comparison of the Main Menus of each GUI

The applications developed were compared in a similar manner; however, these comparisons were omitted from this paper to avoid monotony and repetitiveness.

It can easily be shown here that form factor, i.e. the size of the screen greatly affects the layout of a GUI. Based on these comparisons, it can be stated that despite the more comprehensive GUI on the Android OS, using Qt and Embedded Linux, a low-cost implementation is feasible and in the event that the developed GUI is allowed to run on a platform with hardware specifications similar to that of the Samsung Galaxy SIII, there may be potential to penetrate the open source smartphone market. Since the notion of open source software remains the same here, and judging from the similarities in the GUIs, it was shown that hardware effectively increases the price of these smartphones and the fact that the software is open source lends to its popularity.

To further demonstrate the correlation between the bill of materials and cost, consider the following comparisons of the Samsung Galaxy SIII and the Mini2440 on the basis of cost of hardware. Note that while the telecommunications aspect of the Mini2440 was not used here, a GPRS module can be included for use with a SIM card (e.g. a USB modem), and a WLAN driver can be installed for Wi-Fi connectivity, which will lead to it being more similar to an actual smartphone. Also, a camera can be included. Note that between both devices there are several tradeoffs in terms of hardware, for example the faster processor and larger memory resources on the SIII, and the USB, SDCARD and serial ports on the Mini2440.

The Mini2440 is essentially a development board open to many more possibilities than a smartphone. All comparisons are given in US prices.

Consider the core of each device – the microprocessor. The Samsung Galaxy SIII is equipped with the 1.4GHz quad-core Cortex-A9 based Exynos 4412 Quad, while the Mini2440 is equipped with the ARM9 based S3C2440. The Exynos 4412 is listed at US$33.47/piece while the S3C2440 is listed at US$10.31/piece.

Overall, the Mini2440 SDK is listed at roughly US$99.00, with a standard deviation of approximately US$25. This is considered an empty slate which designers and developers can customize as they see fit. On average, as of the fourth quarter of 2012, the Samsung Galaxy SIII was listed at US$399 to US$499, with a plan, varying by handset provider. To further suit the openness offered by the Mini2440, the price of an unlocked Samsung Galaxy SIII was considered; the unlocked smartphone was listed at roughly US$599-US$699, depending on the sales agents. These are online prices and do not take into account shipping and handling.

With this being said, it can clearly be seen that the Embedded Linux platform, coupled with the open tools used for development, in terms of the IDE result in a low-cost implementation of a smartphone GUI.

7. CONCLUSION

After investigations were carried out into the design of a mobile GUI, the implementation was successful, adequately fulfilling the objectives. Based on the outcomes of this work, it can be said that the use of open tools in development of an open source GUI can effectively reduce the overall cost of development. Additionally, with the use of an Embedded Linux platform and an ARM core, low-cost development of a stable, dependable system was also verified. The fact that the SoC was ARM-based alluded to the system being a candidate for today's ARM dominated market. The use of an ARM core facilitated an energy efficient, low power system, possessing high code density and a small silicon footprint.

Factors leading to the hefty prices of smartphones which were eliminated during implementation included the cost of development tools used and bill of materials of hardware. Conversely, due to the poor performance and unresponsiveness of the GUI, the impact on user/customer experience will detract from popularity and innovation. However, these aspects of poor performance were most likely due to the programming approach taken, whereby when calling a lower level application, in Qt, applications were launched and run in parallel, which lead to shared resources between applications. This can be rectified using a smarter programming approach such as one application with different classes.

As per the specifications of the GNU LGPL v2.1, under which the open source version of Qt is licensed, the software produced in this project will be classified as open, upon release, and automatically licensed under the GNU LGPL v2.1.

During this research and development, due to the specifications of the target platform, the performance and usability of the GUI were compromised. Conversely, if the GUI were to be implemented on a device with specifications such as those of today's market-leading smartphones, the performance of the GUI may be just as responsive and usable, and there can even be the potential of permeating the open source market.

The target platform used in this project was the main limitation. Apart from the limited processing and memory resources, the fact that a resistive touchscreen was used hindered design. All multi-touch gestures were discarded during design, thus crippling the profoundness of the GUI. Today, multi-touch gestures are widely used and offer an important avenue to human-machine interaction.

The major shortcoming of this project was the scope of the GUI feature set. A minimal number of applications were implemented, which pales in comparison to existing market-leading GUIs. The fact that development was solely in the portrait orientation limited the use of space on the device and by extension, the flexibility of the GUI. Also, the lack of kinetic features in such actions as scrolling and swiping detracted from the human-machine interactivity of the final product.

Since GUI was implemented and that an avenue has been paved for open GUI development on an Embedded Linux platform, given sufficient time, the work done here can be taken further to develop a full-scale smartphone system, which incorporates the software developed during this research.

REFERENCES

[1] Androustellis-Theotokis, Stephanos, DiomidisSpinellis, Maria Kechagia, and George Gousios, "Open Source Software: A Survey from 10,000 Feet," Foundations and Trends in Technology, Information and Operations Management, 4(3-4), 187-347, 2010. [Online]. Available: http://www.dmst.aueb.gr/dds/pubs/jrnl/2010-TOMS-OSS-Survey/html/ASKG10.pdf. [Accessed Feb. 14, 2013].

[2] Apple Inc., "iOS Human Interface Guidelines", Last modified Dec. 17, 2012, [Accessed Feb. 14, 2013] Available: http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/IconsImages/IconsImages.html#//apple_ref/doc/uid/TP40006556-CH14-SW1.

[3] ARMLtd.ARMProcessorArchitecture,ARMLtd.,2012,[E-book] Available:http://infocenter.arm.com/help/topic/com.arm.doc.ddi0151c/ARM920T_TRM1_S.pdf.

[4] Blanchette, Jasmin and Mark Summerfield, C++ GUI Programming with Qt 4, Prentice Hall, 2008.

[5] Cha, Bonnie, "Smartphones Unlocked: Understanding Processes", Last modified Aug. 8, 2011, [Accessed Jan. 22, 2013] Available: http://www.cnet.com/8301-17918_1-20088704-85/smartphones-unlocked-understanding-processors/.

[6] Clarke, Josh, "Designing for touch", .net Magazine, [Online]. Available: http://www.netmagazine.com/features/designing-touch. [Accessed Jan. 13, 2013].

[7] Columbus, Louis, "2013 Roundup of Smartphone and Tablet Forecast and Market Estimates". Forbes Inc., Last modified Jan. 17, 2013, [Accessed Feb. 5, 2013] Available: http://www.forbes.com/sites/louiscolumbus/2013/01/17/2013-roundup-of-mobility-forecasts-and-market-estimates/.

[8] EngineersGarage, "Embedded Linux: Understanding The Embedded Linux", Last modified Aug. 30, 2012, [Accessed Mar. 3, 2013]

Available:http://www.engineersgarage.com/articles/what-is-embedded-linux.

[9] Furber, Steve, ARM System-on-Chip Architecture, Addison-Wesley Professional, 2000.

[10] International Teleccomunication Union, E.161: International Operation – Numbering Plan of the International Telephone Service, ITU-T, 2001.

[11] ISO/IEC, Information Technology – Screen icons and symbols for personal mobile communication devices, 2012.

[12] Meredith, Wade, "Best Practices of Touch Screen Interface Design", Last modified Aug. 16, 2008, [Accessed Nov. 20, 2012] Available: http://voltagecreative.com/articles/best-practices-of-touch-screen-interface-design/.

[13] MicroARM Systems Inc., Mini2440 User's Manual, MicroARM Systems Inc., 2009

[14] Miller, Michael J., "Smartphone Processors Getting Smarter", PCMAG, [Online]

Available:http://www.pcmag.com/article2/0,2817,2360152,00.asp[Accessed Jan. 5, 2013].

[15] Nakagawa, Takao, and Hideteke Uwano, "Usability Differential in Positions of Software Keyboard on Smartphone", in The 1st IEEE Global Conference on Consumer Electronics 2012, IEEE.

[16] Nokia Corporation, "Touchscreen Usability", Last modified May 9, 2012, [Accessed Feb. 17, 2013], Available: http://www.developer.nokia.com/Community/Wiki/TouchScreen_Usability.

[17] O2, "Making calls has become fifth most frequent use for a Smartphone for newly-networked generation of users", Last modified Jun. 29, 2012, [Accessed Mar. 3, 2013] Available: http://news.o2.co.uk/?press-release=Making-calls-has-become-fifth-most-frequent-use-for-a-Smartphone-for-newly-networked-generation-of-users

[18] Open Source Initiative, "The Open Source Definition", [Accessed Dec. 29, 2012] Available: http://opensource.org/osd

[19] Rosen, Lawrence, Open Source Licensing – Software Freedom and Intellectual Property Law, Prentice Hall, Upper Sadde River, New Jersey, 2004.

[20] Ryzhyk, Leonid, "The ARM Architecture", University of New South Wales, 2006.

[21] Samsung Electronics Co. Ltd., Installation Manual for S3C2440, Samsung Electronics Co. Ltd., Yonging-City, Gyeonggi-Do, Korea, 2004.

[22] Schiesser, Tim, "Guide to Smartphone Hardware: Processors", Last modified Feb. 12, 2012, [Accessed Oct. 5, 2012] Available: http://www.neowin.net/news/guide-to-smartphone-hardware-17-processors.

[23] Seng, Jacqueline, "The Emerging Smartphone OS Battle: Firefox vs. Tizen vs. Ubuntu", [Accessed Mar. 5, 2013] Available: http://asia.cnet.com/the-emerging-smartphone-os-battle-firefox-vs-tizen-vs-ubuntu-62220705.htm.

[24] Todish, Tim R., "Not Your Parent's Mobile Phone: UX Design Guidelines For Smartphones", Last modified Oct. 6, 2011, [Accessed Nov. 26, 2012]

Available: http://uxdesign.smashingmagazine.com/2011/10/06/not-your-parents-mobile-phone-ux-design-guidelines-smartphones/.

[25] Villamour, Craig, Dan Willis and Luke Wroblewski, "Touch Gesture Refrence Guide", [Accessed Jan. 13, 2013] Available: http://static.lukew.com/TouchGestureGuide.pdf.

[26] Wangfengzhu, Miaoliang, ZhangleiZhangming, and Suichunrong, "Research and Design of Smartphone System based on ARM9 and Embedded Linux Operating System", in 2011 International Conference on Computational and Information Sciences, IEEE Computer Society.

[27] Westaway, Luke, "Asus explains why smartphones are more expensive than the $249 FonePad", CNET, Last modified Feb. 25, 2013, [Accessed Mar. 15, 2013] Available: http://reviews.cnet.com/8301-13970_7-57571119-78/asus-explains-why-smartphones-are-more-expensive-than-the-$249-fonepad/.

[28] Wroblewski, Luke, "Touch Target Sizes", Last modified May 4, 2010, [Accessed Feb. 15, 2013] Available: http://www.lukew.com/ff/entry.asp?1085.

[29] XenStreet, “The Vibrant Open Source Smartphone OS Market”, Last modified Mar. 3, 2013, [Accessed Mar. 15, 2013] Available:http://www.xenstreet.com/2013/03/the-vibrant-open-source-smartphone-os-market/.

[30] Yun, Min-Hong, Do-Hyung Kim, and Sun-Ja Kim, “Experience of Linux and GTK+2 Smartphone”, in 3G Mobile Communication Technologies, IEEE.

Author’ biography with Photo

Daniel Sooknanan has received his B.Sc. (Hons) in Electrical and Computer Engineering from The University of the West Indies, Trinidad and Tobago, in 2013 and is currently an M.Phil student at the University. He has been an IEEE member since 2010. His work has centered on programming and software and, for his M.Phil, in the field of High Performance Computing.

Ajay Joshi holds a Ph.D. in Advanced Computer Architecture, from University of Mumbai, India in 1996. He is currently with the Department of Electrical and Computer Engineering at The University of the West Indies, Trinidad and Tobago. Earlier, he was Head of Technology at IBM Advanced Training Cenntre at Nasik, India. His publications are in the field of Embedded Systems Hardware related to memory and parallel processing. He is an IEEE member since 2004 and has chaired a session in IADAT conference in Spain. Currently, he is a Principal Investigator of NVIDIA CUDA centre at the University.